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

Phan Minh Tâm - Hoàng Minh Thanh

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

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH HOÀN CHỈNH

Tp. Hồ Chí Minh, tháng MM/YYYY

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

Phan Minh Tâm - 18424059

Hoàng Minh Thanh - 18424062

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

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH HOÀN CHỈNH

GIÁO VIÊN HƯỚNG DẪN

ThS. Lê Ngọc Thành

BM. Khoa Học Máy Tính

Tp. Hồ Chí Minh, tháng MM/YYYY

Lời cam đoan

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

Lời cảm ơn

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

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

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

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

Một lần nữa, xin chân thành cảm ơn !

Mục lục

Danh sách hình

Danh sách bảng

Chương 1

Giới thiệu

Ngày nay đồ thị đã được ứng dụng vào mọi mặt của đời sống, với đồ thị về mạng xã hội (Facebook [17]) 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 [1]), 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 [6]) được Google giới thiệu năm 2012 [7], một loại đồ thị biểu diễn thông tin, là một trong những ứng dụng rõ ràng nhất về đồ thị tri thức cũng như cách dữ liệu được khai thác và biểu diễn trên đồ thị tri thức.

Khai thác đồ thị tri thức hiệu quả cung cấp cho người dùng hiểu sâu hơn về những gì đằng sau dữ liệu và từ đó có thể mang lại lợi ích cho nhiều ứng dụng trong thực tế. Tuy nhiên, trong thực tế, luôn có những tri thức mới được sinh ra mỗi ngày, thông tin thu được thường bị mất mát và không đầy đủ, từ đó nảy sinh ra vấn đề hoàn thiện đồ thị tri thức (knowledge graph completion) hay dự đoán liên kết (linking prediction) trong đồ thị tri thức. Hầu hết các phương tiếp cận hiện nay là dự đoán một cạnh mới nối từ đỉnh này tới đỉnh khác. Với cách tiếp cận như vậy đồ thị tri thức có đầy đủ các tri thức nói cách khác làm cho đồ thị đày đặc nhờ tạo thêm cách cạnh nối. Nhưng như vậy chỉ mới giải quyết được vấn đề hoàn thành

đồ thị, vấn đề thêm một (hoặc một lượng) tri thức mới vào đồ thị vẫn còn là một câu hỏi mở.

Hiện nay các bài toán liên quan đến hoàn thành đồ thị tri thức rất được quan tâm có hai cách tiếp cận chính là tối ưu hóa hàm mục tiêu tức là đưa ra dự đoán đựa trên có sai sót ít nhất như trong RuDiK[13], AMIE[5], RuleN[10] liên quan tới các ứng dụng phân loại đỉnh, phân loại cạnh. Hoặc đưa ra một danh sách gồm k ứng viên với số điểm đại điện cho độ tin cậy giảm đần như trong các nghiên cứu TransE[2], ConvKB[18] liên quan tới các hệ thống gợi ý (recommend system)...Cách tiếp cận của chúng tôi dựa trên cách tạo ra một danh sách gồm k ứng viên này.

Với mỗi cách tiếp cận trên có hai phương pháp chính đưa ra nghiên cứu một cách dựa trên luật như trong AnyBURL[8] hoặc dựa trên nhúng đồ thị như trong ConvE[3], TransE[2], ComplEx [16]. Với mong muốn được tiếp cận các phương pháp có hệ thống nên chúng tôi chọn ở cả hai phương pháp để nghiên cứu thực hiện đề tài này. Đối với phương pháp dựa trên luật chúng tôi chọn phương pháp AnyBURL[8] còn với phương pháp dựa trên nhúng đồ thị GAT[12].

Đóng góp của chúng tôi trong phương pháp AnyBURL[8] gồm mã nguồn Python phương pháp AnyBURL. Cùng với đó chúng tôi cung cấp thêm hai chiến lược để thêm tri thức mới vào đồ thị mà chúng tôi gọi là online-to-offline là mở rộng của AnyBURL trong việc tạo ra các luật khi có một lượng (tập hợp) tri thức mới được thêm vào. Online-to-online sẽ tạo ngay các luật mới khi có một tri thức mới(cạnh) được thêm vào.

