Mục Lục

[CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI 3](#_Toc286197324)

[**1.1.** **Giới Thiệu** 3](#_Toc286197325)

[**1.2.** **Mục tiêu và phạm vi đề tài** 5](#_Toc286197326)

[**1.3.** **Các ứng dụng và nghiên cứu hiện nay** 8](#_Toc286197327)

[**1.3.1.** **Các ứng dụng liên quan** 8](#_Toc286197328)

[**1.3.2.** **Các nghiên cứu liên quan** 11](#_Toc286197329)

[**1.4.** **Cơ sở lý thuyết và các thuật ngữ liên quan** 17](#_Toc286197330)

[**1.4.1.** **Giới thiệu về hỏi đáp** 17](#_Toc286197331)

[**1.4.2.** **Thực thể đặt tên** 17](#_Toc286197332)

[**1.4.3.** **Gán nhãn từ loại** 18](#_Toc286197333)

[**1.4.4.** **Bộ ba quan hệ và đồ thị khái niệm** 18](#_Toc286197334)

[**1.4.5.** **Wordnet và Similarity Wordnet** 20](#_Toc286197335)

[CHƯƠNG 2: ĐỀ XUẤT PHƯƠNG PHÁP XỬ LÝ CÂU HỎI NGƯỜI DÙNG ĐỂ TÌM KIẾM BÀI BÁO 21](#_Toc286197336)

[**2.1** **Tạo tập tin cấu hình ngữ nghĩa cho cơ sở dữ liệu** 21](#_Toc286197337)

[**2.2** **Các bước xử lý câu hỏi người dùng** 25](#_Toc286197338)

[**2.2.1** **Mô hình các bước xử lý** 25](#_Toc286197339)

[**2.2.2** **Tiền xử lý** 26](#_Toc286197340)

[**2.2.3** **Liệt kê các bộ ba quan hệ về từ** 31](#_Toc286197341)

[**2.2.4** **Nhận diện thực thể** 34](#_Toc286197342)

[**2.2.5** **Sinh câu truy vấn SQL** 37](#_Toc286197343)

[**2.2.6** **Một số ví dụ** 43](#_Toc286197344)

[CHƯƠNG 3: XÂY DỰNG HỆ THỐNG HỎI ĐÁP ĐỂ TÌM KIẾM BÀI BÁO KHOA HỌC 47](#_Toc286197345)

[**3.1** **Tổng quan về hệ thống** 47](#_Toc286197346)

[**3.1.1** **Mô hình hệ thống:** 47](#_Toc286197347)

[**3.1.2** **Các chức năng trong hệ thống** 48](#_Toc286197348)

[**3.2** **Thiết kế cài đặt hệ thống** 50](#_Toc286197349)

[**3.2.1** **Môi trường xây dựng hệ thống** 50](#_Toc286197350)

[**3.2.2** **Cấu trúc các lớp** 50](#_Toc286197351)

[**3.2.3** **Cơ sở dữ liệu DBLP** 51](#_Toc286197352)

[**3.2.4** **Giao diện chương trình** 54](#_Toc286197353)

[**3.2.5** **Đánh giá kết quả** 55](#_Toc286197354)

[**CHƯƠNG 4: KẾT LUẬN** 56](#_Toc286197355)

[THAM KHẢO 57](#_Toc286197356)

[Tiếng Việt 57](#_Toc286197357)

[Tiếng Anh 57](#_Toc286197358)

[Online 57](#_Toc286197359)

# CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

* 1. **Giới Thiệu**

Hiện nay, cùng với sự phát triển mạnh mẽ của công nghệ thông tin nói chung và Internet nói riêng đã và đang phục vụ rất nhiều lợi ích cho con người. Song song với sự phát triển đó là lượng thông tin ngày càng lớn dần và kéo theo nó là nhu cầu tìm kiếm thông tin trên Internet ngày càng gia tăng. Để đáp ứng các nhu cầu ấy, hàng loạt các công cụ tìm kiếm đã được ra đời như MSN, Yahoo, Google ,… Tuy nhiên, các công cụ ấy hầu hết dựa trên từ khóa hay cụm từ khóa và không đưa ra câu cả lời cụ thể. Đây là cách tìm kiếm quen thuộc đối với người dùng hiện nay.

Với việc tìm kiếm từ khóa như vậy, các công cụ chỉ đưa về các trang liên kết hoặc các văn bản liên quan có chứa từ khóa cần tìm kiếm. Người dùng đôi khi cần phải tối ưu hóa câu truy vấn để tìm được thông tin chính xác nhất và phải chi trả thời gian cho việc tham khảo từng trang hoặc tài liệu để tìm được câu trả lời phù hợp. Điều đó vẫn chưa làm người dùng thực sự hài lòng.

Ngược lại, một hệ thống hỏi đáp sẽ phục vụ tốt hơn trong vấn đề tìm kiếm thông tin. Nó có thể tìm kiếm chính xác câu trả lời bằng việc đưa vào hệ thống một câu hỏi dưới dạng ngôn ngữ tự nhiên. Kết quả trả về thỏa mãn được yêu cầu người dùng. Thời gian hao tốn cho việc tìm kiếm thông tin sẽ giảm bớt. Mục đích cuối cùng của người dùng là mong muốn tìm kiếm được thông tin một cách chính xác và đầy đủ. Hơn thế nữa, điều này có thể làm máy tính “thông minh” hơn, tạo một hướng giao tiếp thân thiện giữa người và máy.

Với nhận định như trên, đề tài hướng tới xây dựng một hệ thống tìm kiếm thông tin dựa theo hướng hỏi đáp bằng ngôn ngữ tự nhiên. Dữ liệu mà nhóm hướng tới là các bài báo khoa học thuộc lĩnh vực công nghệ thông tin. Hệ thống sẽ phục vụ cho việc tìm kiếm các bài báo.

DBLP là một nguồn thích hợp để thực hiện đề tài này. Digital Bibliography & Library Project (DBLP) cung cấp thông tin chỉ mục các bài báo trong lĩnh vực khoa học máy tính, hệ thống này được phát triển bởi trường đại học Universität Trier của Đức. Tính đến tháng 1/2011 DBLP chứa thông tin chỉ mục của 1, 5 triệu bài báo trong lĩnh vực khoa học máy tính được thu thập từ các tạp chí và hội nghị. DBLP được xây dựng lên từ các file danh sách các đề mục – mục lục (tables of contents– TOCs), các file TOCs được tác giả nhận từ các tạp trí, hội nghị sau đó qua quá trình xử lý bằng các bộ phân tích hệ thống sẽ đánh chỉ mục thông tin các bài báo, từ đó lưu xuống và hình thành lên cơ sở dữ liệu chỉ mục DBLP. Dữ liệu của DBLP là dữ liệu mở, dữ liệu được xuất ra các định dạng CDF, XML và MySQL, người phát triển có thể tải các file dữ liệu từ trên trang chủ của DBLP.

Trên thực tế, việc tìm kiếm các bài báo bằng từ khóa cũng chiếm không ít thời gian của những người nghiên cứu và tìm kiếm. Nếu các công cụ tìm kiếm có hỗ trợ chức năng tìm kiếm nâng cao, thì ta phải cung cấp nhiều thông tin đầu vào để có một kết quả tốt hơn. Mà người dùng đôi khi nắm đầy đủ hết thông tin của một bài báo. Một hệ thống hỏi đáp có thể sẽ phục vụ tốt hơn trong vấn đề tìm kiếm các bài báo khoa học. Nó sẽ tiết kiệm thời gian trong vấn đề nghiên cứu. Việc nhập vào nhiều vùng dữ liệu hay tối ưu câu truy vấn để có một kết quả chính xác nay không cần thiết. Thay vào đó là một câu hỏi duy nhất bằng ngôn ngữ tự nhiên. Kết quả trả về sẽ phù hợp theo những gì mà câu hỏi người dùng đưa vào.

Vậy, với ý kiến như trên, nhóm muốn đưa ra một cách thức tìm kiếm bài báo khác với hướng truyền thống là từ khóa. Chính vì thế, nhóm quyết định chọn đề tài:

**“Xây dựng hệ thống tìm kiếm bài báo khoa học dựa trên hỏi đáp bằng ngôn ngữ tự nhiên”**

Tóm lại, nhóm đề xuất cách thức hỏi đáp trong việc tìm kiếm tài liệu bài nhằm thỏa mãn được nhu cầu tìm kiếm của người dùng. Người dùng có một lựa chọn khác để tìm kiếm bài báo. Đó là nhập vào hệ thống một câu hỏi dưới dạng ngôn ngữ tự nhiên thay cho từ khóa. Kết quả trả về của hệ thống là các thông tin và bài báo tương ứng với nội dung của câu hỏi. Kết quả đánh giá sẽ được đánh giá dựa trên thuật toán xử lý câu hỏi người dùng : khả năng rút trích thông tin từ câu hỏi, khả năng tạo câu truy vấn.

* 1. **Mục tiêu và phạm vi đề tài**

Hệ thống được xây dựng mục đích chỉ để phục vụ tìm kiếm các bài báo khoa học chỉ thuộc lĩnh vực công nghệ thông tin. Vậy đối tượng người dùng là các nhà nghiên cứu, các kỹ sư, sinh viên công nghệ thông tin, ... Cách thức tìm kiếm sẽ bao gồm cả việc tìm kiếm bằng từ khóa và câu hỏi dưới dạng ngôn ngữ tự nhiên. Câu hỏi người dùng nhập vào sẽ dưới dạng ngôn ngữ tiếng Anh.

Nguồn dữ liệu chứa thông tin về các bài báo sẽ được lấy từ nguồn DBLP (Digital Bibliography & Library Project). Đây là một kho thư viện điện tử lưu trữ thông tin của hơn 1,5 triệu bài báo [12]. Đây là một kho dữ liệu lớn và thích hợp để nhóm thực hiện đề tài. Ngòai ra, kho dữ liệu này còn được cung cấp miễn phí và tải về tại trang chủ của DBLP[[1]](#footnote-2). Do đó, phạm vi của thông tin chỉ nằm trong kho dữ liệu của DBLP.

Dựa vào nguồn dữ liệu, các câu hỏi của người dùng nhóm giới hạn chỉ bao xoay quanh các vấn đề liên quan đến thông tin của các bài báo khoa học : tiêu đề, tên tác giả, năm công bố và nhà xuất bản.

