

OPTIMIZING MULTI-HOP KNOWLEDGE GRAPH REASONING BASED-ON FOURIER-KNOWLEDGE GRAPH EMBEDDING & REINFORCEMENT LEARNING

TỐI ƯU HÓA SUY DIỄN ĐỒ THỊ TRI THỨC ĐA BƯỚC DỰA TRÊN NHÚNG ĐỒ THỊ FOURIER VÀ PHƯƠNG PHÁP
HỌC TĂNG CƯỜNG

 **LE, Nhut-Nam***

Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam
Vietnam National University, Ho Chi Minh City, Vietnam
227, District 5, Ho Chi Minh City, Vietnam
nam.lnhut@gmail.com

March 12, 2023

Abstract

Các thực thể trong thế giới này có thể được tổ chức thành một đồ thị mà quan hệ giữa những thực thể có những kiểu khác nhau này có thể là những cạnh với những kiểu khác nhau. Đó là đồ thị tri thức. Dữ liệu đồ thị tri thức không bao giờ có thể hoàn thiện. Hiện nay, có nhiều phương pháp đề xuất để mà cố gắng hoàn thiện hay khám phá những dữ liệu chưa được nhìn thấy (unseen facts) hay những mối quan hệ tiềm tàng (latent relationships) bên trong loại dữ liệu này. Một trong những phương pháp tiếp cận hiện nay cho vấn đề này là suy luận đồ thị tri thức đa bước (multi-hop knowledge graph reasoning). Quá trình thực thi của phương pháp này có thể thể hiện như một bài toán quyết định xâu chuỗi (serilized decision problem), và có thể được giải quyết bằng phương pháp học tăng cường (reinforcement learning). Dựa trên công trình đã được công bố trong khoảng thời gian gần đây, RL-based multi-hop KG reasoning model Path Additional Action space Ranking (PAAR), trong bản technical report này, chúng tôi đề xuất mô hình cải tiến cho PAAR dựa trên những đồ thị tri thức Fourier (Fourier-Knowledge Graph Embeddig) và tối ưu hóa cho quá trình học các bản nhúng hiệu quả hơn thông qua phương pháp Quasi-hyperbolic momentum và Adam. Để thêm vào những bản nhúng hữu ích hơn, các vector Fourier-KGE được thêm vào không gian trạng thái và giúp cho việc cải thiện tính thể hiện của không gian trạng thái. Tương tự như PAAR, chúng tôi giải quyết vấn đề thừa phần thưởng (reward sparsity problem) trong học tăng cường bằng cách sử dụng hàm tính điểm (score function) của Fourier-KGE như một phần thưởng mềm (soft-reward). Các kết quả thực nghiệm được báo cáo lại thành dạng bảng dựa trên các bộ dữ liệu thí nghiệm và tái thực nghiệm kết quả của bài báo gốc.

Keywords Optimization methods · Multi-hop reasoning · Knowledge graph embedding · Fourier transformation · Reinforcement learning

*Master Computer Science Student K32/ BS-MS Programme K18

Contents

1	Introduction	4
2	Related works	4
2.1	Knowledge graph reasoning	4
2.2	Multi-hop KG reasoning	5
2.3	Deep reinforcement learning reasoning	5
3	Backgrounds & preliminaries knowledge	5
3.1	Knowledge graphs	5
3.2	Knowledge graph embedding	6
3.2.1	Embedding vectors	6
3.2.2	Score function	6
3.2.3	Negative sampling strategy	6
3.2.4	Loss function	6
3.3	Multi-hop knowledge graph reasoning	6
3.4	Background for optimizations	7
3.4.1	Derivatives	7
3.4.2	Gradients	7
3.4.3	Convex optimization problem	8
3.4.4	Generalized algorithm: Iterative Local Descent Method	9
3.5	Fourier transformation	9
4	Methodology	10
4.1	Reinforcement Learning Framework for Multi-hop KG Reasoning	11
4.1.1	State space	11
4.1.2	Action space	11
4.1.3	Transition	11
4.1.4	Reward	11
4.1.5	Policy network	11
4.2	Fourier-Knowledge graph embedding	11
4.3	Optimization for learning KGE	11
4.3.1	Adaptive Moment Estimation Method	11
4.3.2	Quasi-Hyperbolic Momentum & Adam	11
4.4	Designing for Action space	12
4.5	Designing for State space	12
4.6	Designing for Policy networks	12
5	Experiments	12
5.1	Datasets, metrics, and settings	12

5.1.1	Datasets	12
5.1.2	Metrics	12
5.1.3	Settings	14
5.2	Experimental results	15
6	Conclusion	16

1 Introduction

Hiện nay, máy học với dữ liệu đồ thị và các ứng dụng của nó nhận được nhiều sự chú ý từ lẫn cộng đồng nghiên cứu và doanh nghiệp. Lấy động lực từ sự tiềm năng của cấu trúc dữ liệu đồ thị như một ngôn ngữ phổ phát cho việc mô tả và phân tích nhiều cấu trúc dữ liệu thế giới thật như cấu trúc phân tử, cấu trúc khung xương con người, mạng máy tính, mạng thức ăn, mạng xã hội hay mạng lưới nơ-ron trong não bộ con người, đồ thị trở thành một công cụ hiệu quả cho nghiên cứu lẫn ứng dụng vào nhiều lĩnh vực khác nhau. Dữ liệu đồ thị đa quan hệ như đồ thị tri thức đóng vai trò quan trọng vì nó mô tả và biểu diễn những miền tri thức phức tạp. Khai thác những loại dữ liệu như thế làm tiền đề cho nhiều ứng dụng phía sau. Tuy nhiên, các vấn đề phải đối mặt với dữ liệu đồ thị rất nhiều và phức tạp, ví dụ như: độ phức tạp về không gian và thời gian của mô hình; hay vấn đề biểu diễn đồ thị (mạng nói chung) sao cho bảo toàn cấu trúc và những thông tin tương tác tiềm ẩn bên trong nó.