Về phương pháp dựa trên nhúng đồ thị chúng tôi sẽ trình bày lại về cơ chế chú ý (attention mechanisms), cách cơ chế chú ý được áp dụng vào đồ thị tri thức bằng mô hình GAT và mô hình KBAT Đóng góp của chúng tôi trong phương pháp học sâu bao gồm mã nguồn trực tuyến trên Google Colab về cách cải tiến của mô hình chúng tôi, cũng như các phân tích của chúng tôi về các siêu tham số trong quá trình huấn luyện để đạt được kết quả tốt hơn.

Chương 2

Các công trình liên quan

Hầu hết các nghiên cứu hiện tại về việc dự đoán liên kết của đồ thị tri thức đều liên quan đến các phương pháp tiếp cận tập trung vào khái niệm nhúng một đồ thị đã cho trong một không gian vectơ có số chiều thấp. Ngược lại với các tiếp cận này là một phương pháp đựa trên luật được nghiên cứu trong [8]. Thuật toán cốt lõi của nó dựa trên lấy mẫu một luật bất kỳ, sau đó khái quát thành các quy tắc Horn[20]. Tiếp đó dùng thống kê để tính độ tin cậy của các luật được khái quát. Khi dự đoán một liên kết mới (cạnh mới) của đồ thị chúng ta dự đoán một đỉnh có cạnh nối với một quan hệ cụ thể (label) với đỉnh còn lại hay không. Cũng đã có rất nhiều phương pháp được nghiên cứu, đề xuất để học các các luật trong đồ thị chẳng hạn như trong RuDiK[13], AMIE[5], RuleN[10]. Như đã nói trong phần trước có hai cách tiếp cận chính cho bài toán này một là tối ưu hóa hàm mục tiêu. Tìm ra một bộ quy tắc nhỏ bao gồm phần lớn các ví dụ là đúng và ít sai sót nhất có thể như được ngiên cứu trong RuDiK[13]. Còn cách tiếp cận còn lại cũng là cách tiếp cận mà chúng tôi chọn nghiên cứu là cố gắng tìm hiểu mọi quy tắc khả thi có thể sau đó tạo xếp hạng k ứng viên tiềm năng với một độ tin cậy nhất định được đo trên tập huấn luyện.

Phương pháp đựa trên luật của chúng tôi phần lớn dựa vào phương pháp Anytime Bottom-Up Rule Learning for Knowledge Graph Completion [9]

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

Trong nhánh các phương pháp về học sâu, rất nhiều kỹ thuật học sâu thành công trong xử lý ảnh và xử lý ngôn ngữ tự nhiên được áp dụng vào đồ thị tri thức như : mạng tích chập, mạng hồi qui, cơ chế chú ý hay mạng bao bọc []

Chương 3

Phương pháp đề xuất

3.1 Phương pháp dựa trên luật

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

3.1.1 Luật Horn

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

Một **literal**[21] là một công thức nguyên tử (atom) hoặc phủ định của nó. Định nghĩa chủ yếu xuất hiện trong lý thuyết logic cổ điển. **Literal** có

thể được chia thành hai loại: Một **positive literal** chỉ là một nguyên tử (ví dụ: x). Một **negative literal** là phủ định của một nguyên tử (ví dụ: $\neg x$). Sự phân chia của **literal** là **positive literal** hay **negative literal** tùy thuộc vào việc **literal** được định nghĩa.

Một mệnh đề (clause) là một literal hoặc nối rời của hai hoặc nhiều literal. Ở dạng **Horn** một mệnh đề có nhiều nhất một positive literal. Lưu ý: Không phải mọi công thức trong logic mệnh đề đều có thể đưa về dạng Horn.Mệnh đề xác định không có literal đôi khi được gọi là mệnh đề đơn vị (unit clause) và một mệnh đề đơn vị không có biến đôi khi được gọi là *facts*[20].Một công thức nguyên tử được gọi là *ground* hoặc *ground atoms* nếu nó được xây dựng hoàn toàn từ các mệnh đề đơn vị; tất cả các *ground atoms* có thể ghép lại từ một tập hợp hàm và các ký hiệu vị từ nhất định tạo nên cơ sở Herbrand cho các bộ ký hiệu này[22].

