

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 20, 2023

Tóm tắt

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 (serIALIZED 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 thưa 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.

Các từ khóa Optimization methods · Multi-hop reasoning · Knowledge graph embedding · Fourier transformation · Reinforcement learning

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

Contents

1	Giới thiệu	3
2	Các công trình liên quan	3
2.1	Suy diễn đồ thị tri thức	3
2.2	Suy diễn đồ thị tri thức đa bước	4
2.3	Suy diễn học sâu dựa trên học tăng cường	4
3	Tri thức tiền đề & nền tảng	5
3.1	Đồ thị tri thức	5
3.2	Nhúng đồ thị tri thức	5
3.2.1	Các vectors nhúng	5
3.2.2	Hàm tính điểm	5
3.2.3	Chiến lược phát sinh mẫu âm	5
3.2.4	Hàm mất mát	6
3.3	Suy diễn đồ thị tri thức đa bước	6
4	Phương pháp giải toán	6
4.1	Tổng quan phương pháp	6
4.2	Fourier-Knowledge Graph Embedding	6
4.3	Khung học tăng cường cho suy diễn đa bước	8
4.3.1	Không gian trạng thái (state space)	8
4.3.2	Không gian hành động (action space)	8
4.3.3	Xác suất dịch chuyển trạng thái (transition)	9
4.3.4	Phần thưởng (reward)	9
4.3.5	Mạng chính sách (policy network)	9
4.4	Tối ưu hóa cho khung học tăng cường	10
4.4.1	Phương pháp Adaptive Moment Estimation	10
4.4.2	Phương pháp Quasi-Hyperbolic Momentum & Adam	11
5	Thực nghiệm	11
5.1	Dữ liệu, các độ đo, và cài đặt	11
5.1.1	Các bộ dữ liệu	11
5.1.2	Các độ đo	12
5.1.3	Cài đặt, cấu hình, môi trường thực nghiệm	13
5.2	Các kết quả thực nghiệm dự đoán liên kết	14
5.3	Các kết quả thực nghiệm dự đoán liên kế trên từng loại quan hệ	15
6	Kết luận	15
7	Lời cảm ơn	15

1 Giới thiệu

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 so với phương pháp nhúng trên miền số phức 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 Các công trình liên quan

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 Suy diễn đồ thị tri thức

Những nghiên cứu về suy diễn trong thời gian đầu tập trung vào các yếu tố logic và luật (rules). Các mô hình này học luật xác suất và được chọn lọc bởi con người, và được sử dụng để suy diễn những thể hiện quan hệ hay thể hiện đối tượng mới từ dữ liệu. Tuy nhiên, nhược điểm của chúng rất nhiều, điển hình cả năng mở rộng trên dữ liệu lớn và khả năng tổng quát hóa chưa cao. Hiện nay, nhiều công trình tập trung và phát triển dựa trên phương pháp nhúng (embedding-based methods) mà cho phép suy diễn trên miền không

²<https://github.com/m32us/RL4MRD>

gian liên tục chứ không phải miền không gian rời rạc. Hướng tiếp cận này có nhiều nhóm phương pháp và có thể phân loại chủ yếu thành bốn nhóm chính:

- Translation-based models
- Semantic matching-based models
- Neural network-based models
- Graph neural network-based models

Nhóm phương pháp Translation-based models gồm các mô hình dựa trên các phép biến đổi trên không gian đặc trưng khác nhau mà chủ yếu là phép tịnh tiến trong không gian vector để mô hình hóa tương tác giữa các đối tượng trong dữ liệu đồ thị. Đây là một trong những nhóm phương pháp cổ điển mà vẫn đảm bảo được tính hiệu quả. Một số mô hình như TransE[3], TransH[30], và TransR[14] thể hiện trong không gian Euclidean, còn một số mô hình khác như TorusE[9] sử dụng nền tảng của một nhóm Lie compact, RotatE[21] sử dụng phép xoay trong không gian số phức hoặc ManifoldE[31] sử dụng phép tịnh tiến trong không gian đa tạp. Nhìn chung, nhóm các mô hình này được xây dựng trên các nền tảng toán học vững chắc từ hình học đến phương pháp tô-pô. Hệ quả là, với chiều nhúng thấp, chúng cho phép ta mô hình hóa và đạt được kết quả suy diễn khả quan với hiệu quả tính toán và chi phí thấp. Tuy nhiên, không dễ để có thể chọn lựa một không gian đặc trưng phù hợp cho dữ liệu.