Dữ liệu đồ thị tri thức thường được khởi tạo và làm giàu từ nhiều nguồn tài nguyên khác nhau (các nguồn tài nguyên phải đảm bảo tính hợp pháp) như ngữ liệu văn bản, các nguồn dữ liệu có cấu trúc như JSON, XML, hay CSV. Do những kỹ thuật thu thập, xử lý và khởi tạo cho đồ thị tri thức sơ khai thường dựa trên các cơ chế thủ công hoặc bán tự động, thế nên nó thường không hoàn thiện. Để giải quyết vấn đề này, một trong các cách tiếp cận hiện nay là suy diễn đồ thị tri thức (knowledge graph reasoning). Một trong nhiều các phương pháp cho suy diễn đồ thị tri thức là nhúng đồ thị tri thức (knowledge graph embedding). Phương pháp này xây dựng hàm ánh xạ những bộ ba dữ kiện (h, r, t) vào không gian vector liên tục thấp chiều trong khi vẫn bảo toàn cấu trúc nội tại của những đối tượng bên trong đồ thị (Ở đây, đối tượng bao gồm những thực thể và quan hệ liên kết giữa chúng).

Xem xét bài toán với truy vấn $(?, r, t)$, các mô hình nhúng đồ thị tính toán vector đặc trưng (representation vector) \mathbf{v} từ những thông tin sẵn có. Quá trình dự đoán liên kết được thực thi bằng cách truy vấn những thực thể mục tiêu \mathbf{h} nào có vector đặc trưng "gần" với \mathbf{v} nhất. Tuy nhiên, phương pháp có nhược điểm khi ngăn chặn tính kết hợp giữa các quan hệ. Để giải quyết vấn đề này, bài toán suy diễn đồ thị phát triển thành dạng suy diễn những bộ ba dữ kiện bị thiếu bằng những thông tin tổng hợp được từ các đường đi đa bước (multi-hop paths). Quá trình suy diễn đa bước này có thể được biểu diễn thành bài toán chuỗi quyết định (serialized decision problem) và hoàn toàn có thể giải quyết với các mô hình học tăng cường.

Lấy cảm hứng từ tác vụ xử lý ảnh trong miền tần số, phép biến đổi Fourier cho ta nhiều lợi ích trong việc tính toán và xử lý dữ liệu ảnh số. Một cách cụ thể, ta dễ dàng lọc những tần số không cần thiết (thông tin nhiễu) và thực hiện các xử lý trong miền tần số nhanh hơn rất nhiều so với miền không gian. Dựa trên công trình multi-hop KG reasoning PAAR (Path Additional Action-space Ranking), chúng tôi xây dựng và tối ưu mô hình trong miền tần số với mô hình nhúng đồ thị Fourier. Một cách cụ thể, trong bản technical report, chúng tôi đóng góp:

- Đề xuất mô hình nhúng đồ thị trong miền tần số sử dụng phép biến đổi Fourier (Fourier-Knowledge Graph Embedding), tăng tốc độ xử lý và tính toán lọc bỏ nhiễu dựa trên miền không gian mới.
- Cải thiện tối ưu hóa mô hình bằng cách sử dụng Quasi-Hyperbolic momentum và Adam.
- Thực hiện và báo cáo lại các kết quả trên các tập dữ liệu phòng thí nghiệm bao gồm FB15k-237 10% và FB15k-237 20%. Mã nguồn thực hiện đề tài được công khai ở Github²

2 Related works

Trong phần này, báo cáo khảo sát các công trình liên quan từ các phương pháp: knowledge graph reasoning, multi-hop KG reasoning, và deep reinforcement learning reasoning.

2.1 Knowledge graph reasoning

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque

²<https://github.com/m32us/RL4MRD>

cursus luctus mauris. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2.2 Multi-hop KG reasoning

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2.3 Deep reinforcement learning reasoning

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

3 Backgrounds & preliminaries knowledge

Trong phần này, chúng tôi hệ thống lại các khái niệm nền tảng và cơ sở, bao gồm đồ thị tri thức, nhúng đồ thị tri thức, suy diễn đồ thị tri thức đa bước, nền tảng cho các phương pháp tối ưu hóa, và phép biến đổi Fourier.

3.1 Knowledge graphs

Đồ thị tri thức là một loại dữ liệu đa quan hệ được tổ chức dưới dạng đồ thị, mà trong đó các nút thể hiện những thực thể và các cạnh nối giữa các nút này thể hiện mối quan hệ giữa chúng. Trong bản technical report này, chúng tôi hình thức hóa dạng toán cho đồ thị tri thức dưới dạng toán học như sau $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{F}\}$, trong đó \mathcal{E} , \mathcal{R} , và \mathcal{F} lần lượt là tập hợp các thực thể, quan hệ, và bộ ba dữ kiện dưới dạng (h, r, t) . Ví dụ: (Hà Nội, là thủ đô, Việt Nam)

3.2 Knowledge graph embedding

Một mô hình nhúng đồ thị tri thức thông thường bao gồm bốn thành phần chính yếu: các vector nhúng (embedding vectors), hàm tính điểm (score functions), chiến lược (phát sinh) lấy mẫu âm (negative sampling strategy), và hàm mất mát (lossfunction).

3.2.1 Embedding vectors

Mục tiêu của nhúng đồ thị tri thức là thể hiện dữ liệu đồ thị tri thức ở không gian (vectors) liên tục thấp chiều (low-dimensional continuous space) mà vẫn bảo toàn cấu trúc đồ thị nhiều nhất có thể.

Biểu diễn nhúng đồ thị ngây thơ (naive approach) là biểu diễn ma trận kề mà sử dụng một ma trận vuông có kích thước bình phương số lượng thực thể. Điều này gây ra tiêu tốn quá nhiều bộ nhớ, chưa kể dữ liệu đa quan hệ phức tạp hơn rất nhiều so với đồ thị tổng quát.