3.1.2 Định nghĩa đồ thị tri thức

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

Chúng ta định nghĩa một quy tắc là $h(c_0, c_n) \leftarrow b_1(c_0, c_1), \ldots, b_n(c_n, c_{n+1})$ là một đường dẫn ground atoms có chiều dài n. Trong đó $h(\ldots)$ được gọi là *head atoms* và $b_1(c_0, c_1), \ldots, b_n(c_n, c_{n+1})$ được gọi là *body atoms*. Chúng tôi sẽ phân biệt dưới đây ba loại quy tắc mà chúng tôi gọi là: *quy tắc nhị phân* (**B**) là quy tắc trong head atoms chứa 2 biến, quy tắc đơn nguyên kết thúc bằng một đỉnh treo và atom này chỉ chứa biến không chứa hằng

số($\mathbf{U_d}$) và head atoms chỉ chứa 1 biến. Còn quy tắc đơn nguyên kết thúc bằng một atom ($\mathbf{U_c}$) và head atoms cũng chỉ chứa 1 biến.($\mathbf{U_c}$) có thể là một đỉnh treo tới một hằng số bất kì nếu hằng số này trùng mới hằng số trong head atom thì tạo thành một đường đẫn có chu trình.

$$B \qquad h(A_0, A_n) \leftarrow \bigwedge_{i=1}^{n} b_i(A_{i-1}, A_i)$$

$$U_d \qquad h(A_0, c) \leftarrow \bigwedge_{i=1}^{n} b_i(A_{i-1}, A_i)$$

$$U_c \qquad h(A_0, c) \leftarrow \bigwedge_{i=1}^{n-1} b_i(A_{i-1}, A_i) \wedge b_n(A_{n-1}, c')$$

Chúng tôi gọi các quy tắc của các loại này là quy tắc đường đi (path rules), bởi vì các body atoms (phần sau đấu $\leftarrow$) tạo thành một đường đi. Lưu ý rằng nó cũng bao gồm các biến thể quy tắc với các biến được đảo ngược trong các nguyên tử: được đưa ra trong đồ thị tri thức $\mathbb{G}$, đường dẫn có độ dài n là một chuỗi gồm n bộ ba $p_i(c_i, c_i+1)$ với $p_i(c_i, c_i+1) \in \mathbb{G}$ hoặc $p_i(c_i+1, c_i) \in \mathbb{G}$ với $0 \geq i \leq n$. Các mẫu quy tắc trừu tượng (abstract rule patterns) được cho ở trên có độ dài n vì body atoms của chúng có thể được khởi tạo thành một đường dẫn có độ dài n.

Ngoài ra Quy tắc B và quy tắc U_c cũng được gọi là quy tắc kết nối kín. Chúng có thể được học bởi hệ thống khai thác AMIE được mô tả trong [4, 5]. Quy tắc U_d là quy tắc không đóng hay đường đi không tạo thành chu trình vì A_n là biến chỉ xuất hiện một lần. Ví dụ:

$$\begin{aligned}
\mathit{speaks}(X, Y) &\leftarrow \mathit{lives}(X, Y) && (1)\\
\mathit{lives_in_city}(X, Y) &\leftarrow \mathit{lives}(X, A), \mathit{within}(Y, A) && (2)\\
\mathit{gen}(X, \mathit{female}) &\leftarrow \mathit{married}(X, A), \mathit{gen}(A, \mathit{male}) && (3)\\
\mathit{profession}(X, \mathit{actor}) &\leftarrow \mathit{acted_in}(X, A) && (4)
\end{aligned}$$

Quy tắc (1) là quy tắc $\mathbf{B}$(quy tắc nhị phân) quy tắc này nói rằng nếu một

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

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

3.1.3 Thuật toán

Trong phần này chúng tôi mô tả lại thuật toán chính của phương pháp AnyBURL nó cũng được mô tả trong [8] cũng như hai thuật toán mở rộng của chúng tôi để giải quyết vấn đề khi đồ thị được thêm một hoặc một lượng tri thức mới (thêm cạnh). Ngoài ra chúng tôi cũng mô tả sơ lược lại cách khởi tạo một luật cũng như cách thức tính toán độ tin cậy bằng cách lấy mẫu trên tập huấn luyện và vấn đề độ tin cậy khi dự đoán một luật

khi tính toán độ tin cậy bằng việc lấy mẫu.