Tuy nhiên, số lượng câu hỏi mà hệ thống có thế giải quyết giới hạn bởi hai điều sau. Thứ nhất, nó phải có nội dung nằm trong các thông tin của bài báo. Thứ hai, các câu hỏi chỉ thuộc về một số loại mà nhóm đã phân ra sẵn.

Trong tiếng Anh, câu hỏi cơ bản được chia làm ba loại chính: Yes/No Question, Question Word Question và Choice Question. Đây là các lọai câu hỏi thông dụng mà ta thường thấy. Ngòai ra, ta còn có thể thấy một số lọai câu hỏi khác như tag question, open question, closed question, direct/ indirect question [13-16]. Tuy nhiên, trong phạm vi đề tài, ta chỉ xét tới 3 dạng chính nêu trên, vì tùy theo mục đích sử dụng mà còn các loại câu hỏi khác nhau thêm nữa.

Theo TREC[[2]](#footnote-3), nơi chứa một tập các câu hỏi liên quan đến vấn đề hỏi đáp, các câu hỏi lại được phân làm 3 lọai : FACTOID, LIST và OTHER. Câu hỏi thuộc lọai FACTOID là một loại câu hỏi dành cho những vấn đề chưa được xác minh về thông tin và không chắc là có thông tin để trả lời. Câu hỏi LIST là câu hỏi thuộc dạng liệt kê một danh sách theo yêu cầu của câu hỏi và còn lại là các câu hỏi OTHER.

Đặc biệt trong các hệ thống hỏi đáp, đa số các nghiên cứu thì không phân loại theo như trên. Theo một số nghiên cứu [5,6] về việc phân lọai câu hỏi mà nhóm biết được, thì câu hỏi được phân lọai theo 6 mục thô và 50 mục mịn. Các mục này được dưa ra bởi Li và Roth (2002) [4]. Đây là hình thức phân lọai mà được sử dụng riêng cho vấn đề hỏi đáp.

|  |  |
| --- | --- |
| **Mục thô** | **Mục mịn** |
| ABBR | abbreviation,expansion |
| DESC | definition, description, manner, reason |
| ENTY | animal, body, color, creation, currency, disease, event, food, instrument, language, letter, other, plant, product, religion, sport, subtance, symbol, technique, term, vehicle, word |
| HUM | description, group, individual, title |
| LOC | city, country, mountain, other, state |
| NUM | Code, count, date, distance, money, order, other, percent, period, speed, temperature, size, weight |

Bảng 1 – Phân loại 6 mục thô và 50 mục mịn theo Li và Roth

Vậy, tùy theo mục đích sử dụng mà câu hỏi còn đựơc phân lọai theo nhiều mục.

* Đối với 3 lọai câu hỏi cơ bản trong tiếng Anh, thì hệ thống của nhóm hiện tại không giải quyết các câu hỏi choice question. Đó là câu hỏi thuộc dạng cho người dùng lựa chọn đáp án đã có sẵn. Chúng không phù hợp cho vấn đề tìm kiếm. Hệ thống chỉ giải quyết được một số các câu hỏi thuộc loại Question Word Question và Yes/No question. Đối với câu hỏi Yes/No question hệ thống xử lý tương tự câu hỏi Question Word Question. Vì câu hỏi loại này chỉ nhằm xác minh một vấn đề, cho nên, nếu hệ thống kiếm được kết quả thì sẽ trả về và ngược lại.
* Đối với phân loại của TREC thì các phân loại như FACTOID, LIST và OTHER vẫn chưa được rõ ràng. Nếu phân loại theo 3 mục này thì sẽ khó xác định được mục đích của câu hỏi muốn hỏi tới vấn đề gì.
* Đối với các câu hỏi được phân loại theo 6 mục thô (ABBR, DESC, ENTY, HUM, LOC, NUM). Đây là cách phân loại hợp lý nhất đây là cách phân loại mà các hệ thống hỏi đáp thường dùng. Tuy nhiên, số lượng 6 các mục thô và 50 mục mịn thì không phù hợp để phân loại cho các câu hỏi thuộc đề tài, vì thực tế các câu hỏi trong vệc tìm kiếm bài báo không cần thiết phải phân loại chi tiết đến 50 mục mịn.

Vì số lượng câu hỏi giải quyết được trong đề tài chỉ thuộc về một phạm vi xung quanh các bài báo công nghệ thông tin, nên nhóm chỉ chọn một số dạng câu hỏi để xử lý tùy dựa trên:

* Dựa trên nguồn dữ liệu của DBLP ( các thuộc tính trong các bảng)
* Nhu cầu người dùng : các câu hỏi liên quan đến các vấn đề mà người dùng cần biết.
* Một số các mẫu câu trong tập TREC 2003 và 2007

Bảng phân loại các câu hỏi mà hệ thống xử lý:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Loại** | **Xử lý được** | **Không xử lý** | **Ví dụ** | **Chú thích** |
| Yes/No Question | X |  | *Are they good jobs?*  *Did he know just how deeply his father was opposed to things?* | Xử lý tương tự question-word question |
| Question -word question | X |  | *What books were published in Acm in 2010?’* | Chỉ xử lý những câu hỏi liên quan đến thông tin bài báo khoa học |
| Choice question |  | X | *Do you want a strong cup or a weak one?* | Không có giá trị trong tìm kiếm |
| Tag question | X |  | The weather's bad, isn't it? | Xử lý tương tự question-word question |
| Open/closed question |  | X | *Do you know what i mean?*  *How does it work?* | Không xử lý |
| Direct/Indirect question |  | X | Could you tell me where he is? (where is he? | Không xử lý |
| List question | X |  | Find somes books published in 2010 |  |

Bảng 2 – Phân loại các câu xử lý/ không xử lý.

Vậy, hệ thống chỉ giải quyết được các câu hỏi liên quan đến sách, bài báo, tác giả, nhà xuất bản hay các câu hỏi về số lượng. Những dạng câu hỏi bao gồm: Question-Word question (tức câu hỏi có các tư hỏi như What, When, Where, Why...), Yes/No question và tag question. Các loại câu hỏi khác hiện nhóm chưa thể xử lý được do các lý do: về giải pháp, ngữ nghĩa và mục đích sử dụng. (Các loại câu hỏi chưa được đề cập trong báo này mặc định là chưa xử lý được ).

Từ phạm vi các câu hỏi, nhóm xây dựng một tập kiểm thử gồm 100 câu hỏi để phục vụ cho kiểm tra và đánh giá thuật toán. Tập này được tập hợp từ ý kiến của các thành viên và tham khảo một số mẫu câu thông dụng từ nguồn đáng tin cậy là TREC.

* 1. **Các ứng dụng và nghiên cứu hiện nay**
     1. **Các ứng dụng liên quan**

Hiện nay đã có một số trang web phục vụ riêng cho vấn đề tìm kiếm các bài báo khoa học như CiteXeer, IEEE, ACM, ScienceDirect, SSRN, PaperCube … Các trang web này đều có hệ thống tìm kiếm bài báo dựa trên từ khóa. Ngoài ra, người dùng còn có thể chọn lựa các tùy chọn để có kết quả tốt hơn (tìm kiếm theo tên tác giả, tìm kiếm theo tiêu đề, tìm kiếm theo nội dung abstract …). Một số trang như PaperCube có khả năng hiển thị bài báo theo dạng mô hình đồ thị. Điều này giúp người dùng dễ dàng định hướng được trong việc tìm kiếm.

Sau khi khảo sát một số trang web trên, nhóm tóm tắt lại ở bảng sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên trang** | **Hình thức**  **tìm kiếm** | **Cách thức tìm kiếm** | **Hiển thị** |
| CiteXeer | Từ khóa | Theo tiêu đề, theo tên tác giả, tìm các bảng (table) có trong nội dung bài báo. | Danh sách kết quả và nội dung chi tiết |
| IEEE | Từ khóa | Theo tiêu đề | Danh sách kết quả và nội dung chi tiết |
| ACM | Từ khóa | Theo tiêu đề | Danh sách kết quả và nội dung chi tiết |
| Sciencedirect | Từ khóa | Tất cả các trường, tên tác giả, tiêu đề | Danh sách kết quả và nội dung chi tiết và outline của bài báo |
| SSRN | Từ khóa | Tiêu đề, tên tác giả, nội dung tóm tắt (abstract) , từ khóa , ngày. | Danh sách kết quả và nội dung chi tiết |
| PaperCube | Từ khóa | Tiêu đề, nội dung tóm tắt, năm, chủ đề, tên tác giả | Danh sách kết quả, nội dung chi tiết bài báo, hiển thị bài báo dưới dạng đồ thị, cây … |

Bảng 3 – Khảo sát các trang tìm kiếm bài báo

Như bảng đánh giá trên thì hầu hết các trang đều hỗ trợ việc tìm kiếm dựa trên tiêu đề và tên tác giả. Các kết quả từ việc tìm kiếm đều trả một danh sách các bài báo có liên quan đến từ khóa. Vì thế một số trang phải hỗ trợ thêm chức năng tìm kiếm nâng cao (advance search) để nâng cao kết quả tìm kiếm dựa trên việc nhập thêm nhiều thông tin tìm kiếm và ràng buộc. Đối với các chức năng này thì buộc người dùng phải nhập thêm một số thông tin để có được đáp án chắc chắn. Cho nên, như đã trình bày, việc tìm kiếm bài báo vẫn còn hạn chế về thời gian tìm kiếm.