Phương pháp nhúng đồ thị tri thức sử dụng các vector nhúng số thực cho các thành phần của đồ thị tri thức, $(\mathbf{h}, \mathbf{r}, \mathbf{t}) \in \mathbb{R}^d$, trong đó d là số chiều nhúng cho mỗi vectors.

3.2.2 Score function

Hàm tính điểm trong mô hình nhúng đồ thị đóng vai trò đo tính hợp lý của một bộ ba dữ kiện, ký hiệu là $f_r(\mathbf{h}, \mathbf{r}, \mathbf{t})$. Dựa trên các công trình hiện nay, có hai loại hàm tính điểm chính: dựa trên khoảng cách (tức là các khoảng cách thông dụng như Euclidean distance, hay Manhattan distance) và dựa trên mức độ tương đồng (cosine similarity, hay Jaccard similarity).

- Một hàm tính điểm dựa trên khoảng cách tính toán mức độ hợp lý của một bộ ba bằng cách sử dụng metric khoảng cách để tính toán khoảng cách giữa thực thể nguồn và thực thể mục tiêu.
- Một hàm tính điểm dựa trên mức độ tương đồng tính toán mức độ hợp lý của một bộ ba dữ kiện bằng cách sử dụng các phương pháp khớp ngữ nghĩa (dựa trên phép tích vô hướng)

Nếu một bộ ba dữ kiện có tính hợp lý cao, điểm số của nó càng cao.

3.2.3 Negative sampling strategy

Đồ thị tri thức luôn phải chứa những tri thức đúng đắn về thế giới này. Thế nào là tri thức đúng đắn? Thế nào là tri thức? Tri thức là những gì chúng ta tin là đúng. Do vậy, những dữ liệu để xây dựng đồ thị tri thức cần phải xác minh một cách rõ ràng. Nhưng vấn đề nảy sinh ở đây là khi áp dụng các phương pháp học máy để giải quyết các bài toán khai thác dữ liệu, chúng ta không có mẫu âm. Các chiến lược phát sinh mẫu âm được đề xuất để phát sinh mẫu âm một cách hợp lý cho các mô hình học. Một số cách tiếp cận truyền thống sử dụng các phân phối như Uniform hay Bernoulli negative sampling hoặc Adversarial negative sampling.

3.2.4 Loss function

Quá trình học là một quá trình tối ưu một hàm mục tiêu. Mô hình nhúng đồ thị tri thức cố gắng hiệu chỉnh các bản nhúng sao cho những mẫu dương luôn có điểm số cao hơn các mẫu âm. Và đó, hiển nhiên các phương pháp dựa trên local descent được sử dụng như Adam, AdaGrad. Ở phần phương pháp đề xuất, chúng tôi sử dụng pháp Quasi-hyperbolic momentum kết hợp với Adam để tối ưu hóa hàm mục tiêu.

3.3 Multi-hop knowledge graph reasoning

Bài toán suy diễn đồ thị tri thức đa bước (Multi-hop knowledge graph reasoning) được phát biểu như sau: Cho trước một mệnh đề truy vấn (query statement), mệnh đề này có thể một trong ba kiểu khác nhau: $(h, r, ?)$, $(?, r, t)$, hay $(h, ?, t)$. Mục tiêu của bài toán là tìm kiếm những thực thể tiềm năng có thể thay thế những vị trí bị thiếu mất trong bộ ba dữ kiện cho trước trong một đường dẫn suy diễn k bước (k -hop reasoning path) $e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} e_3 \cdots \xrightarrow{r_k} e_{k+1}$

3.4 Background for optimizations

Các phương pháp tối ưu (optimization methods) đối mặt với vấn đề tìm điểm (design point) mà cực tiểu (hoặc cực đại - tùy vào bài toán đang xem xét) một hàm mục tiêu (objective function). Việc biết được giá trị của một hàm số thay đổi như thế với các đầu vào của nó được xem là một điều cần thiết và hữu ích. Bởi vì điều này cho chúng ta biết hướng nào để mà di chuyển đến gần với điểm tối ưu hơn so với vị trí trước đó. Sự thay đổi trong giá trị của hàm số được đo bằng cách tính toán đạo hàm trong một trường hợp hàm một biến và tính toán gradient trong trường hợp hàm nhiều biến.

3.4.1 Derivatives

Đạo hàm (derivative) $f'(x)$ của một hàm f một biến x được hiểu là tỉ lệ mà giá trị của hàm số thay đổi tại x . Đạo hàm có thể được sử dụng để cho ta một xấp xỉ tuyến tính (linear approximation) của một hàm số gần x , cụ thể:

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x \quad (1)$$

Một cách toán học, đạo hàm là tỉ lệ giữa sự thay đổi trong f và sự thay đổi trong x tại điểm x :

$$f'(x) = \frac{\Delta f(x)}{\Delta x} \equiv \frac{df(x)}{dx} \quad (2)$$

mà đó là sự thay đổi trong $f(x)$ được chia bởi sự thay đổi trong x khi bước nhảy trở nên vô cùng nhỏ. Lưu ý: ký hiệu $f'(x)$ dựa theo Lagrange, $df(x)/dx$ dựa theo Leibniz.

Định nghĩa dựa trên giới hạn của đạo hàm có thể được biểu diễn theo ba cách khác nhau, bao gồm: hiệu tiến (forward difference), hiệu trung tâm (central difference) và hiệu ngược (backward difference)

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} = \lim_{h \rightarrow 0} \frac{f(x+h/2) - f(x-h/2)}{h} = \lim_{h \rightarrow 0} \frac{f(x) - f(x-h)}{h} \quad (3)$$

3.4.2 Gradients

Xem xét không gian Euclidean có d chiều, ký hiệu là \mathbb{R}^d . Các vector trong không gian ký hiệu là \mathbf{x} với giá trị trong chiều thứ i là x_i . Tích vô hướng giữa hai vector \mathbf{x} và \mathbf{y} được ký hiệu $\langle \mathbf{x}, \mathbf{y} \rangle$.