Thuật toán 1 AnyBURL

Algorithm 1 Anytime Bottom-up Rule Learning

```
1:  procedure AnyBURL(𝔾, s, sat, Q, ts)
2:      n = 2
3:      R = ∅
4:      loop
5:          R_s = ∅
6:          start = currentTime()
7:          repeat
8:              p = samplePath(𝔾, n)
9:              R_p = generateRules(p)
10:             for r ∈ R_p do
11:                 score(r, s)
12:                 if Q(r) then
13:                     R_s = R_s ∪ {r}
14:         until currentTime() > start + ts
15:         R'_s = R_s ∩ R
16:         if | R' | / | R | > SAT then
17:             n = n + 1
18:         R = R_s ∩ R
        return R
```

Đầu vào của thuật toán $\mathbb{G}, S, SAT, Q, TS$. Đầu ra là tập hợp R các luật học được. Trong đó $\mathbb{G}$ là một đồ thị tri thức được cho từ tập dữ liệu đào tạo. S là tham số cho biết kích thước của một lần lấy mẫu trên dữ liệu đào tạo để tính toán độ tin cậy. SAT cho biết độ bão hòa(saturation) của các luật được sinh ra trong 1 lần lặp độ bão hòa này được tính bằng số luật **mới** học được ở lần lặp hiện tại so số sô luật đã học được. Nếu nhỏ hơn độ bão hòa thì chúng tôi cho rằng vẫn còn tiềm năng để khai thác các luật với độ dài n. ngược lại chúng tôi tăng độ dài của luật sau đó tiếp tục khai thác. Q là một ngưỡng để xác định xem luật mới được sinh ra có được thêm vào kết quả trả về hay không. Còn TS cho biết thời gian

học của thuật toán. Chúng tôi bắt đầu với n bằng 2 tức là các luật có độ dài đường đẫn bằng 2 vì trong path rule yêu cầu ít nhất 1 literal trong head atom và 1 trong body atoms. Ở phần lấy mẫu 1 luật(*samplePath*) chỉ đơn giản là chúng ta chọn 1 đỉnh bất kì trong đồ thị duyệt qua tất cả các đường đẫn từ đỉnh đó đi qua n đỉnh khác, sau đó chọn ngẫu nhiên 1 đường đẫn trong số các trường đẫn duyệt được.

Thuật toán 2 tạo 1 luật

Algorithm 2 Generate Rules(p)

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

Ở thuật toán này chúng tôi thay các hằng số vào các head và tail trong toàn bộ path rule của luật được lấy mẫu ở bước trước nếu luật cần học là luật nhị phân ngược lại chúng tôi chỉ thay hoặc head hoặc tail rồi thêm vào luật trả về sau đó chúng tôi lấy mẫu trên tập huấn luyện 1 tập hợp các luật sau đó tính toán độ tin cây như được mô tả trong phần 2.3.2. Để giảm chi phí tính toán chúng tôi chọn cách lấy mẫu trên tập huấn luyện để tính toán. Khi đưa ra dự đoán các ứng cử viên của một luật chúng tôi sẽ tính toán lại bằng cách thêm vào một lượng biểu diễn số luật bị sai mà chúng tôi chưa nhìn thấy trong quá trình lấy mẫu để tính toán độ tin cậy. Dối với mô hình của chúng thôi sau khi thử nghiệm tham số trong khoảng

[5, 10] cho kết quả tốt nhất.

Thuật toán 3 học offline-to-online

Algorithm 3 AnyBURL Learning batch size

procedure ANYBURLBATCH($\mathbb{G}$, S, SAT, Q, TS, BATCH_EDGE)
  $is\_connected = add(\mathbb{G}, batch\_edge)$
  if $is\_connected$ **then**
    $G' = \mathbb{G} \oplus batch\_edge$
  else
    $G' = batch\_edge$
  $n = 2$
  $R = \emptyset$
  loop
    $R_s = \emptyset$
    $start = currentTime()$
    repeat
      $p = samplePath(G', n)$
      $R_p = generateRules(p)$
      for $r \in R_p$ **do**
        $score(r, s)$
        if $Q(r)$ **then**
          $R_s = R_s \cup \{r\}$
    until $currentTime() > start + ts$
    $R'_s = R_s \cap R$
    if $\mid R' \mid / \mid R \mid > SAT$ **then**
      $n = n + 1$
    $R = R_s \cap R$
  return R

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

chúng tôi lấy tất cả các phần trong *batch_edge* sau đó thực hiện lại các bước như thuật toán Anytime Bottom-up Rule Learning.