Đó là những ý kiến về các hệ thống tìm kiếm bài báo hiện nay. Thêm vào đó, nhóm tham khảo và tìm hiểu về một số hệ thống hỏi đáp. Hiện theo như khảo sát của nhóm, đã có một số hệ thống hỏi đáp như Answer.com, NSIR, START, Answerbus.com … Các hệ thống này đã và đang được sử dụng trên internet. Tuy nhiên, các hệ thống hỏi đáp này phục vụ cho việc tìm kiếm thông tin như địa danh, nơi chốn, các vấn đề khoa học, lịch sử, địa lý …

* Question Answering System – NSIR [21]. Được phát triển bởi nhóm nghiên cứu CLAIR tại đại học Michigan. Hệ thống trả lời câu hỏi trong hầu hết các lĩnh vực, từ đơn giản đến phức tạp. Bên cạnh đó, hệ thống cũng lưu trữ một ngân hàng câu hỏi cho người sử dụng chọn lựa.
* [The START Natural Language Question Answering System](http://start.csail.mit.edu/) [22]. Là hệ thống hỏi và trả lời dựa trên nền Web đầu tiên trên thế giới. Không giống như các hệ thống rút trích thông tin khác, mục đích của START là hỗ trợ cho người sử dụng tìm kiếm được thông tin chính xác thay cho việc chỉ đơn thuần cung cấp một danh sách các kết quả tìm kiếm được. Hiện tại, hệ thống có thể trả lời hàng triệu câu hỏi bằng tiếng Anh về địa điểm, phim ảnh, con người, khái niệm…
* Ask.com [23] Là hệ thống hỗ trợ tìm kiếm các trang Web, hình ảnh, thông tin, bản đồ, lời chỉ dẫn, tìm kiếm cục bộ và mua sắm. Ask.com đang có những cải tiến đáng kể như thu hẹp phạm vi tìm kiếm để nâng cao chất lượng kết quả tìm được chứ không tìm kiếm trên tất cả các lĩnh vực. Nó cung cấp khả năng tìm kiếm, trả lời các câu hỏi tập trung vào các vấn đề cơ bản như các công thức món ăn, sở thích, bài tập cho trẻ em, giải trí và sức khỏe.

Tóm lại, khảo sát cho thấy có nhiều ứng dụng tìm kiếm và hỏi đáp đã và đang chạy trên thực thế. Việc sử dụng hỏi đáp để phục vụ tìm kiếm thông tin cũng như bài báo là một hướng khả thi. Tuy nhiên, riêng với một hệ thống hỏi đáp để tìm kiếm các bài báo khoa học, nhóm chưa tìm thấy một hệ thống nào có hỗ trợ chức năng này. Có thể tạm kết luận là do người dùng vẫn còn quen thuộc với việc tìm kiếm bằng từ khóa hoặc các hệ thống còn đang trong thời gian nghiên cứu.

* + 1. **Các nghiên cứu liên quan**

Với những hệ thống hỏi đáp nêu trên cho thấy rằng có những nghiên cứu về vấn đề xử lý câu hỏi tự nhiên đề tìm kiếm thông tin. Đã có nhiều bài báo trình bày về hệ thống hỏi đáp bằng ngôn ngữ tự nhiên [1,2,7,8,9].

Vấn đề chung của các nghiên cứu là hiểu được câu hỏi mà người dùng đưa vào. Câu hỏi bằng nhiều cách sẽ được chuyển thành dạng ngôn ngữ mà máy có thể hiểu được. Ví dụ như câu truy vấn SQL.

* Một nghiên cứu gần đây của Wael Salloum, 2009 [7]. Tác giả đưa ra hướng tiếp cận bằng cách mô hình các văn bản và câu hỏi người dùng thành một dạng đồ thị gọi là Conceptual Graph Formalism (CGF). Đồ thị khái niệm là một hình thức biểu diễn tri thức được John F. Sowa đưa ra năm 1976.

Nguồn dữ liệu của đề tài này là từ các văn bản (document) nói chung không thuộc một lĩnh vực nào cả. Một văn bản được tách thành nhiều câu. Và mỗi câu dựa trên việc phân tích cú pháp, mỗi nút trong cây cú pháp sẽ được rút khái niệm từ wordnet kết hợp với công cụ verbnet để xác định mối quan hệ giữa động từ với chủ từ, đối tượng. Câu sẽ được chuyển thành các Đồ thị khái niệm (gọi là Concept Graph (CG)). Cuối cùng các đồ thị này sẽ được lưu xuống cơ sở dữ liệu. Tương tự cho câu hỏi cũng chuyển thành một CG.

Cuối cùng, tác giả có đề cập tới một phép chiếu (projection operator) dùng để so sánh giữa CG của một câu hỏi với CG của một câu trong văn bản. Đó là cách mà câu trả lời được rút ra, xếp hạng và trả về cho người dùng.

Ví dụ : với câu “Mark Twain wrote Tom Sawyer”, thì đồ thị khái niệm sẽ được tạo như sau:



Hình 1 – Ví dụ về một đồ thị khái niệm

( được lấy từ Figure 3. tài liệu tham khảo [2])

Trong đồ thị ở hình 1, theo cách hiểu của nhóm, *Author* là một lớp đối tượng và một thể hiện của nó là *Mark Twain*. Cũng như thế, *Novel* có thể hiện là Tom Sawyer. Giữa *Author* và *Novel,* ta có mối quan hệ *Write*. *Agent* và *Patient* được coi là các mối quan hệ trung gian: *Author* là tác nhân (agent) của hành động *write* và *Novel* là một thể bị động (patient) so với hành động *write*.

Tương tự câu hỏi cũng được chuyển thành đồ thị khái niệm. Vấn đề ở các câu hỏi là cần xác định đối tượng cần hỏi.

Ví dụ: *“ Who write Tom Sawyer? “* sẽ đuợc chuyển thành như sau:

*[write]-*

*(Agnt) -> [Person: \*] ?*

*(Ptnt) -> [Novel:Tom Sawyer]*

*Hay trình bày theo cách hiểu:*

*[Person: ?] -(Agnt)-[write]-(Ptnt)- [Novel:Tom Sawyer]*

Theo cách ghi ở đây, là một cách khác để thể hiện, *Person* là đối tượng cần biết trong câu hỏi , giữa *person* và *Tom Sawyer* có mối quan hệ *write*. *Agnt* và *Ptnt* là hai từ viết tắt của *Agent, Patient*. Đây là một cách thể hiện khác của đồ thị so với hình 1. Sau đó, công việc tiếp theo là so sánh ( projection operator) giữa đồ thị CG của câu hỏi với các đồ thị khái niệm CG mỗi câu để tìm câu trả lời.

Kết quả bài báo trên đạt được là biểu diễn văn bản và câu hỏi dưới dạng các CG. Bằng cách so sánh giữa các CG, tìm ra câu trả lời. Hướng đi tiếp theo của tác giả là tìm cách tổng hợp lại các câu trả lời với mục đích là đưa ra một câu trả lời duy nhất.

* Với một nghiên cứu trong nước, nhóm tìm hiểu được một đề tài “Xây dựng công cụ tìm kiếm tài liệu học tập bằng các truy vấn ngôn ngữ tự nhiên trên kho học liệu mở tiếng Việt” [2]. Với tên đề tài thì hệ thống hỏi đáp mà tác giả hướng tới là một hệ thống hỏi và trả lời trên ngôn ngữ tiếng việt.

Với cách tiếp cận của đề tài trên, câu hỏi tiếng việt được đưa vào hệ thống phân tích. Một tập 40 luật cú pháp tiếng việt được tác giả tạo ra sẵn. Nếu phân tích được, thành phần sẽ tạo ra một cây cú pháp tương ứng. Từ đó, cây cú pháp được chuyển thành cây sinh mã truy vấn bằng cách ánh xạ lên một số nút của cây cú pháp vào ontology. Cuối cùng, câu truy vấn SPARQL được tạo ra và lấy dữ liệu trả lời.

Ontology có thể xem là một kho dữ liệu được lưu trữ dưới dạng ngữ nghĩa. SPARQL là ngôn ngữ để truy vấn vào kho dữ liệu này [17,18].

Dữ liệu trong đề tài này được lấy từ kho học liệu mở vocw[[3]](#footnote-4). Các metadata được rút từ các trang của vocw và cập nhật dữ liệu vào ontology.

Kết quả của đề tài trên được đánh giá như sau: với 40 câu hỏi chuẩn thì hệ thống chính xác 100%, với 91 câu hỏi ngẫu nhiên thì độ chính xác là 84,62%. Hướng đi tiếp theo của đề tài là cải thiện bộ phân tích cú pháp với mục đích tạo ra bộ phân tích mạnh hơn và nhanh hơn. Đồng thời, việc mở rộng không gian tìm kiếm cũng được tác giả lưu ý tới. Vậy, đề tài bị hạn chế bởi 40 luật cú pháp, do việc xử lý ngôn ngữ Tiếng Việt còn gặp nhiều khó khăn.

* Một đề tài khác là “ Dịch câu truy vấn có cấu trúc sang đồ thị ý niệm: cách tiếp cận ít phụ thuộc vào cú pháp”, 2008 [3]. Theo ý kiến của tác giả, việc phân tích cú pháp để hình thành đồ thị ý niệm gặp khó khăn khi các câu truy vấn không đúng cú pháp. Vì thế tác giả đưa ra một phương pháp dịch câu truy vấn sang đồ thị ý niệm mà không dựa trên việc phân tích cú pháp câu.

Ở đề tài này, phương pháp tiếp cận dựa trên việc nhận diện các thực thể đặt tên, không đặt tên và từ quan hệ. Sau khi nhận diện được, thao tác tiếp theo là nhận diện lớp của thực thể mà không xác định được nó thuộc lớp nào ( như What, Who ...). Bước kế tiếp là gom các thực thể được cho là giống nhau, xác định những mối quan hệ ẩn mà không được nêu trong câu hỏi. Cuối cùng, xác định quan hệ giữa các thực thể, xóa bỏ những mối quan hệ không thích hợp và xây dựng đồ thị ý niệm.

Công cụ đựơc sử dụng trong đề tài này là GATE [20]. Do GATE có khả năng làm việc với Ontology, công cụ OCAT (Ontology-based Corpus Annotation tool) được sử dụng để chú thích các thực thể đặt tên. Ngòai ra, công cụ ANNIE được dùng để nhận diện các từ quan hệ và thực thể không đặt tên bằng các lập ra các danh sách trong Gazetteer. Ontology được sử dụng ở đây có tên là PROTON.

Tập dữ liệu câu hỏi kiểm nghiệm được lấy từ 440 câu hỏi của TREC 2002 và 445 câu hỏi của TREC 2007.