Gọi hàm $f : \mathbb{R}^d \rightarrow \mathbb{R}$ là một hàm số nhiều biến liên tục. Nếu f là một hàm khả vi (nhiều biến liên tục khả vi - differentiable), gradient của f tại điểm \mathbf{x} , ký hiệu là $\nabla f(x)$, là một vector có d chiều, trong đó chiều thứ i là đạo hàm riêng (partial derivative) $\frac{\partial f}{\partial x_i}$ tại \mathbf{x} .

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \dots, \frac{\partial f(x)}{\partial x_n} \right] \quad (4)$$

Nếu hàm số f có đạo hàm cấp hai, ma trận Hessian của f tại \mathbf{x} , ký hiệu là $\nabla^2 f(x)$ là một ma trận vuông có kích thước $d \times d$; hàng thứ i và cột thứ j có giá trị:

$$\nabla^2 f(x)[i, j] = \frac{\partial^2 f}{\partial x_i \partial x_j} \quad (5)$$

Ma trận Hessian có dạng như sau:

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1}, \frac{\partial^2 f(x)}{\partial x_1 \partial x_2}, \dots, \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1}, \frac{\partial^2 f(x)}{\partial x_n \partial x_2}, \dots, \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{bmatrix} \quad (6)$$

Đạo hàm có hướng $\nabla_{\mathbf{s}} f(\mathbf{x})$ là tỷ lệ thay đổi tức thời của $f(\mathbf{x})$ khi \mathbf{x} di chuyển với vận tốc \mathbf{s} , được định nghĩa theo giới hạn như sau:

$$\nabla_{\mathbf{s}} f(x) = \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{s}) - f(\mathbf{x})}{h} = \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{s}/2) - f(\mathbf{x} - h\mathbf{s}/2)}{h} = \lim_{h \rightarrow 0} \frac{f(\mathbf{x}) - f(\mathbf{x} - h\mathbf{s})}{h} \quad (7)$$

Sử dụng gradient cho ta biểu thức gọn hơn

$$\nabla_{\mathbf{s}} f(x) = \nabla f(x)^\top \mathbf{s} \quad (8)$$

3.4.3 Convex optimization problem

Một tập $\mathcal{X} \subseteq \mathbb{R}^d$ là tập lồi nếu

$$t\mathbf{x} + (1-t)\mathbf{y} \in \mathcal{X} \quad (9)$$

với mọi $\mathbf{x}, \mathbf{y} \in \text{dom} f$ và $t \in [0, 1]$. Điểm $t\mathbf{x} + (1-t)\mathbf{y}$ được gọi là một tổ hợp lồi (convex combination) của hai điểm \mathbf{x} và \mathbf{y} .

Một hàm f là hàm lồi nếu

$$f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)f(\mathbf{y}) \quad (10)$$

với mọi $\mathbf{x}, \mathbf{y} \in \text{dom} f$ và $t \in [0, 1]$.

Điều kiện trên được gọi là điều kiện lồi bậc không (zeroth-order condition) bởi vì điều kiện này không liên quan tới tính toán đạo hàm hay gradient. Ngoài điều kiện bậc không, ta còn có điều kiện lồi bậc một (first-order condition): Hàm khả vi f là một hàm lồi nếu:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle, \forall \mathbf{x}, \mathbf{y} \in \mathcal{X} \quad (11)$$

Đối với các hàm khả vi, điều kiện lồi bậc không và bậc một là tương đương. Từ đó, ta có thể phát biểu bổ đề như sau

Bổ đề 1. Nếu f là hàm khả vi trên tập xác định của nó thì điều kiện lồi bậc không và điều kiện lồi bậc hai là tương đương.

Bên cạnh đó, khi xem xét các hàm có đạo hàm cấp hai thì ta còn có điều kiện lồi bậc hai: Hàm f có đạo hàm cấp hai là một hàm lồi nếu $\nabla^2 f(\mathbf{x})$ là ma trận nửa xác định dương (positive semi-definite)

Nếu bất đẳng thức nghiêm ngặt (tức là $<$ thay thì \leq) cho tất cả $t \in [0, 1]$ và $\mathbf{x} \neq \mathbf{y}$, thì ta nói f là hàm lồi nghiêm ngặt (strictly convex)

Một hàm f là một hàm lồi mạnh (strongly convex) với tham số m nếu hàm

$$\mathbf{x} \mapsto f(\mathbf{x}) - \frac{m}{2} \|\mathbf{x}\|_2^2 \quad (12)$$

Nếu bất đẳng thức là nghiêm ngặt, tức là $<$ thay vì \leq với mọi $t \in (0, 1)$ và $\mathbf{x} \neq \mathbf{y}$, thì ta nói rằng f là lồi nghiêm ngặt (strictly convex).

Một hàm f là lồi mạnh với tham số m (m -strongly convex) nếu hàm

$$\mathbf{x} \mapsto f(\mathbf{x}) - \frac{m}{2} \|\mathbf{x}\|_2^2 \quad (13)$$

là hàm lồi.

Những điều kiện này cho thấy: tính lồi mạnh ám chỉ lồi nghiêm ngặt. Một cách hình học, tính lồi có nghĩa là đường thẳng phân chia giữa hai điểm trên một đồ thị của hàm f nằm trên hoặc trên đồ thị đó. Lồi nghiêm ngặt có nghĩa là đường thẳng phân đoạn nằm trên đồ thị một cách nghiêm ngặt, ngoại trừ những điểm chặn (endpoints).