Thuật toán 4 học online-to-online

Algorithm 4 AnyBURL Learning batch size

```
1: procedure AnyBURLbatch(𝔾, s, sat, Q, ts, edge)
2:     is_connected = add(𝔾, edge)
3:     R = ∅
4:     if is_connected then
5:         n = 2
6:         R_s = ∅
7:         repeat
8:             p = samplePath(edge, n)
9:             R_p = generateRules(p)
10:            for r ∈ R_p do
11:                score(r, s)
12:                if Q(r) then
13:                    R_s = R_s ∪ {r}
14:        until currentTime() > start + ts
15:        R'_s = R_s ∩ R
16:        if | R' | / | R |> SAT then
17:            n = n + 1
18:        R = R_s ∩ R
19:        return R
```

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

Chương 4

Kết quả thí nghiệm

Trong phần này chúng tôi mô tả lại các bộ dữ liệu mà chúng tôi đùng để thực nghiệm đánh giá phương pháp của chúng tôi cùng với so sánh với các kết quả khác của các công trình nổi bật khác được báo cáo trong bảng 2.1. Ngoài ra chúng tôi cũng cố gắng đánh giá hai đề xuất của chúng tôi trong việc thêm một lượng tri thức mới vào đồ thị bằng cách chúng tôi xem tập test là một lượng tri thức mới cần thêm vào và dùng tập vadidation để đánh giá lại phương pháp của chúng tôi. Kết quả chi tiết được báo cáo ở bảng 2.2

4.1 Các tập dữ liệu huấn luyện

Trong thí nghiệm của chúng tôi, chúng tôi thực hiện trên bốn tập dữ liệu phổ biến là FB15k, FB15-237, WN18 và WN18RR.Các bộ dữ liệu này là tập hợp các bộ ba (triple) $\langle head, relation, tail \rangle$ biểu thị cho thực thể đầu có một mối quan hệ với thực thể cuối.

4.1.1 Bộ dữ liệu FB15k

Bộ dữ liệu này được tạo bởi nhóm nghiên cứu A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko [2] bằng cách trích xuất từ

bộ dữ liệu Wikilinks database [1].Wikilinks database thu thập các siêu liên kết(hyperlinks) đến Wikipedia gồm 40 triệu lượt đề cập trên 3 triệu thực thể, họ trích xuất tất cả các dữ kiện liên quan đến một thực thể nhất định có hơn 100 lần được đề cập đến bởi các tài liệu khác cùng với tất cả các dữ kiện liên quan đến thực thể đó (bao gồm cả những thực thể con được nhắc đến trong tài liệu Wikipedia đó), ngoại trừ những thonng tin như: ngày tháng, danh từ riêng, v.v ... Họ cũng chuyển đổi các đỉnh có bậc n được biểu diễn thành các nhóm các cạnh nhị phân tức là liệt kê các cạnh và quan hệ của mọi đỉnh. Tập dữ liệu được chia ngẫu nhiên thành 3 tập: tập training với 1345 relations, 14834 head entities và 14903 tail entities, tập test gồm 916 relations, 11886 head entities, và 11285 tail entities, tập vadiation gồm 961 relations, 12297 head entities, và 11825 tail entities.