Việc đánh giá của đề tài dựa trên độ chính xác của việc tạo đồ thị ý niệm. Kết quả của đề tài trên nhóm xin trình bày theo bảng sau:

(Quá trình làm giàu Ontology trong nghiên cứu này chủ yếu thủ công bằng cách bổ sung các lớp và mối quan hệ)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tổng Số câu | Trước khi làm giàu  Ontology | Sau khi làm giàu  Ontology |
| TREC 2002 | 440 | 15.23% | 78.64% |
| TREC 2007 | 445 | 8.54% | 60.45% |

Bảng 4- Kết quả nghiên cứu tạo đồ thị ý niệm từ tài liệu [3]

* Với hệ thống eLSSNL (eLibrary Searching System by Natural Language) của tài liệu [1], câu truy vấn dưới dạng ngôn ngữ tự nhiên không được xây dựng thành đồ thị ý niệm như tài liệu [3] và [7].

Đối với hệ thống eLSSNL, tác giả dùng một phương pháp khác so với việc xây dựng đồ thị ý niệm. Câu truy vấn tự nhiên của người dùng được nhận dạng các giá trị thuộc tính tương ứng với các thuộc tính của bảng trong cơ sở dữ liệu. Do hệ thống này sử dụng dữ liệu miễn phí từ dự án Gutenberg[[4]](#footnote-5) chứa thông tin của khoảng 33,000 ebooks miễn phí.

Cách để hiểu câu truy vấn dưới dạng ngôn ngữ tự nhiên của người dùng được tài liệu [1] giải quyết bằng việc xây dựng một tập từ khóa dựa trên các chỉ định từ mà tác giả đã liệt kê. Sau đó, các giá trị hay từ khóa đó được nhận diện thuộc về thuộc tính nào trong bảng cơ sở dữ liệu.

Ví dụ:

*Nhận dạng giá trị cho thuộc tính Title/FriendlyTitle:*

+  DOM(Title) hoặc  DOM(FriendlyTitle) (DOM(fileld): là miền giá trị của một thuộc tính (field) trong table BOOKS )

+ Đi kèm theo các chỉ định từ: title/heading

*Nhận dạng giá trị cho thuộc tính NumDownload:*

+ Là số *(isNumber)*, ưu tiên tìm theo ID trước

+ Đi kèm theo các chỉ định từ: [more] than/>=/>; less/<=/<; top; best; most; numdownload.

Đối với từng thuộc tính của bảng sẽ có những điều kiện để nhận dạng như ví dụ trên. Cuối cùng bằng các phép kết hợp, câu truy vấn được sinh ra và tìm các dữ liệu thỏa mãn.

Ngoài ra, hệ thống eLSSNL còn hỗ trợ lưu lại các các câu truy vấn mà người dùng nhập vào và lưu trong một bảng dưới cơ sở dữ liệu. Người dùng còn được hỗ trợ gợi ý cho người dùng các câu truy vấn tương tự. Việc đó dựa vào việc so sánh các tập từ khóa được lưu lại trong bảng vừa đề cập và so sánh với tập từ khóa trong câu hỏi người dùng vừa nhập vào.

Theo đánh giá về hiệu quả tìm kiếm, hạn chế lớn nhất của eLSSNL là về mặt tốc độ. Nếu dùng người nhập vào một câu truy vấn tự nhiên mới hoàn toàn – tức là tập từ khóa không trùng với các tập từ khóa được lưu trữ- thì tốc độ tìm kiếm sẽ khá lâu nếu người dùng tìm trên nhiều giá trị thuộc tính.

**Nhận xét:**

* Đối với các phương pháp tạo đồ thị ý niệm từ câu hỏi [3,7], thì việc tạo đồ thị dựa trên nhận diện các thực thể đặt tên, không đặt tên. Các công cụ để nhận diện có thể dùng là Wordnet hay một Ontology (PROTON). Với cách phân tích cú pháp để tạo đồ thị ý niệm của [7] thì có gặp khó khăn nếu người dùng nhập một câu có nhập nhằng về cú pháp (hoặc sai cú pháp). Vì thế nhóm thấy hướng tiếp cận không theo việc phân tích cú pháp của [3] sẽ có thể khắc phục được điều này.
* Tương tự với nghiên cứu của tài liệu [2], theo hướng hỏi đáp trên ngôn ngữ tiếng việt, việc phân tích cú pháp của dựa theo 40 luật của tác giả. Nếu câu không có cú pháp đúng theo 40 luật này thì quá trình phân tích cú pháp thất bại.
* Hệ thống eLSSNL có hướng đi gần giống với nhóm, do hệ thống phục vụ tìm kiếm các sách báo trong kho dữ liệu Gutenberg bằng câu truy vấn tự nhiên. Tuy nhiên, việc xác định các thuộc tính bằng các chỉ định từ chỉ cho ra các kết quả là các cuốn sách. Nếu người dùng muốn biết các thông tin thêm về tác giả, các cuốn sách mà tác giả đó đã viết thì hệ thống eLSSNL không hỗ trợ.
* Ngòai ra, để nhận diện được thực thể đặt tên, là các từ (cụm từ) chỉ về tên người, địa danh , tổ chức...[19] thì ta cần có một từ điển như Wordnet để nhận diện. Hoặc với một cách khác, ta có thể dùng một ontology như theo tài liệu [3]. Đối với đề tài của nhóm, thì các thực thể đặt tên trong dữ liệu DBLP là các title, publisher, source, ... có số lượng rất lớn. Do dữ liệu của DBLP chứa hơn 1,5 triệu bài báo. Vì thế, việc nhận diện thực thể đặt tên (như tên tiêu đề bài báo) theo các cách trên có thể sẽ ảnh hưởng nhiều tới thời gian xử lý và nhận diện.
  1. **Cơ sở lý thuyết và các thuật ngữ liên quan**
     1. **Giới thiệu về hỏi đáp**

QA hay hỏi đáp là một loại rút trích thông tin, cung cấp một tập các tài liệu (ví dụ như các trang Web hay tài liệu cục bộ). Hệ thống có thể đưa ra câu trả lời cho những câu hỏi đặt ra bằng ngôn ngữ tự nhiên. Bởi vì phải xử lý ngôn ngữ tự nhiên (NLP) nên QA phức tạp hơn các loại rút trích thông tin khác (như rút trích văn bản chẳng hạn) và đôi khi nó được xem như là bước kế tiếp của bộ máy tìm kiếm.

QA nghiên cứu giải quyết một phạm vi rộng lớn các loại câu hỏi khác nhau, bao gồm: sự việc, danh sách, sự xác định, như thế nào, tại sao, giả thuyết, ràng buộc ngữ nghĩa và các câu hỏi đa ngôn ngữ. QA tìm kiếm trên phạm vi rộng lớn các tập hợp khác nhau, từ tập hợp văn bản cục bộ đến nội dung trên các trang Web. Bao gồm hai phân hệ chính:

* **Closed-domain QA:** giải quyết những câu hỏi với những lĩnh vực xác định. Công việc này tương đối dễ dàng bởi vì hệ thống NLP có thể khai thác kiến thức lĩnh vực chuyên ngành khá chuẩn xác.
* **Open-domain QA:** giải quyết những câu hỏi về mọi lĩnh vực và có thể chỉ cung cấp kiến thức tổng quát chung chung. Những hệ thống như thế này thường có nhiều dữ liệu sẵn có để rút trích câu trả lời.
  + 1. **Thực thể đặt tên**

Thực thể đặt tên là những thành phần của văn bảng thuộc vào những mục sau: tên người, tên tổ chức, địa điểm, thời gian, số lượng, phần trăm … Trong câu hỏi người dùng hay trong văn bản, các thực thể đặt tên cũng thường hay xuất hiện.

Ví dụ trong câu *“What books were written by Philip K. Chan”* thì *Philip K. Chan* được xem là một thực thể đặt tên (tên tác giả).

Do thực thể đặt tên không theo một nguyên tắc và giới hạn nào. Để trong vấn đề nhận biết thực thể đặt tên, ta thường áp dụng một số từ điển như Wordnet. Tuy nhiên, số lượng thực thể đặt tên trong Wordnet cũng có giới hạn.

GATE là một công cụ hỗ trợ mạnh mẽ trong việc xử lý văn bản. GATE hỗ trợ nhận diện các thực thể đặt tên như tên người, nơi chốn, năm .... Do GATE dựa trên các luật và một tập từ điển lưu trữ trong một danh sách gọi là Gazetteer.

* + 1. **Gán nhãn từ loại**

Trong phạm trù ngơn ngữ học, gán nhãn từ loại (part-of-speech tagging) còn được gọi là gán nhãn ngữ pháp hay định hướng từ loại. Đó là quá trình đánh dấu lên những từ trong một văn bản thuộc về một phần của bài phát biểu (tài liệu). Quá trình gán nhãn dựa trên định nghĩa của từ, cũng như ngữ cảnh (Ví dụ: mối quan hệ của nó với các từ lân cận hoặc liên quan trong cùng một câu, cụm từ, hoặc một đoạn. Một hình thức đơn giản của việc này là việc xác định các từ như danh từ, động từ, tính từ, trạng từ ... (theo nguồn wikipedia)

Việc sử dụng gán nhãn từ loại sẽ giúp ta biết được vai trò của mỗi từ trong câu (danh từ, động từ hay tính từ ... ). Từ đó, sẽ hỗ trợ ta cho xử lý về sau như việc phân tích cú pháp, hoặc loại bỏ hay nhận biết các từ loại cần thiết trong câu, so sánh ngữ nghĩa giữa hai từ có cùng từ loại ...

Một số công cụ gán nhãn từ loại thường được biết đến: Wordnet, Gate pos tagger, OpenNLP ...

* + 1. **Bộ ba quan hệ và đồ thị khái niệm**

Trong các web ngữ nghĩa hiện nay thường sử dụng mô hình lưu trữ RDF ([Resource Description Framework](http://www.w3.org/RDF/)) [24]. RDF là một mô hình lưu trữ các đối tượng (resource) và các mối quan hệ giữa chúng. Đây là một mô hình ngữ nghĩa đơn giản và được biểu diễn trong cú pháp XML.

Theo [24] và [2] thì RDF là một mô hình lưu trữ theo dạng các bộ ba subject-predicate- object. Trong đó Subject là một chủ từ bắt đầu cho một bộ ba. Predicate là các vị từ ở giữa biểu diễn cho mối quan hệ giữa Subject và Object. Object là đối tượng còn là mà subject có quan hệ tới.