Điều gì khiến ta lại quan tâm đến một hàm có phải lồi (nghiêm ngặt/ mạnh) hay không? Một cách đơn giản, trên nhiều định nghĩa của chúng ta về tính lồi ám chỉ bản chất của cực tiểu. Nó không có gì ngạc nhiên mà những điều kiện mạnh hơn nói cho chúng ta nhiều hơn về cực tiểu. Tính lồi cho chúng ta những hệ quả được phát biểu dưới dạng mệnh đề như sau (ở đây không chứng minh các mệnh đề này)

Mệnh đề 1. Cho \mathcal{X} là một tập lồi. Nếu f là một hàm lồi, thì bất kỳ cực tiểu địa phương (local minimum) nào của f trong \mathcal{X} cũng là một cực tiểu toàn cục (global minimum)

Mệnh đề 2. Cho \mathcal{X} là một tập lồi. Nếu f là một hàm lồi nghiêm ngặt (strictly convex), thì tồn tại ít nhất một cực tiểu địa phương của f trong \mathcal{X} . Hệ quả là, nếu nó tồn tại, nó là một cực tiểu toàn cục duy nhất của f trong \mathcal{X}

Bài toán tối ưu lồi được phát biểu như sau: Cho một hàm lồi $f : \mathbb{R}^d \rightarrow \mathbb{R}$ với tập xác định lồi K , bài toán này tìm một điểm $\hat{\mathbf{x}}$ sao cho:

$$f(\hat{\mathbf{x}}) - \min_{\mathbf{x} \in K} f \leq \epsilon \quad (14)$$

với ϵ là một số thực cho trước, nhỏ hơn 1.

Bài toán tối ưu lồi có hai biến thể: bài toán tối ưu lồi không ràng buộc (unconstrained convex optimization problem) và bài toán tối ưu lồi có ràng buộc (constrained convex optimization problem)

- Bài toán tối ưu lồi không ràng buộc: Nếu tập xác định lồi K là toàn bộ không gian \mathbb{R}^d
- Bài toán tối ưu lồi có ràng buộc: Nếu tập xác định lồi K không là toàn bộ không gian \mathbb{R}^d

Để tìm được một tối ưu, ta cần phải biết được tính chất của nó. Trong trường hợp bài toán tối ưu lồi không ràng buộc, tức là $K = \mathbb{R}^d$, điểm \mathbf{x}^* tối ưu sẽ thỏa mãn tính chất gradient của f tại \mathbf{x}^* là 0 (0 này vector 0)

$$\nabla f(\mathbf{x}^*) = \mathbf{0} \quad (15)$$

Trong trường hợp bài toán tối ưu lồi có ràng buộc, tức là $K \neq \mathbb{R}^d$, gradient của f tại \mathbf{x}^* có thể không bằng 0. Thì \mathbf{x}^* sẽ thỏa mãn tính chất

$$\langle \nabla f(\mathbf{x}^*), \mathbf{x} - \mathbf{x}^* \rangle \geq 0, \forall \mathbf{x} \in K \quad (16)$$

3.4.4 Generalized algorithm: Iterative Local Descent Method

Phương pháp phổ thông để tiếp cận bài toán tối ưu là cập nhật một điểm thiết kế (design point) \mathbf{x} từng bước mà cực tiểu giá trị hàm mục tiêu dựa trên một mô hình cục bộ (local model). Các mô hình cục bộ có thể được phát triển dựa trên nhiều hướng tiếp cận như tiếp cận bậc nhất (first-order approach); tiếp cận bậc hai (second-order) xấp xỉ Taylor.

Các thuật toán tối ưu được thiết kế theo cách tiếp cận tổng quát, được gọi là descent direction methods. Chúng bắt đầu với một điểm thiết kế ban đầu (điểm khởi tạo ban đầu) $\mathbf{x}^{(1)}$ và sau đó sinh ra một chuỗi các điểm, ta gọi là lần lặp (iterates), để mà hội tụ về một cực bộ địa phương (local minimum).

Algorithm 1 Thủ tục iterative descent direction

- 1: Kiểm tra liệu $\mathbf{x}^{(k)}$ có thỏa mãn điều kiện dừng hay không. Nếu thỏa mãn thì dừng thuật toán; không thỏa thì tiếp tục chuyển đến bước tiếp theo
 - 2: Xác định hướng đi xuống (descent direction) $\mathbf{d}^{(k)}$ bằng cách sử dụng thông tin cục bộ như Gradient cho những phương pháp first-order (hay Hessian cho những phương pháp second-order)
 - 3: Xác định kích thước của bước nhảy hay tốc độ học (learning rate) $\alpha^{(k)}$
 - 4: Tính toán điểm thiết kế kế tiếp dựa trên cập nhật
-

Với các phương pháp khác nhau, chúng sẽ có cách riêng trong việc quyết định tốc độ học (tỷ lệ học) α và \mathbf{d}

3.5 Fourier transformation

Biến đổi Fourier là một phiên bản tổng quát hóa của chuỗi (phức) Fourier (complex Fourier series) trong giới hạn $L \rightarrow \infty$. Xem xét một hàm thực $f(x)$, ta có:

$$f(x) = \sum_{n=-\infty}^{\infty} A_n e^{inx} \quad (17)$$

Biến đổi

$$\begin{aligned}
\int_{-\pi}^{\pi} f(x) e^{-imx} dx &= \int_{-\pi}^{\pi} \left(\sum_{n=-\infty}^{\infty} A_n e^{inx} \right) e^{-imx} dx \\
&= \sum_{n=-\infty}^{\infty} A_n \int_{-\pi}^{\pi} e^{i(m-n)x} dx \\
&= \sum_{n=-\infty}^{\infty} A_n \int_{-\pi}^{\pi} (\cos((n-m)x) + i \sin((n-m)x)) dx \\
&= \sum_{n=-\infty}^{\infty} A_n 2\pi \delta_{mn} \\
&= 2\pi A_m
\end{aligned} \tag{18}$$