Nhóm phương pháp Semantic matching-based models gồm các mô hình dựa trên việc sử dụng một hàm tương đồng ngữ nghĩa (mà chủ yếu là phép nhân ma trận) để mà tính điểm cho một bộ ba dữ kiện mới. Nhóm này có thể phân loại thành những nhóm mô hình RESCAL và biến thể của nó và nhóm mô hình dựa trên hệ phức. Đối với các mô hình cùng họ với RESCAL như RESCAL[18], DistMult[33], và TuckER[1], chúng sử dụng nền tảng toán học phân rã ma trận. Các mô hình dựa trên hệ phức như ComplEx[25], QuatE[34], và QuatRE[17] sử dụng không gian vector phức, và siêu phức để mô hình và tính toán tương đồng ngữ nghĩa.

Nhóm phương pháp Neural network-based models sử dụng các mô hình mạng Neural thể mô hình hóa những đặc trưng phi tuyến nhằm thể hiện tri thức tiềm ẩn của dữ liệu. Các mô hình trong phương pháp này chủ yếu sử dụng các mô hình, kỹ thuật tính toán như Convolution, Capsule Network, và GAN để xây dựng mô hình nhúng đồ thị. Một số mô hình tiêu biểu có thể kể đến như ConvE[8], ConvR[11], InteractE[26] và KBGAN[4].

Nhóm Graph neural network-based models tổng quát hóa các phương pháp Neural network-based models cho dữ liệu phi Euclidean. Đối với dữ liệu đồ thị tri thức, do tính phức tạp của nó nên hầu hết các mô hình thuộc nhóm này để được xây dựng dựa trên framework encoder-decoder. Một số mô hình tiêu biểu có thể kể đến như R-GCN[19], SACN[20], và CompGCN[27].

Hầu hết các phương pháp nhúng đồ thị đều hiệu quả cho tác vụ suy diễn một bước (có thể hiểu là cận suy diễn). Tuy nhiên, vấn đề với các bài toán suy diễn đa bước, các phương pháp tiếp cận như thế hầu như đã chạm tới giới hạn.

2.2 Suy diễn đồ thị tri thức đa bước

Công trình nổi bật làm tiền đề cho suy diễn đa bước là PRA, mà sử dụng bước đi ngẫu nhiên (random walk) để có được diễn giải hiệu quả cho các đường dẫn suy diễn trong dữ liệu đồ thị. Dựa trên PRA, các phương pháp sau này cải tiến để mô hình tính toán suy diễn hiệu quả hơn và bớt tốn kém chi phí hơn. Các mô hình nổi bật bao gồm: Compositional Reasoning[16], Chains-of-Reasoning[7], cor-PRA[12], và CPR[29].

2.3 Suy diễn học sâu dựa trên học tăng cường

Trong tác vụ suy diễn đa bước, những phương pháp dựa trên học tăng cường xem quá trình suy diễn như một bài toán chuỗi quyết định (serialized decision problem) và sử dụng chiến lược huấn luyện của học tăng cường để quyết nó. Một số mô hình như DeepPath[32] và MINERVA[6] thể hiện những đường đi như một quá trình quyết định Markov (Markov Decision Process) và mã hóa lịch sử các quyết định bởi các mô hình mạng neural bộ nhớ (memory neural architecture) như RNN, LSTM. Nhờ đó, các mô hình dựa trên hướng tiếp cận này có tiềm năng về khả năng bền vững và học các chuỗi suy diễn rất dài. Một số mô hình giải quyết vấn đề mở rộng động của dữ liệu đồ thị tri thức bằng cách kết hợp suy diễn đa bước và rút trích dữ kiện, điển hình là mô hình CPL[10].

Với các phương pháp tối ưu gradient, các chiến lược cho việc chọn lựa luật phù hợp được học mà giảm được độ phức tạp tính toán cũng như khả năng mở rộng của mô hình. Các mô hình như MultiHopKG[13] sử dụng

một mô hình nhúng được huấn luyện trước để ước lượng phần thưởng cho những dữ kiện chưa được nhìn thấy và sử dụng các cạnh mặt nạ được phát sinh ngẫu nhiên để khám phá nhiều hơn những đường đi có thể có. Tuy nhiên, vấn đề thông tin không chắc chắn và sự phân cấp của dữ liệu đồ thị vẫn là những thách thức cho nhiều mô hình hiện tại.