Việc hình thành các bộ ba như trên sẽ có lợi cho việc lưu trữ ngữ nghĩa. Các bộ ba có thể kết hợp lại thành một đồ thị ngữ nghĩa (Hình 2). Đồ thị khái niệm (hay ngữ nghĩa ) như trong hình là sự kết hợp giữa các bộ ba quan hệ. Đây là một cách để biểu diễn tri thức dưới dạng đồ thị được John F. Sowa đưa ra năm 1976 .



Hình 2 – Thông tin được biểu diễn dưới dạng đồ thị liên kết bởi các bộ ba

(Hình lấy từ tài liệu tham khảo [24])

Với hình trên, các bộ ba được liên kết thành một đồ thị. Một độ thị như thế sẽ có thể biểu diễn cho một đoạn văn bản, một thông tin trên web, một câu nói ... Ta có thể dễ dàng thấy rằng “*the\_thirteenth\_floor*” là một “*movie*”. Do đó nó có mối quan hệ “*is\_a*”. Đồng thời nó cũng có mối quan hệ để thể hiện ngày phát hành là “*released\_in*” và có giá trị là năm “*1999*”. Vậy tóm lại, với cách hiểu như thế thì câu nói có thể là: *The Thirteenth Floor is a movie which are realesed in 1999*.

Dựa vào các bộ ba quan hệ này mà đã có một số nghiên cứu đã áp dụng để phục vụ cho vấn đề hỏi đáp như [2], [7], [8], [9] và [10]. Chủ yếu các nghiên cứu trình bày cách xây dựng các bộ ba từ câu hỏi và sử dụng chúng để truy vấn xuống một cơ sở dữ liệu đã lưu trữ sẵn các bộ ba như cơ sở dữ liệu MSQL/MYSQL hay Ontology. Ontology một dạng mô hình có thể lưu trữ dữ liệu theo dạng RDF.

* + 1. **Wordnet và Similarity Wordnet**

WordNet: là một dạng từ điển ngữ nghĩa tiếng Anh, được tạo ra từ năm 1985 và đang được duy trì ở các phòng thí nghiệm khoa học nhận thức của Đại học Princeton dưới sự hướng dẫn của giáo sư tâm lý học A. George Miller. Các cơ sở dữ liệu và phần mềm công cụ đã được phát hành theo một giấy phép kiểu BSD và có thể được tải về và sử dụng một cách tự do. Các cơ sở dữ liệu cũng có thể được duyệt trực tuyến. WordNet đã được hỗ trợ từ các Quỹ khoa học quốc gia, DARPA, các DTO (trước đây là Advanced Research and Development Activity), và REFLEX.

Trong WordNet, các từ được xếp vào các nhóm đồng nghĩa gọi là synsets (tập các nghĩa có thể thay thế nhau trong một ngữ cảnh nào đó), cung cấp các định nghĩa ngắn, tổng quát và ghi lại những quan hệ ngữ nghĩa khác nhau giữa các nhóm từ đồng nghĩa. Các nhóm đồng nghĩa liên kết với nhau thông qua các quan hệ ngữ nghĩa được xây dựng thông qua các nghiên cứu về cách sử dụng, cách lưu trữ các tri thức về ngôn ngữ trong bộ não con người. Nhờ cách tổ chức như vậy, WordNet đã cung cấp nhiều tri thức hữu dụng cho việc xử lý ngôn ngữ tự nhiên. Nó được sử dụng với 2 mục đích: để tạo ra sự kết hợp giữa từ điển và từ điển đồng nghĩa có thể sử dụng hợp lý hơn, và để hỗ trợ phân tích văn bản tự động và ứng dụng trí tuệ nhân tạo.

Similarity Wordnet: là một công cụ phát triển dựa trên bộ từ điển Wordnet. Nó tính toán mức độ tương tự ngữ nghĩa giữa hai từ và cho ra kết quả là một giá trị trong khoảng [0,1]. Việc sử dụng công cụ này giúp ta vẫn hiểu được ngữ nghĩa của từ nếu người dùng nhập vào một từ đồng nghĩa.

Ví dụ: *author* và *writer* là hai từ có cùng ngữ nghĩa. Nếu sử dụng công cụ này, độ tương tự giữa hai từ sẽ là *1.0*.

# CHƯƠNG 2: ĐỀ XUẤT PHƯƠNG PHÁP XỬ LÝ CÂU HỎI NGƯỜI DÙNG ĐỂ TÌM KIẾM BÀI BÁO

Từ các nghiên cứu tham khảo, nhóm đề xuất một phương pháp xử lý ngôn ngữ tự nhiên dựa trên việc rút trích các bộ ba từ trong câu hỏi và biến chúng thành câu truy vấn SQL xuống cơ sở dữ liệu. Phương pháp đề ra một số luật về từ loại để rút ra được bộ ba (ví dụ: NN – VB – NN). Do hệ thống của nhóm chỉ riêng về một lĩnh vực: các bài báo liên quan đến công nghệ thông tin, nên không thể áp dụng một số các từ điển như Wordnet, hoặc công cụ như Gate để nhận diện thực thể. Thay vào đó, nhóm tạo ra một tập tin cấu hình ngữ nghĩa cho cơ sở dữ liệu. Tập tin này sẽ định nghĩa các mối quan hệ giữa bảng (table) với các thuộc tính trong bảng (properties), hay giữa bảng với bảng. Thêm vào đó, tập tin còn liệt kê một số từ (cụm từ) cho từng mối quan hệ nêu trên.

1. **Tạo tập tin cấu hình ngữ nghĩa cho cơ sở dữ liệu**

Bởi vì thông tin trên Database chưa thật sự miêu tả rõ các mối quan hệ ngữ nghĩa giữa các bảng với nhau hoặc giữa bảng với các thuộc tính của bảng đó. Chẳng hạn, với một bảng có tên là Book và nó có thuộc tính là publisher, ta có thể định nghĩa quan hệ giữa bảng Book và thuộc tính publisher là “is published”.

Hơn nữa, tên của các bảng hay thuộc tính trong database đôi lúc được đặt bởi người dùng, điều đó gây ra sự khó khăn cho việc nhận dạng các thực thể của câu hỏi.

Ví dụ: với câu hỏi *Who write “*Active Database Systems*”*. Giả sử trong database, bảng chứa thông tin để trả lời câu hỏi vừa rồi có tên là dblp\_pub\_new. Đây là một tên được đặt bởi người tạo ra database, cho nên nó không mang một ngữ nghĩa nào hết. Vì thế, việc tận dụng các từ điển đồng nghĩa để xác định sự xuất hiện của tên bảng trong câu trở nên khó khăn. Nhưng nếu ta đặt tên bảng *dblp\_pub\_new* bằng một tên tượng trưng khác có nghĩa như *publication* thì việc nhận dạng sẽ dễ dàng hơn nhiều.

Đồng thời, cần có một thông tin để miêu tả các mối quan hệ giữa bảng và các thuộc tính trong bảng, ánh xạ giữa các bảng nếu các bảng đó có quan hệ với nhau. Điều này sẽ giúp ích nhiều trong việc phát sinh câu truy vấn SQL tự động. Từ một câu hỏi được đưa ra, dựa vào các đối tượng trong câu hỏi đó và những mối quan hệ ánh xạ của các bảng với nhau trong database, hệ thống sẽ sinh ra câu truy vấn.

Chính vì thế, nhóm đề xuất một lược đồ XML để mô tả thông tin của database và các mối quan hệ (hình 3).



Hình 3 – Lược đồ XML biểu diển mối quan hệ ngữ nghĩa cho cơ sở dữ liệu

Trong đó: (giá trị đầu là tên của nút cha, giá trị trong dấu “[]” là tên của nút con)

* Database[database-name]: tên của database.
* database [mapping-table]: là thông tin của table ở giữa dùng để ánh xạ mối quan hệ n-n giữa 2 table với nhau.
* mapping-table [table-key]: là khóa chính của table đầu tiên.
* mapping-table [related-table-key]: là khóa chính của table quan hệ.
* mapping-table [mapping-table-name]: là tên của table ánh xạ đã có sẵn trong database.
* mapping-table [name] tên của mapping-table do người dùng đặt.
* database[table]: thông tin của table trong database
* table [primary-key]: Khóa chính của table.
* table [table-name]: Tên của table.
* table[visible]: là thuộc tính dùng để thể hiện table đó có được hiện ra cho người dùng thấy hay không.
* table[table-alias]: tên gọi khác của table đó. Ví dụ: Một table có table-name là bks. Ta có thể đặt table-alias là Books để người dùng dễ hiểu.
* table[field] thông tin thuộc tính của table.
* field[field-name] tên của thuộc tính trong database.
* field[field-alias] tên gọi khác của thuộc tính.
* field[type]: Kiểu dữ liệu của thuộc tính đó.
* field[visible] thể hiện thuộc tính đó có hiện cho người dùng thấy hay không.
* field[relation]: thể hiện mối quan hệ giữa table và thuộc tính.
* relation[relation-name] tên của mối quan hệ.
* relation[related-table] tên của table quan hệ. Thông tin này sẽ hiểu rõ hơn trong ví dụ.
* relation[reversed-relation] tên quan hệ đảo.
* relation[mapping-table] tên của table-mapping ở giữa 2 table

Ví dụ: với database của DBLP, ta có thể xây dựng file XML sau:

<database>

<!-- Tên do người dùng đặt thay thế cho dblp\_pub\_new -->

<table table-alias="publication">

<table-name>dblp\_pub\_new</table-name>

<!--Thông tin này không được hiển thị cho người dùng-->

<field type="String" field-alias="author" visible="false">

<relation>

<!--Thiết lập các loại quan hệ chính sẽ xuất hiện trong câu hỏi -->

<relation-name>has author,is written</relation-name>

<!--Thiết lập các loại quan hệ đảo sẽ xuất hiện trong câu hỏi -->

<reversed-relation>write</reversed-relation>

<!-- Thông tin ánh xạ -->

<mapping-table>pub\_author</mapping-table> </relation>

</field>

<field type="String" field-alias="title" visible="true">

<relation>

<relation-name>has title</relation-name>

<!-- Vì thuộc tính này không được ánh xạ thông qua table nào khác nên nó sẽ không có mapping-table hoặc related-table -->