Do đó:

$$A_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-inx} dx \tag{19}$$

Với một hàm tuần hoàn (function periodic) trong $[-L/2, L/2]$, ta có:

$$f(x) = \sum_{n=-\infty}^{\infty} A_n e^{i(2\pi nx/L)} \tag{20}$$

$$A_n = \frac{1}{L} \int_{-L/2}^{L/2} f(x) e^{-i(2\pi nx/L)} dx \tag{21}$$

Thay thế hàm rời rạc A_n bằng hàm liên tục $F(k)dk$ khi $n/L \rightarrow k$. Thay đổi tổng thành tích phân, và biểu thức trở thành

$$f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi i k x} dk = \mathcal{F}_x[f(x)](k) \tag{22}$$

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx = \mathcal{F}_k^{-1}[F(k)](x) \tag{23}$$

Biểu thức (23) được gọi phép biến đổi Fourier thuận, và biểu thức (22) được gọi là phép biến đổi Fourier nghịch.

4 Methodology

Trong phần này, báo cáo kỹ thuật trình bày tổng quan phương pháp đề xuất cho tác vụ nhúng đồ thị tri thức và tối ưu hóa mô hình. Chúng tôi cũng điểm lại những khái niệm cơ sở của học tăng cường trong ngữ cảnh dữ liệu đa quan hệ như đồ thị tri thức.

4.1 Reinforcement Learning Framework for Multi-hop KG Reasoning

4.1.1 State space

4.1.2 Action space

4.1.3 Transition

4.1.4 Reward

4.1.5 Policy network

4.2 Fourier-Knowledge graph embedding

4.3 Optimization for learning KGE

4.3.1 Adaptive Moment Estimation Method

Adaptive Moment Estimation Method hay Adam cũng thích ứng learning rate đến mỗi tham số. Nó lưu giữ cả mức giảm trung bình cấp số nhân giống như RMSProp và AdaDelta, và cả mức giảm gradient cấp số nhân như momentum. Nói một cách khác, Adam là một phương pháp tận dụng những lợi thế của những bộ tối ưu đi trước.

Khởi tạo gradient và gradient bình phương về 0 dẫn đến sai lệch. Một cài đặt tinh chỉnh bias giúp giảm bớt các vấn đề và theo bài báo gốc, $\alpha = 0,001$, $\gamma_v = 0,9$, $\gamma_s = 0,999$ và $\epsilon = 1 \times 10^{-8}$. Các bước cập nhật cho thuật toán diễn ra như sau trong quá trình huấn luyện mạng học sâu:

1) Cập nhật ước lượng momentum có thiên vị.

$$\mathbf{v}^{(k+1)} = \gamma_v \mathbf{v}^{(k)} + (1 - \gamma_v) \mathbf{g}^{(k)} \quad (24)$$

2) Cập nhật ước lượng gradient có thiên vị.

$$\mathbf{s}^{(k+1)} = \gamma_s \mathbf{s}^{(k)} + (1 - \gamma_s) (\mathbf{g}^{(k)} \odot \mathbf{g}^{(k)}) \quad (25)$$

3) Tính toán hiệu chỉnh ước lượng momentum có thiên vị.

$$\hat{\mathbf{v}}^{(k+1)} = \mathbf{v}^{(k+1)} / (1 - \gamma_v^k) \quad (26)$$

4) Tính toán hiệu chỉnh ước lượng gradient có thiên vị.

$$\hat{\mathbf{s}}^{(k+1)} = \mathbf{s}^{(k+1)} / (1 - \gamma_s^k) \quad (27)$$

5) Cập nhật tham số

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \hat{\mathbf{v}}^{(k+1)} / \left(\epsilon + \sqrt{\hat{\mathbf{s}}^{(k+1)}} \right) \quad (28)$$

4.3.2 Quasi-Hyperbolic Momentum & Adam

QHM (Quasi-Hyperbolic Momentum) là một thuật toán momentum thích ứng khác mà tách momentum ra khỏi gradient hiện tại khi cập nhật trọng số. Nói một cách khác, nó là trung bình có trọng số của momentum và SGD đơn giản, tính theo gradient hiện tại với discount factor tức thời. Công thức cập nhật của Quasi-Hyperbolic Momentum có thể biến đổi trở về để suy biến trở về Nesterov Momentum, Synthesized Nesterov Variants, accSGD và một số trường hợp khác. Luật cập nhật của thuật toán như sau:

$$\mathbf{v}^{(k+1)} = \beta \mathbf{v}^{(k)} + (1 - \beta) \cdot \mathbf{g}^{(k)} \quad (29)$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \left[(1 - \nu) \cdot \mathbf{g}^{(k)} + \nu \cdot \mathbf{v}^{(k+1)} \right] \quad (30)$$

trong đó các hệ số $\nu = 0.7$ và $\beta = 0.999$

4.4 Designing for Action space

4.5 Designing for State space

4.6 Designing for Policy networks

5 Experiments

5.1 Datasets, metrics, and settings

5.1.1 Datasets

Trong bài báo kỹ thuật này, chúng tôi sử dụng bốn tập dữ liệu thí nghiệm được lấy mẫu từ cơ sở dữ liệu Freebase[1], NELL[2], và Wikidata[5] cho phần thực nghiệm và đánh giá. Một cách cụ thể hơn, để thấy được mức độ hiệu quả của mô hình trên những bậc thưa (degree of sparsity) khác nhau, chúng tôi sử dụng hai bộ dữ liệu được xây dựng từ tập FB15k-237[4], bao gồm: FB15k-237-10% và FB15k-237-20%. Những tập dữ liệu này được tạo ra bằng cách lấy ngẫu nhiên lần lượt 10%, và 20% số lượng bộ ba từ tập dữ liệu ban đầu.