3 Tri thức tiền đề & nền tảng

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 Đồ thị tri thức

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 Nhúng đồ thị tri thức

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 (loss function).

3.2.1 Các vectors nhúng

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 Hàm tính điểm

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 Chiến lược phát sinh mẫu âm

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 Hàm mất mát

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 Suy diễn đồ thị tri thức đa bước

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) $e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} e_3 \cdots \xrightarrow{r_k} e_{k+1}$

4 Phương pháp giải toán

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 Tổng quan phương pháp

4.2 Fourier-Knowledge Graph Embedding

Lấy động lực từ việc mô hình ComplEx không có khả năng mô hình hóa các bộ ba dữ kiện với những quan hệ bắc cầu (transitive relation) do đó mô hình này chưa thật sự tốt trên các bộ dữ liệu thực nghiệm, điều này được chỉ ra trong bài báo của Sun et al.[21]. Trong báo cáo kỹ thuật này, chúng tôi đề xuất sử dụng phép biến đổi Fourier trong không gian số phức, gọi là Fourier-Knowledge Graph Embedding (Fkge) nhằm cải thiện hiệu quả của mô hình pretrain-KGE cho Reinforcement Learning framework để giải quyết bài toán suy diễn đồ thị tri thức đa bước.

Cho trước một bộ ba dữ kiện (h, r, t) , gọi các embedding vectors của chúng lần lượt là $(\mathbf{h}, \mathbf{r}, \mathbf{t})$ trong không gian phức. Hàm tính điểm của Fkge : $\mathbb{C}^{3d} \rightarrow \mathbb{R}$:

$$\text{Fkge}(\mathbf{h}, \mathbf{r}, \mathbf{t}) = \sigma(\text{rfft} \left(\frac{1}{2} (\max(\text{irfft}(\mathbf{h}), \text{Re}(\mathbf{r})) + \min(\text{irfft}(\mathbf{h}), \text{Im}(\mathbf{r}))) \right) \circ \mathbf{t}) \quad (1)$$

trong đó:

- σ : là hàm kích hoạt phi tuyến, ở đây sử dụng hàm sigmoid
- rfft : phép biến đổi Fourier thực thuận
- rfft : phép biến đổi Fourier thực nghịch

Để mà huấn luyện mô hình nhúng đồ thị, chúng tôi sử dụng hàm mất mát lấy mẫu âm với chiến lược self-adversarial training[21] như sau:

$$\mathcal{L} = -\log(\sigma(\lambda - \text{Fkge}(\mathbf{h}, \mathbf{r}, \mathbf{t}))) - \sum_{i=1}^n p(\mathbf{h}', \mathbf{r}', \mathbf{t}') \log(\sigma(\text{Fkge}(\mathbf{h}, \mathbf{r}, \mathbf{t}) - \lambda)) \quad (2)$$

trong đó: λ là siêu tham số lề (margin) cho trước.

Với những mẫu âm, chúng được lấy mẫu dựa trên phân phối được định nghĩa như sau:

$$p(\mathbf{h}'_j, \mathbf{r}, \mathbf{t}'_j \mid \mathbf{h}_k, \mathbf{r}, \mathbf{t}_k) = \frac{\exp(\alpha \text{Fkge}(\mathbf{h}'_j, \mathbf{r}, \mathbf{t}'_j))}{\sum_{k=1} \exp(\alpha \text{Fkge}(\mathbf{h}'_k, \mathbf{r}, \mathbf{t}'_k))} \quad (3)$$

trong đó: α là hệ số nhiệt lượng lấy mẫu (temperature of sampling)