</relation>

<field-name>title</field-name>

</field>

<field type="String" field-alias="source" visible="true">

<relation>

<relation-name>has source</relation-name>

</relation>

<field-name>source</field-name>

</field>

<!-- Các thuộc tính khác -->

<!-- ................... -->

<field type="Int" field-alias="year" visible="true">

<relation>

<relation-name>has year,in,since</relation-name>

</relation>

<field-name>year</field-name>

</field>

<field type="String" field-alias="type" visible="true">

<relation>

<relation-name>has type</relation-name>

</relation>

<field-name>type</field-name>

</field>

<field type="String" field-alias="publisher" visible="true">

<relation>

<relation-name>has publisher,is published</relation-name>

<reversed-relation>publish</ reversed-relation >

</relation>

<field-name>publisher</field-name>

</field>

<primary-key>id</primary-key>

</table>

<!-- Các table khác -->

<!-- ................... -->

<database-name>dblp</database-name>

<mapping-table>

<!-- Tên table mapping là dblp\_author\_ref\_new trong database -->

<mapping-table-name>dblp\_author\_ref\_new</mapping-table-name>

<!-- Tên table mapping do người dùng đặt -->

<name>pub\_author</name>

<!—Khóa chính của table dblp\_pub\_new tương ứng trong table dblp\_author\_ref\_new-->

<table-key>pub\_id</table-key>

<related-table-key>author</related-table-key>

</mapping-table>

</database>

1. **Các bước xử lý câu hỏi người dùng**
   1. **Mô hình các bước xử lý**

****

* 1. **Tiền xử lý**

**Định dạng lại thực thể đặt tên**

Thực thể đặt tên là những thành phần của văn bản thuộc vào những mục sau: tên người, tên tổ chức, địa điểm, thời gian, số lượng, phần trăm …

Theo ý kiến đề xuất của nhóm, trong câu, thực thể đặt tên là những từ hoặc cụm từ được xác định bởi quy tắc sau (không xét đến các thực thể có kiểu dữ liệu là số và ngày tháng):

Những từ viết hoa hoặc những cụm từ viết hoa liên tiếp.

Ví dụ: Mark Twain, John, ...

Những từ hoặc cụm từ trong dấu ngoặc kép <“>.

Ví dụ: “Active Database Systems”.

Những từ viết hoa tòan bộ

Ví dụ: ACM, IEEE …

Việc định dạng thực thể đặt tên sẽ giúp cho việc xử lí gán nhãn từ lọai (Pos Tagger) sẽ tốt hơn. Vì xử lí Pos Tagger là quá trình xác định từ loại của một từ trong câu. Hiện nay, có các nhiều công cụ để giúp ta có thể xử lý Pos Tagger được đưa ra như OPENNLP[[5]](#footnote-6), ANNIE POS TAGGER[[6]](#footnote-7) … Mặc dù các công cụ đó đã giúp người dùng có thể gán nhãn hầu như chính xác các từ loại trong câu tiếng anh, tuy nhiên với những câu có những thực thể đặt tên là một cụm danh từ thì việc gán nhãn từ loại từng loại từ trong câu lại không chính xác về mặt ý nghĩa của cụm từ đó**.** Điển hình: Các công cụ đó chưa có thể hiểu được những thực thể đặt tên là những cụm từ viết hoa liên tiếp hay những cụm từ trong dấu <”> hoặc <’> như đã nêu mà thay vào đó là hiểu những từ riêng lẽ trong những cụm từ đó.

Chẳng hạn với câu hỏi:

*Who is the author of* ***“A Distributed Object Manager for the Smalltalk-80 System”****?*

*Thì kết quả xử lý Pos Tagger với công cụ OpenNLP sẽ cho ra kết quả như sau:*



Với kết quả trên thì kết quả của câu trên sẽ là:

*Who(*WP) *is(*VBZ) *the(DT) author(NN) of(IN) “(NN)* ***A(DT) Distributed(NNP) Object(NNP) Manager(NNP) for(IN) the(DT) Smalltalk-80(JJ) System(NNP) ”(NNP) ?***

Trong đó:

WP: Đại từ câu hỏi.

VBZ: Động từ số ít.

JJ: Tính từ

NN: danh từ số ít

NNP:Danh từ viết hoa số ít.

DT: từ hạn định

IN: Giới từ.

Công cụ ANNIE POS TAGGER sẽ cho kết quả tương tự với khác biệt là dấu ” thứ 2 được gán nhãn là NN và xuất hiện nhãn NNP CD thay vì JJ như ở OpenNLP.



Với câu trên thì dấu “ đã được gán nhãn là NNP hoặc NN. Điều này thật sự không chính xác. Hơn nữa, kết quả này cũng đã làm mất đi ý nghĩa của cụm từ ***“A Distributed Object Manager for the Smalltalk-80 System”***

Vì thế, nhóm đề xuất quy tắc định dạng lại thực thể đặt tên như sau:

Thay những cụm từ của những thực thể đặt tên đã xác định bằng quy tắc trước đó bằng một dãy các ký tự bắt đầu bằng chữ hoa tương ứng với mã số kèm theo. Như vậy, các công cụ xử lí Pos Tagger sẽ hiểu những cụm từ đó là các NNP.

Cụ thể, các thực thể đặt tên đó sẽ thay theo mẫu : NER+SỐ THỨ TỰ. Số thứ tự này và thực thể đặt tên sẽ được lưu vết lại để xác định thực thể đặt tên tương ứng với mã số.

Ví dụ: Với câu hỏi *Who is the author of* ***“A Distributed Object Manager for the Smalltalk-80 System.”****?*, kết quả xử lí Pos Tagger (bằng OpenNLP) như sau:

* Qua xử lý đặt “***A Distributed Object Manager for the Smalltalk-80 System.***” là NER001
* *Kết quả gán nhãn từ lọai: Who(WP) is(VBZ) the(DT) author(NN) of(IN) NER001(NNP) ?*

**Xây dựng danh sách Pos Tagger**

Việc xử lý Pos Tagger sẽ giúp ta dễ dàng xác định vai trò các từ trong câu. Từ đó, ta dễ dàng ánh xạ với các bộ từ điển ngữ nghĩa như Wordnet để so sánh đối chiếu, hay tìm đồng nghĩa, tìm khái niệm, tổng quá hóa... Hơn nữa nó còn giúp ta xác định các từ quan hệ và các từ mà các bước sau dùng để xác định thực thể chưa đặt tên.

Ở bước này, Nhóm sử dụng công cụ OpenNLP để gán nhãn từ loại. Và đối với các nhãn NNP có giá trị là NER+Mã Số, sẽ được gán lại giá trị đã lưu vết ở bước trên.

**Tối ưu hóa danh sách Pos Tagger ( hay đơn giản hóa danh sách)**

Mặc dù đã tiền xử lí ở bước trên, nhưng các động từ trong câu ở các kết quả ở trên vẫn cần được xác định thêm. Do đôi khi trong câu xuất hiện các ngữ động từ.

Ví dụ: động từ “look up” là sự kết hợp giữa động từ “look” và giới từ “up”. Với kết quả ở các bước xử lý trên thì “look” và “up” là hai từ riêng lẻ.

Do vậy, nhóm đề xuất phương pháp tối ưu hóa danh sách Pos Tagger cùng với sự hỗ trợ của bộ từ điển Wordnet để xác định nghĩa của từ, với các bước sau:

Chuyển các động từ về dạng đơn giản (hay nguyên mẫu). Trong câu, các động từ thường được chia bởi nhiều thì khác nhau. Động từ nên chuyển thành dạng đơn giản để tiện cho việc xử lý. Công cụ Wordnet được sử dụng để chuyển động từ về dạng nguyên mẫu. Sau đây là một số quy tắc đề xuất để đơn giản hóa cụm động từ:

* VBP/VBZ(1) + VBN(2) 🡪 VBP(2).

Ví dụ: has written 🡪 write.

* VBP/VBZ (1) + VBN (2) + VBN (3) 🡪 VBZ(2) + VBP(3)

Ví dụ : has been written 🡪 is write

* VBP/VBZ (1) + VBN (2) + VBG (3) 🡪 VBP(3)

Ví dụ: has been writing 🡪 write.

* VBP/VBZ(1) + VBG(2) 🡪 VBP(2).

Ví dụ: is writing 🡪 write.

* VBP/VBZ(1) + VBN(2) 🡪 VBZ(1) + VBP(2) (VBP/VBZ là động từ TOBE)

Ví dụ: are written 🡪 be write

* VBP/VBZ(1) + VBG(2) + VBN(3) 🡪 VBZ(1) + VBP(3)

Ví dụ: is being written 🡪 be write

* VBD(1) 🡪 VBP(1).

Ví dụ: wrote🡪 write.

* VBD(1) + VBG(2) 🡪 VBP(2).

Ví dụ: was writing 🡪 write.

* VBD(1) + VBN(2) 🡪 VBZ(1) + VBP(2)

Ví dụ: was written 🡪 be write

* VBD(1) + VBG(2) + VBN(3) 🡪 VBZ(1) + VBP(3)

Ví dụ: was being written 🡪 be write.

* MD(1) + VBN(2) 🡪 VBP(3)

Ví dụ: can write 🡪 write.

* MD(1) +VB(2) + VBN(2) 🡪VBZ(2) + VBP(3)

Ví dụ: can be written 🡪 be write

Như vậy, nếu một câu hỏi tương tự với từ “search” được thay thế cho từ “look up” thì với việc sử dụng từ điển WordNet thì ngữ nghĩa câu hỏi không thay đổi.

Ngòai ra, các mạo từ như: *the, a, an và một số ký hiệu đặc biệt như (?,:,;...)* được lọai bỏ khỏi danh sách do chúng không có ý nghĩa gì trong thao tác truy vấn.

* 1. **Liệt kê các bộ ba quan hệ về từ**

Đối với các cách xác định bộ ba của một số nghiên cứu đã nêu, ta có thể dùng một từ điển như Wordnet để xác định các thực thể đặt tên, không đặt tên, động từ quan hệ. Ngòai ra, công cụ Gate – một công cụ xử lý văn bản – cũng hỗ trợ tốt cho vấn đề xác định thực thể đặt tên dựa trên các tập luật mà Gate đề ra. Hơn thế nữa, Gate còn hỗ trợ các plugin như Gazeteer và ONCAT để hỗ trợ thêm cho việc nhận dạng thực thể. Nhưng đối với thông tin về tiêu đề sách của DBLP thì quá lớn (hơn 1,5 triệu ) nên không thể áp dụng một bộ từ điển nào như Wordnet để nhận diện các tiêu đề.