4.1.2 Bộ dữ liệu FB15k-237

Bộ dữ liệu này là một tập hợp con của FB15k được xây dựng bởi Toutanova và Chen [15] lấy cảm hứng từ quan sát rằng FB15k bao gồm dữ liệu thử nghiệm được các mô hình nhìn thấy tại thời điểm đào tạo(test leekage). Trong FB15k, vấn đề này là do sự hiện diện của các quan hệ gần giống nhau hoặc nghịch đảo của nhau.FB15k-237 được xây dựng để trở thành một tập dữ liệu thách thức hơn: các tác giả đã chọn các dữ kiện liên quan đến 401 quan hệ xuất hiện nhiều nhất và loại bỏ tất cả các quan hệ tương đương hoặc nghịch đảo. Họ cũng đảm bảo rằng không có thực thể nào được kết nối trong tập huấn luyện cũng được liên kết trực tiếp trong tập test và validation.Tập training gồm 237 relations, 13781 head entities, và 13379 tail entities, tập test gồm 223 relations, 7652 head entities, và 5804 tail entities, tập vadition gồm 224 relations, 8171 head entities, and 6376 tail entities.

[1]https://code.google.com/archive/p/wiki-links/

4.1.3 Bộ dữ liệu WN18

Bộ dữ liệu này được giới thiệu bởi các tác giả của TransE [2], được trích xuất từ WordNet[2], một bản thể học ngôn ngữ KG có nghĩa là cung cấp một từ điển/từ đồng nghĩa để hỗ trợ NLP và phân tích văn bản tự động. Trong WordNet, các thực thể tương ứng với các tập hợp (*word senses*) và các quan hệ đại diện cho các kết nối từ vựng của chúng (ví dụ: "hypernym"). Để xây dựng WN18, các tác giả đã sử dụng WordNet làm điểm bắt đầu và sau đó lặp đi lặp lại lọc ra các thực thể và mối quan hệ với quá ít lần được đề cập. Tập dữ liệu được chia ngẫu nhiên thành 3 tập: tập training với 18 relations, 40504 head entities, và 40551 tail entities, tập test gồm 18 relations, 4262 head entities, and 4338 tail entities, tập vadiation gồm 18 relations, 4349 head entities, and 4263 tail entities.

4.1.4 Bộ dữ liệu WN18RR

Bộ dữ liệu này là một tập hợp con của WN18 được xây dựng bởi DeŠmers et al.[3], cũng là người giải quyết vấn đề rò rỉ thử nghiệm (test leakage)trong WN18. Để giải quyết vấn đề đó, họ xây dựng tập dữ liệu WN18RR thách thức hơn nhiều bằng cách áp dụng một phương pháp tương tự đươợc sử dụng cho FB15k-237 [15]. Training gồm 11 relations, 39610 head entities, và 31881 tail entities, tập test gồm 11 relations, 2958 head entities, và 2619 tail entities, tập vadition gồm 11 relations, 2851 head entities, and 2575 tail entities.

4.2 Kết quả thực nghiệm

Trong phần này chúng tôi mô tả lại các phương pháp đánh giá(độ do), môi trường thực hiện cũng như các tập dữ liệu mà chúng tôi sử dụng để dánh giá phương pháp của mình. Các phương pháp đánh giá(độ do) này

[2]https://wordnet.princeton.edu/

cũng phổ biến nó được đánh giá cho hầu hết các mô hình dự đoán liên kết trên đồ thị. Chúng tôi tiến hành so sánh với bốn phương pháp nổi bật khác được báo cáo trong [14].

4.2.1 Các độ đo

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

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

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

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

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

Hit@K (H@K). Đó là tỷ lệ các dự đoán đúng mà rank nhỏ hơn hoặc

bằng ngưỡng K:

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

4.2.2 Kết quả

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

Như mô tả ở phần thuật toán AnyBURL thuật toán này sẽ học các luật được sinh ra trong một khoảng thời gian nhất định do người dùng cấu hình. Ở đây chúng tôi chọn cấu hình thời gian là 1000 giây tương đương khoảng 17 phút đào tạo, với độ bão hòa(SAT) 0.85, độ tin cậy Q 0.05, kích thước mẫu S ($\frac{1}{10}$ tập huấn luyện). Với cấu hình như vậy mô hình phiên bản Python của chúng tối cho kết quả tương đương với phiên bản Java nhóm tác giả Meilicke, Christian et al. [8] với cấu hình tương tự nhưng thời gian training là 100 giây. Sự khác biệt về thời gian học tập ở đây chủ yếu là do hiệu năng của hai ngôn ngữ Python và Java. Ở đây chúng tôi chọn ngôn ngữ Python vì nó được dùng làm ngôn ngữ chính cho nhiều mô hình trí tuệ nhân tạo gần đây, và cũng thuận tiện cho chúng tôi khi so sánh hiệu năng cũng như đánh giá với các phương pháp học sâu khác da số được viết bằng Python.