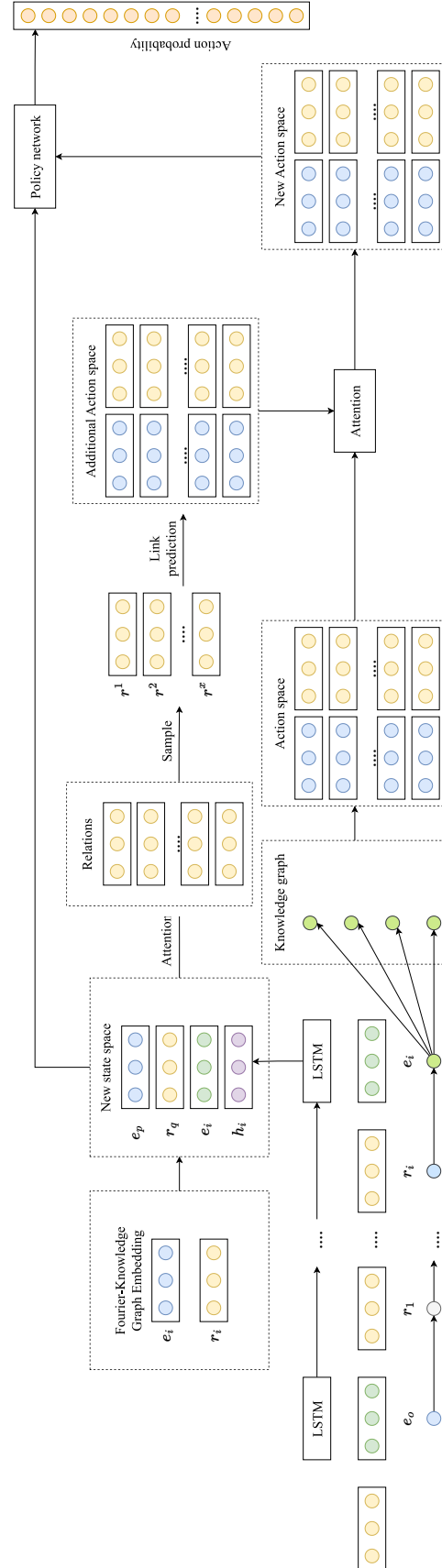


Figure 1: Tổng thể kiến trúc RL framework đề xuất.

4.3 Khung học tăng cường cho suy diễn đa bước

Trong phần này, chúng tôi định nghĩa lại những khái niệm cơ sở của Markov Decision Process (MDP) trong mô hình suy diễn đồ thị tri thức đa bước.

4.3.1 Không gian trạng thái (state space)

Không gian hành động trong suy diễn đồ thị tri thức đa bước có thể được định nghĩa từ quan hệ mục tiêu (target relation), nút hiện tại (current node) và các đường đi lịch sử (historical paths). Trạng thái của bước thứ i có thể được định nghĩa:

$$s_i = (r_q, e_i, h_i) \quad (4)$$

trong đó: r_q , e_i , và h_i lần lượt là các vector đặc trưng của quan hệ, thực thể, và đường đi lịch sử tương ứng.

Trong quá trình suy diễn đa bước, cạnh (quan hệ) mà tác tử lựa chọn không chỉ phụ thuộc vào quan hệ r_q và thực thể hiện tại e_i , mà còn phụ thuộc vào những đường đi ở trong quá khứ. Dựa trên điều này, chúng tôi sử dụng LSTM để mã hóa thông tin h_i .

Dựa trên phát hiện của Das et al., Lin et al., về hiện tượng các mô hình KGE không phụ thuộc vào tính liên thông của đồ thị tri thức. Bằng cách sử dụng mô hình KGE được huấn luyện từ trước, vector xác suất của tất cả thực thể đuôi. Một cách cụ thể, ta gọi vector xác suất $\mathbf{p} \in \mathbf{R}^{|\mathcal{E}|}$, trong đó giá trị của chiều thứ i thể hiện xác suất mà e_i là thực thể đuôi đúng. Chiến lược dynamic anticipation được đề xuất bởi Lv et al.[15] cho phép ta thay đổi công thức thể hiện trạng thái như sau:

$$\mathbf{s}_i = (\mathbf{e}_p, \mathbf{r}_q, \mathbf{e}_i, \mathbf{h}_i) \quad (5)$$

trong đó: \mathbf{e}_p là thông tin dự đoán được từ mô hình KGE.