Vì thế, nhóm sẽ không sử dụng Wordnet hay Gate để nhận diện thực thể. Thay vào đó, nhóm đề xuất sử dụng một số luật từ loại để rút ra các bộ ba quan hệ từ. Từ đó, xác định chúng thuộc mối quan hệ giữa các đối tượng nào trong cơ sở dữ liệu. Cách xác định sẽ trình bày ở bước sau.

Theo cách tạo đồ thị ý niệm của Wael Salloum, một số dạng mẫu của câu được xác định trong quá trình hình thành đồ thị. Những mẫu này dựa trên cây cú pháp của câu sau khi được phân tích. Ví dụ: “The human liver secretes the bile” thì có mẫu NP-VP-NP (Noun Phrase - Verb Pharse – Noun Pharse) tương ứng với câu :

(The(DT) human(ADJ) liver(NN)) (**NP**) - (secretes(VBZ))(**VP**) - (the(DT) bile (NN))(**NP**)

Ở đây, nhóm sử dụng nhãn từ loại thay cho cây cú pháp. Vì theo nhận định, nếu dùng cây cú pháp thông tin có thể bị sai lệch nếu người dùng nhập sai cú pháp. Ngoài ra, sử dụng gán nhãn từ loại sẽ dễ dàng xử lý hơn so với cây cú pháp (giữa duyệt cây và duyệt mảng). Từ đó, nhóm đề ra những mẫu quy tắc từ lọai như sau.

Gồm 19 mẫu từ loại:

1. (NN|NNP)-1 VB-2 (NN)-3 NNP-4-NA VB-5 --> NNP-4-NA,VB-4,(NN|NNP)-1 -->DEL
2. (NN)-1 WDT-2 (NN|NNP)-3-NA VB-4 -->(NN|NNP)-3-NA,VB-4,(NN)-1 -->DEL
3. (NN)-1 WDT-2 VB-3 VB-4 IN-5 (NN|NNP|CD)-6-NA --> NN-1,VB-3 VB-4 IN-5,(NN|NNP|CD)-6 -->DEL
4. (NN|NNP)-1 VB-2 (NN|NNP)-3-NA VB-4 --> (NN|NNP)-3-NA,VB-4,(NN|NNP)-1 -->DEL
5. (WP|WDT|NNP|NN)-1 VB-2 VB-3 IN-4 (NN)-5 (NNP|CD)-6-NA --> (WP|WDT|NNP|NN)-1,VB-2 VB-3 IN-4,(NNP|CD)-6 -->DEL
6. (WP|WDT|NNP|NN)-1 VB-2 VB-3 IN-4 (NN|NNP|CD)-5-NA --> (WP|WDT|NNP|NN)-1,VB-2 VB-3 IN-4,(NN|NNP|CD)-5 -->DEL
7. (WP|WDT|NNP|NN)-1 (VBN)-2 IN-3 (NN|NNP|CD)-4-NA --> (WP|WDT|NNP|NN)-1,VB-2 IN-3,(NN|NNP|CD)-4 -->DEL
8. (NN)-1 NNP-2-NA VB-3 --> NNP-2-NA,VB-3,(NN)-1 -->DEL
9. (WP|WDT|NNP|NN)-1 (VB)-2 (NN)-3 NNP-4-NA --> (WP|WDT|NNP|NN)-1,VB-2,NNP-4
10. (WP|WDT|NNP|NN)-1 (VB)-2 (NN|NNP)-3-NA --> (WP|WDT|NNP|NN)-1,VB-2,(NN)-3
11. (WP|WDT|NNP|NN)-1 (VB)-2 IN-3 (NN)-4 NNP-5-NA --> (WP|WDT|NNP|NN)-1,VB-2,NNP-5
12. (WP|WDT|NNP|NN)-1 (VB)-2 (IN|TO)-3 (NN|NNP)-4-NA --> (WP|WDT|NNP|NN)-1,VB-2,(NN|NNP)-4
13. (NNP|NN)-1 IN-2 (NN)-3 (NNP|CD)-4-NA --> (WP|WDT|NNP|NN)-1,IN-2,(NNP|CD)-4-NA
14. (NNP|NN)-1 IN-2 (NN|NNP|CD)-3-NA --> (WP|WDT|NNP|NN)-1,IN-2,(NN|NNP|CD)-3-NA -->DEL
15. (NNP|NN)-1 TO-2 (NN|NNP|CD)-3-NA --> (WP|WDT|NNP|NN)-1,TO-2,(NN|NNP|CD)-3-NA -->DEL
16. (NNP|NN)-1 IN-2 (NN|NNP)-3 IN-4 (NN|NNP|CD)-5-NA --> (NNP|NN)-3,IN-4,(NN|NNP|CD)-5-NA
17. (NNP|NN)-1 IN-2 CD-3 IN-4 (NN|NNP)-5-NA --> (NNP|NN)-1,IN-4,(NN|NNP|CD)-5-NA
18. VB-1 (NN|NNP)-2 CD-3 --> (NN|NNP)-2,has,CD-3
19. (WP)-1 (NN)-2 VB-3 NNP-4 VBN-5 IN-6 --> NNP-4,VB-5 IN-6,(NN)-2-->DEL

Trong luật, vế trái và vế phải phân cách bởi dấu -->. Trong đó, vế trái là đầu vào, vế phải là đầu ra.

Trong mỗi vế sẽ có những từ loại, thông tin trong từ lọai sẽ cách nhau bởi dấu “–”. Trong đó:

* Vế thứ 1( NPP,NN,VBN...) là từ loại của từ.
* Vế thứ 2 (1,2,…) chỉ vị trí của từ loại trong câu.
* Vế thứ 3 (NA) (có thể có hoặc không) chỉ giá trị của từ loại đó là liên tục hay không.

Ví dụ: Who(WP) write(VB) DataBase(NNP) and(CC) Network Sytem(NNP).

Trong câu trên thì NNP sẽ có giá trị là *NA* vì nó bao gồm 2 danh từ liên tiếp nhau (là Database và Network System) và giữa chúng không có bất kỳ từ quan hệ là động từ hay giới từ nào.

* Từ DEL nói lên rằng các từ loại trong câu có được loại bỏ đi hay không. Việc xóa sẽ bắt đầu từ từ ví trí sau danh từ đầu tiên trong luật.

Ví dụ: Which book was written by Philip K. Chan in 1990?

Câu trên sẽ được chuyển thành:

Which(WP) book(NN) be(VB) write(VB) by(IN) Philip K. Chan(NNP) in(IN) 1990(CD)

Đầu tiên nó sẽ thõa mãn luật thứ 5 và nó sẽ rút ra được bộ ba

<Book, be write by, Philip K.Chan>.

Tiếp theo luật này sẽ tiếp tục xóa các từ loại kể từ từ *be* (theo luật nó sẽ bắt đầu xóa các từ loại sau: be, write,by, Philip K. Chan). Và trở thành: Which(WP) book(NN) in(IN) 1990(CD)

Tiếp theo nó sẽ chạy các luật kế tiếp nó và sao khớp với luật thứ 14 và nó rút ra được bộ ba

<book, in, 1990>

* 1. **Nhận diện thực thể**

**Nhận diện tất cả các thực thể dựa vào từ quan hệ:**

Ta dựa vào từ quan hệ trong bộ ba được liệt kê ở bước trên. Nó sẽ được đem tra vào các từ quan hệ được liệt kê cho mỗi thuộc tính ở tập tin cấu hình ngữ nghĩa nêu ở mục *2.3.3.1* .Sau đó, dựa vào cấu trúc tổ chức ngữ nghĩa của tập tin cấu hình. Ta nhận diện đó là mối quan hệ giữa hai đối tượng nào trong dữ liệu. Trong trường hợp không tìm thấy, ta có thể kết hợp với từ điển đồng nghĩa của Wordnet để hỗ trợ tìm kiếm tốt hơn (ví dụ: nếu từ quan hệ là *compose* không định nghĩa trong tập tin cấu hình, ta có thể thay thế và tìm bằng từ *write )*.

Ta có các trường hợp sau:

* Nếu tìm thấy một trường hợp duy nhất so khớp với từ quan hệ (Hình 4):

+ Nếu từ quan hệ được tìm thấy ở mối quan hệ ngược thì thứ tự quan hệ trong tập tin cấu hình sẽ đảo ngựơc lại.

Ví dụ: Who published “Active Database Systems”. Từ quan hệ sau các bước trên sẽ là publish, bộ 3 liệt kê được là <Who, publish, “Active Database Systems”>. Khi ánh xạ vào quan hệ “publish” vào XML sẽ tìm thấy một bộ ba duy nhất ở mối quan hệ ngược là <Book, publish, publisher>. Như vậy ta chỉ cần đảo ngược <publisher, publish, book> và ta sẽ có được bộ ba đã nhận diện là < “Active Database Systems” : Book, publish,Who : publisher>

+ Ngược lại nếu ở tìm thấy ở quan hệ bình thường thì thứ tự sẽ đựơc giữ nguyên.



Hình 4 - Nhận diện thực thể trong bộ ba

* Nếu tồn tại nhiều kết quả, thì dùng các phương pháp ở mục sau để nhận diện các thực thể, lọai bỏ các bộ ba không phù hợp. Việc nhận diện thực thể sẽ ưu tiên cho nhận diện thực thể chưa đặt tên hơn là nhận diện thực thể đặt tên bởi vì nhận diện thực thể đặt tên sẽ tốn chi phí thời gian nhiều hơn do việc truy xuất database với dữ liệu lớn.

Ví dụ: Với bộ ba từ là <book,in, ACM> sẽ tìm được ở các bộ ba quan hệ trong file cấu hình là :

{<Book,in, publisher>,<Book,in,year>}. Khi ta nhận diện ACM là publisher thì ta chỉ còn lại một bộ ba quan hệ duy nhất tìm thấy là <Book,in, publisher>.