Bên cạnh đó, chúng tôi sử dụng thêm hai bộ dữ liệu bao gồm NELL23k và WD-singer được lấy mẫu từ hai cơ sở dữ liệu từ NELL và Wikidata. Với NELL23k, nó được tạo ra bằng cách lấy mẫu ngẫu nhiên một số lượng thực thể từ cơ sở dữ liệu, sau đó dựa trên các thực thể thu được tiếp tục truy vấn ra các bộ ba có chứa chúng để hình thành tập dữ liệu. Với WD-singer, nó được tạo ra từ những đối tượng có liên hệ với ca sĩ từ cơ sở dữ liệu Wikidata. Phương pháp hình thành tập dữ liệu cũng tương tự như NELL23k. Thông tin mô tả thống kê cho các bộ dữ liệu đánh giá được trình bày trong Bảng 5.1.1

Table 1: Thống kê mô tả của bốn tập dữ liệu thực nghiệm.

Dataset	#Entities	#Relation	#Fact	#degree	
				mean	median
FB15K-237-10%	11,512	237	60,985	5.8	4
FB15K-237-20%	13,166	237	91,162	7.5	5
NELL23K	22,925	200	35,358	2.21	1
WD-singer	10,282	135	20,508	2.35	2

5.1.2 Metrics

Với mỗi bộ ba dữ kiện (h, r, t) trong tập dữ liệu, đầu tiên nó sẽ được chuyển đổi thành một bộ ba truy vấn $(h, r, ?)$, và sử dụng mô hình để có được danh sách thứ hạng của những thực thể bị thiếu tương ứng. Để đánh giá khả năng của phương pháp đề xuất, hai độ đo đánh giá được sử dụng: độ đo xếp hạng tương hỗ trung bình (mean reciprocal rank - MRR), và độ đo Hits@K.

Độ đo xếp hạng tương hỗ trung bình (MRR) cho điểm bộ ba dự đoán dựa trên việc chúng có đúng hay không. Nếu bộ ba dự đoán đầu tiên đúng thì điểm của nó là 1 và điểm đúng thứ hai là $\frac{1}{2}$ và cứ tiếp tục tính cho đến bộ ba thứ n, giá trị MRR cuối cùng được tính bằng cách lấy tổng của tất cả các điểm. Độ đo được tính theo công thức:

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{q} \quad (31)$$

Độ đo MRR lấy giá trị nghịch đảo nên khắc phục được độ nhạy cảm với nhiễu của độ đo xếp hạng trung bình (MR) [3]. Giá trị độ đo MRR càng lớn thì kết quả mô hình càng tốt.

Ví dụ: Xem xét các bộ ba kiểm tra bao gồm

Head entity	Relation	Tail entity
Jack	born_in	Italy
Jack	friend_with	Thomas

Giả sử những bộ đúng trong các bộ ba kiểm tra (được đánh dấu *) được xếp hạng với những bộ ba bị phá hỏng (mẫu âm)

Head entity	Relation	Tail entity	Score	Rank
Jack	born_in	Ireland	0.789	1
Jack	born_in	Italy	0.753	2 *
Jack	born_in	Germany	0.695	3
Jack	born_in	China	0.456	4
Jack	born_in	Thomas	0.234	5

Head entity	Relation	Tail entity	Score	Rank
Jack	friend_with	Thomas	0.901	1 *
Jack	friend_with	China	0.345	2
Jack	friend_with	Italy	0.293	3
Jack	friend_with	Ireland	0.201	4
Jack	friend_with	Germany	0.156	5

Điểm số dự đoán của từ bộ ba được tính toán từ mô hình dự đoán liên kết được huấn luyện trước được gán và sắp xếp theo thứ tự giảm dần. Điểm số của bộ ba (Jack, born_in, Italy) được xếp hạng 2 và bộ ba (Jack, friend_with, Thomas) được xếp hạng 1.

$$MRR = \frac{1}{2} \left(1 + \frac{1}{2} \right) = \frac{3}{4} = 0.75$$

Độ đo Hits@K là xác suất dự đoán đúng mà xếp hạng nhỏ hơn hoặc bằng ngưỡng K. Các giá trị của K thường được sử dụng cho bài toán dự đoán liên kết là 1, 3, 5, 7, 10, còn trong thực nghiệm của mô hình đề xuất ACRM sử dụng K = 1, 3, 10. Giá trị độ đo này càng cao thì kết quả của mô hình càng tốt. Công thức của độ đo:

$$Hits@K = \frac{|q \in Q : q < K|}{|Q|} \quad (32)$$

Hits@K đại diện cho khả năng của thuật toán dự đoán chính xác mối quan hệ giữa các bộ ba, nghĩa là đánh giá khả năng của thuật toán hoàn thành biểu đồ tri thức để dự đoán bộ ba chính xác. Đây là một trong những độ đo không thể thiếu đối với bài toán dự đoán liên kết.

Ví dụ: Xem xét các bộ ba kiểm tra bao gồm

Head entity	Relation	Tail entity
Jack	born_in	Italy
Jack	friend_with	Thomas

Giả sử những bộ đúng trong các bộ ba kiểm tra (được đánh dấu *) được xếp hạng với những bộ ba bị phá hỏng (mẫu âm)

Head entity	Relation	Tail entity	Score	Rank
Jack	born_in	Ireland	0.789	1
Jack	born_in	Italy	0.753	2 *
Jack	born_in	Germany	0.695	3
Jack	born_in	China	0.456	4
Jack	born_in	Thomas	0.234	5

Head entity	Relation	Tail entity	Score	Rank
Jack	friend_with	Thomas	0.901	1 *
Jack	friend_with	China	0.345	2
Jack	friend_with	Italy	0.293	3
Jack	friend_with	Ireland	0.201	4
Jack	friend_with	Germany	0.156	5