4.3.2 Không gian hành động (action space)

Với $s_i = (r_q, e_i, h_i)$, nếu tồn tại một bộ ba dữ kiện (e_i, r_n, e_n) trong đồ thị tri thức, ta có thể xem (r_n, e_n) là một trong các hành động, nhiều hành động tạo nên không gian hành động của tác tử. Thế nên, không gian hành động của tác tử trong trạng thái hiện tại được định nghĩa:

$$\mathcal{A}_i = \{(r_n, e_n) \mid (e_i, r_n, e_n) \in \mathcal{T}\} \quad (6)$$

trong đó: \mathcal{T} thể hiện cho tất cả bộ ba dữ kiện của đồ thị tri thức. Để đảm bảo rằng tác tử có thể dừng được trong quá trình suy diễn, các khuyên (r_{loop}, e_i) cũng được thêm vào dữ liệu đồ thị. Đây có thể xem như là một kỹ thuật tăng cường dữ liệu.

Chúng tôi kết hợp phương pháp Dynamic Completion được đề xuất bởi Lv et al.[15], và phương pháp được đề xuất bởi Zhou et al.[35] để tăng cường không gian hành động cho mô hình. Ta gọi tập hợp các hành động bổ trợ (additional actions) $C_i = \{(r, e) \mid (r, e) \in \mathcal{R} \wedge e \in \mathcal{E} \wedge (e_i, r, e) \notin \mathcal{T}\}$. Ta sẽ chọn những hành động với xác suất cao từ C_i , xác suất này có thể được định nghĩa:

$$p((r, e) \mid s_i) = p(r \mid s_t)p(e \mid r, s_i) \quad (7)$$

Để tối ưu hóa thời gian trong việc lập trình, ta lựa chọn một số quan hệ với xác suất cao bằng cách dùng $p(r \mid s_t)$, sau đó chọn những quan hệ với xác suất cao cho một số quan hệ đã chọn này bằng $p(e \mid r, s_t)$

Với trạng thái hiện tại, ta tính được giá trị attention trên tất cả các quan hệ $p(r \mid s_t)$,

$$\mathbf{w} = \text{softmax}(\text{MLP}(\mathbf{s}_i)[\mathbf{r}_1, \dots, \mathbf{r}_{\mathcal{R}}]) \quad (8)$$

Kích thước của không gian hành động của trạng thái hiện tại được kiểm soát bởi tham số α và số lượng hành động cực đại M

$$N_{add} = \min([\alpha N], M) \quad (9)$$

Sau đó, chọn x quan hệ với giá trị attention lớn nhất trong \mathbf{w} để hình thành một tập quan hệ mới $\mathcal{R}_{add} = \{r^1, r^2, \dots, r^x\}$. Với một mô hình KGE được huấn luyện sẵn, không gian hành động bổ trợ được định nghĩa:

$$\mathcal{A}_i^{add} = \{(r^i, e_{r^i}^1), \dots, (r^k, e_{r^k}^1)\} \quad (10)$$

Để hình thành không gian hành động mang ngữ nghĩa tốt hơn, cơ chế attention được tiếp tục sử dụng:

$$\mathcal{A}_i = \text{Attention}(\mathcal{A}_i, \mathcal{A}_i^{add}, [\mathcal{A}_i; \mathcal{A}_i^{add}]) \quad (11)$$

4.3.3 Xác suất dịch chuyển trạng thái (transition)

Nếu trạng thái hiện tại là $s_i = (r_q, e_i, h_i)$, và tác tử chọn hành động tiếp theo là $(r_n, e_n) \in \mathcal{A}_i$, thì trạng thái hiện tại s_i sẽ bị biến đổi thành trạng thái kế tiếp

$$s_{i+1} = (r_q, e_n, h_{i+1}) \quad (12)$$

Trong lập trình, chúng ta sẽ phải giới hạn số bước về T , và xác suất chuyển ở bước cuối cùng sẽ là

$$s_T = (r_q, e_T, h_T) \quad (13)$$

4.3.4 Phần thưởng (reward)

Với một truy vấn cho trước $(e_s, r_q, ?)$, gọi đáp án truy vấn chính xác là e_o . Nếu tác tử dừng tại thực thể chính xác ấy, tức là $e_T = e_o$, thì phần thưởng (reward) lúc này bằng 1. Ngược lại, phần thưởng có giá trị trong khoảng từ 0 đến 1 với hàm tính toán $f(e_s, r_q, e_T)$, trong đó $f(\cdot)$ là một hàm của một mô hình nhúng đồ thị tri thức (knowledge graph embedding - KGE) được huấn luyện trước để đánh giá tính hợp lý của bộ ba (e_s, r_q, e_T)