Bảng 4.1, và bảng 4.2 bên dưới mô tả các kết quả thực nghiệm của chúng tôi với các độ đo $H@K$ cùng với các kết quả thực nghiệm của các phương pháp khác được đề cập trong khảo sát [14]

Bảng 4.3 bên dưới mô tả các kết quả thực nghiệm của chúng tôi với hai

Bảng 4.1: Kết quả phương pháp AnyBURL trên FB15k, FB15k-237

	FB15k-237				**FB15k-237**			
	H@1	**H@10**	**MR**	**MRR**	**H@1**	**H@10**	**MR**	**MRR**
ComplEx	81.56	90.53	34	0.848	25.72	52.97	202	0.349
TuckER	72.89	88.88	39	0.788	25.90	53.61	162	0.352
TransE	49.36	84.73	45	0.628	21.72	49.65	209	0.31
RoteE	73.93	88.10	42	0.791	23.83	53.06	178	0.336
ConvKb	59.46	84.94	51	0.688	21.90	47.62	281	0.305
GAT								
BURL	**79.13**	**82.30**	**285**	**0.824**	**20.85**	**42.40**	**490**	**0.311**

chiến lược thêm tri thức mới vào đồ thị. Chúng tôi đánh giá trên tổng số luật được sinh ra, và số luật có độ tin cậy $>= 50\%$ và $>= 80\%$.

Bảng 4.2: Kết quả phương pháp AnyBURL trên WN18, WN18RR

	WN18				**WN18RR**			
	H@1	**H@10**	**MR**	**MRR**	**H@1**	**H@10**	**MR**	**MRR**
ComplEx	94.53	95.50	3623	0.349	42.55	52.12	4909	0.458
TuckER	94.64	95.80	510	0.951	42.95	51.40	6239	0.459
TransE	40.56	94.87	279	0.646	2.79	94.87	279	0.646
RoteE	94.30	96.02	274	0.949	42.60	57.35	3318	0.475
ConvKb	93.89	95.68	413	0.945	38.99	50.75	4944	0.427
GAT								
BURL	**93.96**	**95.07**	**230**	**0.955**	**44.22**	**54.40**	**490**	**0.490**

Bảng 4.3: Kết quả hai chiến lược thêm tri thức mới

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

Chương 5

Kết luận

Trong phần này chúng tôi đưa ra các kết quả đạt được của mô hình chúng tôi, chúng tôi cố gắng tìm hiểu các đặc trưng của các bộ dữ liệu tương ứng để cố gắng lý giải thích tại sao mô hình của chúng tôi hoặc các công trình khác có được kết quả tốt trên tập dữ liệu tương ứng đó. Những kết quả của hai đề xuất của chúng tôi cũng như các dịnh hướng nghiên cứu của chúng tôi trong tương lai.

Mặc dù kết quả chúng tôi cho thấy phương pháp của chúng tôi có hiệu suất tương đương với các mô hình học sâu hiện đại(state-of-art) và có ưu thế vượt trội trong thời gian đào tạo khoảng 17 phút so với thời gian hàng giờ của phương pháp học sâu khác nhưng không phải là các mô hình deep learning này không đáng nghiên cứu. Chúng tôi cũng nhận thấy đối với những tập dữ liệu khó như FB15-237 hay WN18RR phương pháp của chúng tôi thường cho kết quả không tốt do các quan hệ tương tự hay nghịch đảo không không xuất hiện trong ví dụ đào tạo nên chúng tôi khó tạo ra các luật đủ tốt để có thể khái quát hóa trên toàn bộ đồ thị dẫn đến các kết quả không tốt. Ngược lại đối với các phương pháp dựa trên học sâu lại có ưu thế rất lớn trong các tập dữ liệu này do có thể dễ đàng tính toán độ gần của các luật mới cần đánh giá so với các luật đã học từ đó có một kết quả khá tốt. Do đó chúng tôi cũng sẽ tiếp tục nghiên cứu các phương pháp học sâu và sẽ dùng phương pháp này làm đường cơ sở(base

line) để so sánh với các nghiên cứu của chúng tôi trong tương lai. Một điểm yếu nữa của mô hình đựa trên luật của chúng tôi là mặc dù thời gian học là vượt trội nhưng thời gian để tính toán đưa ra đự đoán khá lâu do phải duyệt qua tất cả các luật được sinh ra mới có thể đưa ra dự đoán. Không giống như các phương pháp nhúng đồ thị khác thao tác này có thể dễ dàng tính toán.