Điểm số dự đoán của từ bộ ba được tính toán từ mô hình dự đoán liên kết được huấn luyện trước được gán và sắp xếp theo thứ tự giảm dần. Điểm số của bộ ba (Jack, born_in, Italy) được xếp hạng 2 và bộ ba (Jack, friend_with, Thomas) được xếp hạng 1. Điểm số Hits@1 và Hits@3 được tính theo công thức:

$$Hits@1 = \frac{1}{2} = 0.5$$

$$Hits@3 = \frac{1}{1} = 1$$

5.1.3 Settings

Mô hình đề xuất được thực nghiệm trên hệ điều hành Ubuntu 18.04.5 LTS, Intel i7-10700K (16) @ 5.100GHz và GPU NVIDIA GeForce RTX 3070 8 GB. Ngôn ngữ Python 3.7 được sử dụng làm ngôn ngữ lập trình để thực nghiệm vì nó hỗ trợ nhiều thư viện thuận tiện cho việc xây dựng các mô hình học máy cũng như dễ dàng trong việc phân tích đánh giá mô hình. Cụ thể, thuật toán sử dụng PyTorch làm thư viện chính để xây dựng mô hình.

Chúng tôi sử dụng phương pháp tìm kiếm lưới (grid search) để tìm siêu tham số tối ưu cho phương pháp đề xuất. Với module KG, chúng tôi cài đặt số chiều nhúng là 200, và sử dụng Complex-Quasi và Fourier-Quasi như một mô hình được huấn luyện trước cho các module sau. Với policy network module, chúng tôi sử dụng số chiều nhúng cho các bản nhúng thực thể là 512, số chiều nhúng lịch sử là 128, số tầng mạng mô hình hóa lịch sử là 3, tỉ lệ drop cho các bản nhúng là 0.3, tỷ lệ drop cho các tầng feedforward là 0.1, và tỷ lệ drop cho các tầng LSTM là 0.0. Gradient clipping được chọn là 0.0 cho dữ liệu FB15k-237-10%, FB15k-237-20%, và WD-singer, và 5.0 cho NELL23k. Với search strategy module, số lượng cạnh cực đại của các cặp thực thể

là 10, không gian hành động cực đại là 256. Kích thước beam search là 256 cho FB15k-237-10%, FB15k-237-20%, và WD-singer, và 128 cho NELL23k. Với RL module, replay capacity được đặt là 32, decay rate là 0.05. Tỷ lệ drop cho action là 0.5 cho các tập FB15k-237-10%, FB15k-237-20%, và WD-singer, và 0.0 cho NELL23k. Với training module, tần suất bước lấy mẫu là 3, kích thước batch là 128. Tỷ lệ học (learning rate) là 0.001 cho FB15k-237-10%, FB15k-237-20%, và WD-singer, 0.01 cho NELL23k.

5.2 Experimental results

Table 2: Kết quả thực nghiệm trên tập dữ liệu FB15k-237 10%.

	Hits@1	Hits@3	Hits@5	Hits@10	MRR
DacKGR (sample)	—	.239	—	.337	.218
DacKGR (top)	—	.239	—	.335	.219
DacKGR (avg)	—	.232	—	.334	.215
PAAR *	—	—	—	—	—
PAAR ★	.163802	.24	.2787328	.330523	.218495
PAAR-Complex-Quasi	.142424	.166336	.181708	.207163	.164770
PAAR-Fourier-Quasi	.152947	.218072	.254545	.311570	.203906

Table 3: Kết quả thực nghiệm trên tập dữ liệu FB15k-237 20%.

	Hits@1	Hits@3	Hits@5	Hits@10	MRR
DacKGR (sample)	—	.272	—	.391	.247
DacKGR (top)	—	.271	—	.389	.244
DacKGR (avg)	—	.266	—	.388	.242
PAAR *	—	.263	—	.375	.241
PAAR ★	.177184	.264108	.305673	.366707	.239686
PAAR-Complex-Quasi	.157767	.217941	.251922	.300061	.204651
PAAR-Fourier-Quasi	.153216	.213238	.246056	.296875	.200583

Table 4: Kết quả thực nghiệm trên tập dữ liệu NELL23k.

	Hits@1	Hits@3	Hits@5	Hits@10	MRR
DacKGR (sample)	—	.216	—	—	.201
DacKGR (top)	—	.20	—	—	.191
DacKGR (avg)	—	.186	—	—	.171
PAAR *	—	.202	—	.309	.184
PAAR ★	.128635	.216478	.263328	.344709	.196448
PAAR-Complex-Quasi	.118336	.192649	.232835	.295234	.175197
PAAR-Fourier-Quasi	.109249	.187601	.232431	.305735	.171130

Table 5: Kết quả thực nghiệm trên tập dữ liệu WD-Singer.

	Hits@1	Hits@3	Hits@5	Hits@10	MRR
DacKGR (sample)	—	.423	—	.506	.381
DacKGR (top)	—	.405	—	.465	.37
DacKGR (avg)	—	.401	—	.48	.364
PAAR *	—	—	—	—	—
PAAR ★	.292782	.397186	.436223	.486155	.358709
PAAR-Complex-Quasi	.127099	.171584	.191103	.220608	.158798
PAAR-Fourier-Quasi	.224239	.323196	.366319	.408988	.287190

6 Conclusion

Acknowledgment

Đồ án này sẽ không thể hoàn thành được nếu không có sự hỗ trợ của các thầy cô, trợ giảng của môn học. Xin cảm ơn chân thành người yêu tương lai của tôi đã **ân thân** trong suốt thời gian qua để mà tôi có thể tập trung làm việc. Xin cảm ơn bạn Nguyễn Bảo Long đã luôn động viên về mặt tinh thần và hỗ trợ về mặt kỹ thuật.

References

- [1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.
- [2] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1306–1313, 2010.
- [3] Partha Pratim Talukdar et al. Okgit: Open knowledge graph link prediction with implicit types. *arXiv preprint arXiv:2106.12806*, 2021.
- [4] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1499–1509, 2015.
- [5] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.