4.3.5 Mạng chính sách (policy network)

Để có thể hướng dẫn tác tử chọn hành động đúng đắn trong không gian các hành động khác nhau, một mạng chính sách (policy network) là cần thiết. Tổng quan kiến trúc Policy network được thể hiện trong Hình 2

Các đối tượng trong đồ thị tri thức bao gồm các thực thể và quan hệ được thể hiện như những vectors trong không gian ngữ nghĩa, và một hành động (r_i, e_i) tại bước t có thể được biểu diễn như $\mathbf{a}_i = [\mathbf{r}_i, \mathbf{e}_i]$, trong đó \mathbf{r}_i và \mathbf{e}_i là các vector của r_i và e_i tương ứng.

Để mã hóa thông tin đường đi quá khứ, mô hình LSTM (Long-short term memory) được sử dụng. Một cách cụ thể, thể hiện của mỗi hành động được chọn bởi tác tử sẽ được đưa vào LSTM để sinh ra thông tin đường đi quá khứ.

$$\mathbf{h}_i = \text{LSTM}(\mathbf{h}_{i-1}, \mathbf{a}_{i-1}) \quad (14)$$

Biểu diễn trạng thái thứ i : $s_i = (r_q, e_i, h_i)$

$$\mathbf{s}_i = [\mathbf{r}_q, \mathbf{e}_i, \mathbf{h}_i] \quad (15)$$

Hình thành ma trận hành động $\mathbf{A}_t \in \mathbb{R}^{|\mathcal{A}_t| \times 2d}$, trong đó d là chiều của các vector nhúng thực thể và quan hệ.

Mạng policy được định nghĩa:

$$\pi_0(a_t | s_t) = \sigma(\mathbf{A}_t (\mathbf{W}_1 \text{ReLU}(\mathbf{W}_2 \mathbf{s}_t))) \quad (16)$$

trong đó: σ là toán tử softmax, \mathbf{W}_1 và \mathbf{W}_2 là các ma trận biến đổi tuyến tính (linear neural network), và $\pi_0(a_t | s_t)$ là phân phối xác suất trên tất các hành động trong \mathcal{A}_t

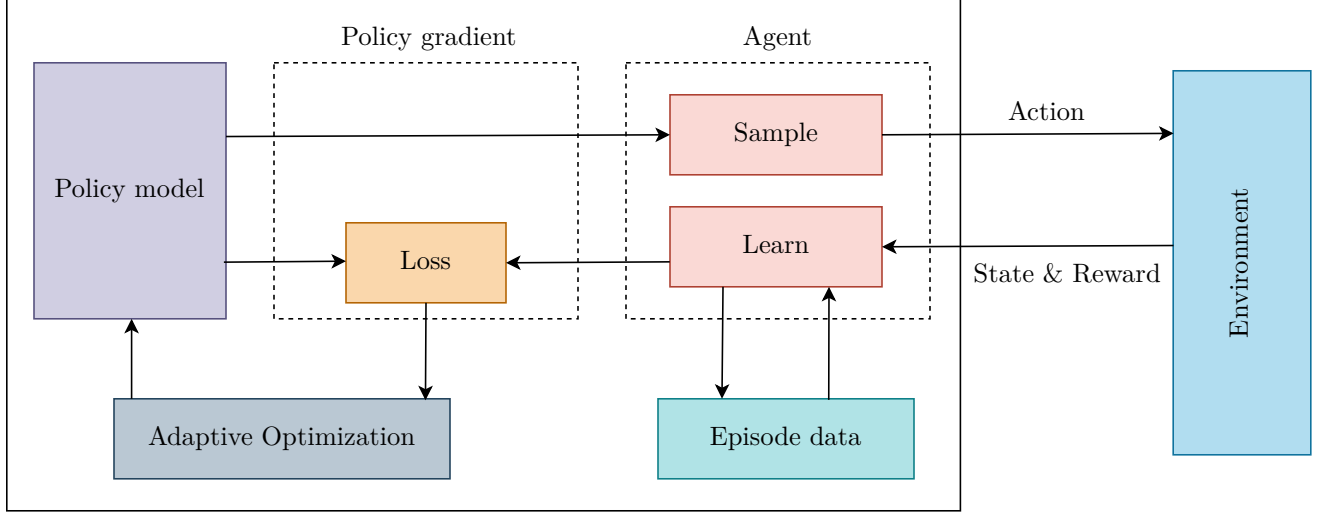


Figure 2: Kiến trúc Policy network.