Đối với hai thuật toán mở rộng của chúng tôi trong việc thêm tri thức mới vào đồ thị chúng tôi nhận thấy rằng là vượt trội hoàn toàn so với các phương pháp học sâu. Ở các phương pháp học sâu điều này đường như chưa được ai chú trọng nghiên cứu mặc dù thời gian đào tạo một mô hình là tương đối mất thời gian. Khi có tri thức mới hầu hết các mô hình phải đào tạo lại toàn bộ điều này khá lãng phí. Chúng tôi cũng xem đây là mục tiêu tiếp theo cho chúng tôi khi nghiên cứu các mô hình học sâu trong tương lai. Gần đây nhánh học tăng cường(Reinforcement learning) khá phát triển và nhóm tác giả Meilicke, Christian and Chekol [11] gần đây cũng đã có 1 nghiên cứu để tối ưu hóa lại phương pháp AnyBURL này. Chúng tôi cũng có ý định nghiên cứu về hướng này và cố gắng báo cáo lại trong một tương lai gần.

Danh mục công trình của tác giả

1. Tạp chí ABC
2. Tạp chí XYZ

Tài liệu tham khảo

Tiếng Anh

[1] Baluja, Shumeet et al. "Video suggestion and discovery for youtube: taking random walks through the view graph". In: *Proceedings of the 17th international conference on World Wide Web*. 2008, pp. 895–904.

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

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

[4] Galárraga, Luis et al. "AMIE: association rule mining under incomplete evidence in ontological knowledge bases". In: *WWW '13*. 2013.

[5] Galárraga, Luis et al. "Fast rule mining in ontological knowledge bases with AMIE". In: *The VLDB Journal* 24.6 (2015), pp. 707–730.

[6] 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`.

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

[8] Meilicke, Christian et al. *Anytime Bottom-Up Rule Learning for Knowledge Graph Completion*. 2019. URL: http://web.informatik.uni-mannheim.de/AnyBURL/meilicke19anyburl.pdf.

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

[10] Meilicke, Christian et al. "Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion". In: *International Semantic Web Conference*. Springer. 2018, pp. 3–20.

[11] Meilicke, Christian et al. "Reinforced Anytime Bottom Up Rule Learning for Knowledge Graph Completion". In: *arXiv preprint arXiv:2004.04412* (2020).

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

[13] Ortona, Stefano, Meduri, Venkata Vamsikrishna, and Papotti, Paolo. "Robust discovery of positive and negative rules in knowledge bases". In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE. 2018, pp. 1168–1179.

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

[15] Toutanova, Kristina and Chen, Danqi. "Observed versus latent features for knowledge base and text inference". In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. 2015, pp. 57–66.

[16] Trouillon, Théo et al. "Complex embeddings for simple link prediction". In: International Conference on Machine Learning (ICML). 2016.

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

[18] 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.

[19] Wikipedia contributors. *Atomic formula — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Atomic_formula&oldid=899265834`. [Online; accessed 31-July-2020]. 2019.

[20] Wikipedia contributors. *Horn clause — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Horn_clause&oldid=931306598`. [Online; accessed 31-July-2020]. 2019.

[21] Wikipedia contributors. *Literal (mathematical logic) — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Literal_(mathematical_logic)&oldid=947684114`. [Online; accessed 31-July-2020]. 2020.

[22] Wikipedia contributors. *Term (logic) — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Term_(logic)&oldid=969831286`. [Online; accessed 31-July-2020]. 2020.

Phụ lục A

Ngữ pháp tiếng Việt

Đây là phụ lục.

Phụ lục B

Ngữ pháp tiếng Nôm

Đây là phụ lục 2.