4.4 Tối ưu hóa cho khung học tăng cường

Trong phần này, chúng tôi trình bày kỹ thuật tối ưu cho RL Learning framework. Một cách cụ thể, chúng tôi sử dụng các phương pháp tối ưu bậc nhất thích nghi dựa trên local model:

- Phương pháp Adaptive Moment Estimation (Adam): Là một phương pháp tối ưu hóa thường được sử dụng trong huấn luyện các mạng học sâu. Ý tưởng đằng sau Adam là tính toán những tỉ lệ học cho từng tham số dựa trên ước lượng bậc nhất và bậc hai moment của gradients.
- Phương pháp Quasi-Hyperbolic Momentum: QHM là một phương pháp cải tiến dựa trên các phương pháp momentum truyền thống. Nó thêm vào một phân số của cập nhật trước đó cho cập nhật hiện tại để giúp tối ưu di chuyển hơn, nhanh hơn theo hướng của gradient. Tác dụng của phân số này giúp cho sự cân bằng của cập nhật trước đó và hiện tại trở nên cân bằng hơn. Điều này được thực hiện bằng việc sử dụng hypergradient như một siêu tham số kiểm soát tính cân bằng.

4.4.1 Phương pháp Adaptive Moment Estimation

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.

$$\mathbf{v}^{(k+1)} = \gamma_v \mathbf{v}^{(k)} + (1 - \gamma_v) \mathbf{g}^{(k)} \quad (17)$$

2) Cập nhật ước lượng gradient có thiên vị.

$$\mathbf{s}^{(k+1)} = \gamma_s \mathbf{s}^{(k)} + (1 - \gamma_s) \left(\mathbf{g}^{(k)} \odot \mathbf{g}^{(k)} \right) \quad (18)$$

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

$$\hat{\mathbf{v}}^{(k+1)} = \mathbf{v}^{(k+1)} / (1 - \gamma_v^k) \quad (19)$$

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

$$\hat{\mathbf{s}}^{(k+1)} = \mathbf{s}^{(k+1)} / (1 - \gamma_s^k) \quad (20)$$

5) Cập nhật tham số

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \hat{\mathbf{v}}^{(k+1)} / \left(\epsilon + \sqrt{\hat{\mathbf{s}}^{(k+1)}} \right) \quad (21)$$

4.4.2 Phương pháp 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:

$$\mathbf{v}^{(k+1)} = \beta \mathbf{v}^{(k)} + (1 - \beta) \cdot \mathbf{g}^{(k)} \quad (22)$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \left[(1 - \nu) \cdot \mathbf{g}^{(k)} + \nu \cdot \mathbf{v}^{(k+1)} \right] \quad (23)$$

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

5 Thực nghiệm

5.1 Dữ liệu, các độ đo, và cài đặt

5.1.1 Các bộ dữ liệu

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[2], NELL[5], và Wikidata[28] 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[24], 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 Các độ đo

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:

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{q} \quad (24)$$

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) [23]. 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:

$$Hits@K = \frac{|q \in Q : q < K|}{|Q|} \quad (25)$$

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 Cài đặt, cấu hình, môi trường thực nghiệm

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, tỷ 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 Các kết quả thực nghiệm dự đoán liên kết

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 *	.164	.240	.278	.331	.218
PAAR-Complex-Quasi	.142	.166	.182	.207	.165
PAAR-Fourier-Quasi	.153	.218	.255	.312	.204

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 *	.177	.264	.306	.367	.240
PAAR-Complex-Quasi	.158	.218	.252	.300	.205
PAAR-Fourier-Quasi	.153	.214	.246	.297	.201

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 *	.129	.216	.263	.345	.196
PAAR-Complex-Quasi	.118	.193	.233	.295	.175
PAAR-Fourier-Quasi	.109	.188	.233	.306	.171

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 *	.293	.397	.436	.486	.359
PAAR-Complex-Quasi	.127	.172	.191	.221	.159
PAAR-Fourier-Quasi	.224	.323	.366	.409	.287

5.3 Các kết quả thực nghiệm dự đoán liên kế trên từng loại quan hệ

Table 6: Thống kê các loại quan hệ trên các tập dữ liệu (test set).

Dataset	to-M relations	to-M examples	to-1 relations	to-1 examples
FB15k-237 10%	55.0/216	6128.0/15624	161.0/216	9496.0/15624
FB15k-237 20%	89.0/223	9386.0/16963	134.0/223	7577.0/16963
NELL23k	68.0/189	3276.0/4961	121.0/189	1685.0/4961

Dataset	Model	Partial graph To-M relations	To-1 relations	Full graph To-M relations	To-1 relations
FB15k-237 10%	PAAR-Complex-Quasi	.0226	.2593	.0234	.2599
	PAAR-Fourier-Quasi	.0225	.250	.2271	.2525
FB15k-237 20%	PAAR-Complex-Quasi	.0402	.4121	.0413	.4142
	PAAR-Fourier-Quasi	.0406	.3982	.0422	.401
Nell23k	PAAR-Complex-Quasi	.1204	.2578	.1228	.2583
	PAAR-Fourier-Quasi	.1194	.2479	.1214	.2485

6 Kết luận

7 Lời cảm ơn

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 **ăn 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] Ivana Balazević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*, 2019.
- [2] 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.
- [3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.
- [4] Liwei Cai and William Yang Wang. Kbgan: Adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071*, 2017.
- [5] 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.

- [6] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*, 2017.
- [7] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*, 2016.
- [8] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [9] Takuma Ebisu and Ryutaro Ichise. Toruse: Knowledge graph embedding on a lie group. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [10] Cong Fu, Tong Chen, Meng Qu, Woojeong Jin, and Xiang Ren. Collaborative policy learning for open knowledge graph reasoning. *arXiv preprint arXiv:1909.00230*, 2019.
- [11] Xiaotian Jiang, Quan Wang, and Bin Wang. Adaptive convolution for multi-relational learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 978–987, 2019.
- [12] Ni Lao, Einat Minkov, and William Cohen. Learning relational features with backward random walks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 666–675, 2015.
- [13] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*, 2018.
- [14] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.
- [15] Xin Lv, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, Wei Zhang, Yichi Zhang, Hao Kong, and Suhui Wu. Dynamic anticipation and completion for multi-hop reasoning over sparse knowledge graph. *arXiv preprint arXiv:2010.01899*, 2020.
- [16] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. Compositional vector space models for knowledge base completion. *arXiv preprint arXiv:1504.06662*, 2015.
- [17] Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, and Dinh Phung. Quatre: Relation-aware quaternions for knowledge graph embeddings. In *Companion Proceedings of the Web Conference 2022*, pages 189–192, 2022.
- [18] Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, et al. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 3104482–3104584, 2011.
- [19] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.
- [20] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3060–3067, 2019.
- [21] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.
- [22] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019.
- [23] Partha Pratim Talukdar et al. Okgit: Open knowledge graph link prediction with implicit types. *arXiv preprint arXiv:2106.12806*, 2021.
- [24] 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.
- [25] Théo Trouillon, Christopher R Dance, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *arXiv preprint arXiv:1702.06879*, 2017.

- [26] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesch Agrawal, and Partha Talukdar. Interact: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3009–3016, 2020.
- [27] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*, 2019.
- [28] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.
- [29] Quan Wang, Jing Liu, Yuanfei Luo, Bin Wang, and Chin-Yew Lin. Knowledge base completion via coupled path ranking. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1308–1318, 2016.
- [30] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.
- [31] Han Xiao, Minlie Huang, and Xiaoyan Zhu. From one point to a manifold: Knowledge graph embedding for precise link prediction. *arXiv preprint arXiv:1512.04792*, 2015.
- [32] Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*, 2017.
- [33] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.
- [34] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. *Advances in neural information processing systems*, 32, 2019.
- [35] Xingchen Zhou, Peng Wang, Qiqing Luo, and Zhe Pan. Multi-hop knowledge graph reasoning based on hyperbolic knowledge graph embedding and reinforcement learning. In *The 10th International Joint Conference on Knowledge Graphs*, pages 1–9, 2021.