* Nếu không tìm thấy bất kỳ trường hợp sao khớp nào, thì tập bộ ba quan hệ cần xét là tất cả các bộ ba trong file cấu hình rồi tiếp tục thực hiện như bước ở trên.
* Nếu không tồn tại kết quả nào, thì bộ ba qua hệ từ đó không được nhận diện, do đó câu hỏi không thể được trả lời.

**Nhận diện thực thể đặt tên**

Đối với trường hợp không nhận diện được do không tìm thấy từ quan hệ trong tập tin cấu hình, ta chỉ cần ánh xạ giá trị của thực thể đặt tên đó vào cơ sở dữ liệu:

* Nếu không xác định được thuộc tính, việc ánh xạ này sẽ được ưu tiên cho các thuộc tính đại diện của cho tên bảng (ví dụ: thuộc tính đại diện của Author sẽ là Name, của Book là title) rồi mới xét đến các thuộc tính khác.
* Nếu giá trị của thực thể có kiểu dữ liệu là chuỗi(String) thì chỉ tìm ở các thuộc tính có kiểu dữ liệu là chuỗi(String), nếu kiểu dữ liệu là số(Integer) thì sẽ tìm ở những thuộc tính có kiểu dữ liệu là số(Integer)...

Với trường này cũng áp dụng đối với một tập các bộ ba có từ quan hệ giống nhau. Phương pháp thực hiện là lấy giá trị của thực thể đặt tên ánh xạ vào miền giá trị của thuộc tính đã biết trong bộ ba. Nếu tồn tại, bộ ba đó được giữ lại, ngược lại bộ ba bị lọai bỏ.

**Nhận diện thực thể chưa đặt tên**

Ngược lại với thực thể đặt tên là thực thể không đặt tên, nó thường là các danh từ xuất hiện trong câu. Theo ý kiến của nhóm, với loại thực này, chúng không có giá trị truy vấn mà đóng vai trò để nhận diện các bảng và thuộc tính phục vụ cho tạo câu lệnh SQL. Vì thế, thực thể không đặt tên được nhận biết để đánh dấu sự xuất hiện của những bảng trong cơ sở dữ liệu.

Tuy nhiên, trong câu hỏi tự nhiên, người dùng có thể tùy ý nhập vào một từ nào đó có cùng nghĩa với từ mình đã xác định (như tên của bảng, thuộc tính). Ví dụ:

Từ “writer” sẽ đồng nghĩa với từ “author” . Nếu như không nhận biết chúng là hai từ đồng nghĩa trả lời được những câu hỏi liên quan đến writer.

Nhóm đề xuất sử dụng công cụ Similarity Wordnet[[7]](#footnote-8) để hỗ trợ cho quá trình nhận dạng thực thể chưa đặt tên. Similarity Wordnet là một công cụ có thể dùng để đánh giá độ tương đồng về nghĩa giữa hai danh từ. Nếu giữa 2 danh từ hoàn tòan cùng nghĩa, thì mức đánh giá giữa hai danh từ đó là 1. Mức thấp nhất là 0, tức là hai danh từ đó không có liên quan gì về nghĩa với nhau.

Như vậy, với cách này đầu tiên ta chỉ cần tìm các bảng mà tên của nó có độ tương đồng về nghĩa với thực thể không đặt tên trong một mức giới hạn. Theo quá trình thực nghiệm, thì nhóm chỉ xét độ tương đồng trên 6.5. Theo một thứ tự ưu tiên, đầu tiên sẽ dùng công cụ so sánh với các tên bảng trong cơ sở dữ liệu. Nếu kết quả đều không thỏa, ta so sánh tiếp với các thuộc tính trong các bảng. Nếu kết quả vẫn tiếp tục không thỏa, ta chọn thuộc tính hay tên bảng có độ tương đồng cao nhất.

Ví dụ: Với câu hỏi *Who is the writer of “*Active Database Systems*” ?*

Thực thể chưa đặt tên ở đây (ngoại trừ từ what) là writer. Giả sử trong cơ sở dữ liệu đã có chứa một bảng có tên là Author.

Bằng công cụ Similarity Wordnet, ta có thể nhận biết được từ *writer* là *author* thay vì thay vì tên bảng nào khác trong cơ sở dữ liệu.

* 1. **Sinh câu truy vấn SQL**

Ở bước cuối cùng, từ các bộ ba, ta phát sinh ra câu truy vấn. Bước này thao tác chỉ đơn giản duyệt qua các bộ ba, rút các phần tử và đưa vào câu truy vấn SQL. Câu truy vấn được chia làm 3 phần SELECT, FROM và WHERE.

* **SELECT**: Chọn đối tượng để lấy dữ liệu. Đối tượng được xác định dựa trên câu hỏi của người dùng. Nhóm dùng một số quy tắc đơn giản tương tự như tài liệu [3] trình bày.
  + Với câu hỏi Who : đối tượng có thể là Person (Author) hoặc Organization (publisher). Tùy theo thực thể nhận diện được trong bộ ba mà quyết định đối tượng cho câu SELECT.
  + Với các hỏi What và Which: xác định dựa trên danh từ đứng sau nó (vd: Which books ... thì lấy từ books là đối tượng SELECT).
  + Các trường hợp khác nếu không xác định được thì mặc định là đối tượng sách trong cơ sở dữ liệu.
* **FROM**: Xác định các bảng xuất hiện trong các bộ ba và bỏ vào câu truy vấn FROM.
* **WHERE**: Bỏ các thuộc tính nhận diện được trong bộ ba vào câu truy vấn. Ngoài ra, còn xác định các kết nối giữa các bảng dựa trên mối quan hệ được đinh nghĩa sẵn trong tập tin cấu hình.

Từ các bộ ba, ta có thể phát sinh ra câu truy vấn như sau:

* **Đối tượng truy vấn (SELECT):**
* Nếu thực thể câu hỏi là thuộc tính:
* Thuộc tính đó có *related-table*:

Ta có câu truy vấn: SELECT `related-table[table-alias]`.\*

Ví dụ: Với câu hỏi: *Who write “Database”?* Ta sẽ rút và nhận diện được bộ ba <Who: *publication.author*,write, “Database”: *publication* >. Dựa vào thông tin cấu hình ở file XML, publication.author là thuộc tính và thuộc tính đó đó có chứa related-table là Author. Nên ta có câu truy vấn:

SELECT Author.\*

* Thuộc tính không có *related-table*:

Ta có câu truy vấn: SELECT `table[table-alias]`.tên thuộc tính

Ví dụ: Với câu hỏi: *Who publish “Database”?* Ta sẽ rút và nhận diện được bộ ba <Who: *publication.publisher*, publish, “Database”: *publication* >. Dựa vào thông tin cấu hình ở file XML, *publication.publisher* là thuộc tính và thuộc tính đó đó không có chứa related-table nào. Nên ta có câu truy vấn:

SELECT `publication`.publisher

* Nếu thực thể câu hỏi là table:

Ta có câu truy vấn: SELECT ` table[table-alias]`.\*

Ví dụ: Với câu hỏi: *Which book is published by O’really?* Ta sẽ rút và nhận diện được bộ ba <book: *publication*, is publish by, “ O’really’ ”: *publisher* >. Dựa vào thông tin cấu hình ở file XML, *publication* là table. Nên ta có câu truy vấn:

SELECT `publication`.\*

**Nguồn truy vấn (FROM):**

Ta cần lập danh sách tất cả các tên table từ danh sách các bộ ba, nếu có thuộc tính có mapping-table hoặc related-table thì thêm mapping-table hoặc related-table đó vào nguồn truy xuất. Một table sẽ được đưa vào nguồn truy vấn FROM theo cấu trúc:

`table[table-name]` `table[table-alias]`

Ví dụ: Với câu hỏi: Who write “Active Database Systems” ?

Ta có bộ 3 quan hệ đã được nhận diện < Who:*Author* , write, “Active Database Systems” :*Publication* >

Ở đây, “Active Database Systems” là thuộc tính *publication* của *Author*, và nó có related-table là *publication*, mapping-table là *pub\_au*. Bởi vì mapping-table *pub\_au* có mapping-table [mapping-table-name] là `dblp\_author\_ref\_new` trùng với table[table-name] của Author nên ta sẽ không thêm vào.

Nên ta có được mẫu truy vấn sau:

SELECT DISTINCT `Author`.\*

FROM

`dblp\_author\_ref\_new` `Author`,`dblp\_pub\_new` `Publication`

Trường hợp nhiều bộ ba được nhận diện giống nhau, có cùng một thực thể giống nhau, và thực thể còn lại khác nhau ở giá trị (thực thể ở dạng plural).

Ví dụ: Who is the author of "Working Models for Uncertain Data" and "Active Database Systems"?

Ta có các bộ ba đã được nhận diện là:

<author: *Author*, of, "Working Models for Uncertain Data": *Author .publication*>, <author:*Author*, of, "Active Database Systems": *Author .publication*>

Ở đây 2 bộ ba được nhận dạng giống nhau, chỉ khác nhau ở giá trị là "Working Models for Uncertain Data" và "Active Database Systems". Ta gọi Author là thực thể *không* ở dạng plural và thực thể còn lại là ở dạng plural.

Với trường hợp này thì nguồn *From* lại bao gồm những câu truy vấn con chứa *Select*, *From*, *Where*. Ta tiếp tục đặt tên cho những câu truy vấn con đó theo quy tắc:

table[table-alias] của\_thực\_thể\_không\_ở dạng\_plural + Số thứ tự n€[1,+∞] (Ngoại trừ trường hợp đầu tiên không đánh số để tối ưu hóa câu truy vấn chính)

Trong đó :

+ Select: là table hay thuộc tính của thực thể đặt tên ở dạng plural. Bước này giống như quá trình tạo câu select ở trên.

+ From: giống như bước tạo From ở trên: xác định tất cả các table trong bộ ba.

+Where: giống như bước tạo Where ở mục dưới.

Sau đó INNER JOIN tất cả các câu truy vấn con đó với nhau với điều kiện là

table[table-alias] + Số thứ tự n. table[primary-key] = table[table-alias] + Số thứ tự n+1. table[primary-key]

Ví dụ: Với ví dụ trên, ta có câu truy vấn cho from như sau:

FROM

(SELECT `Author`.\*

FROM `dblp\_author\_ref\_new` `Author`,`dblp\_pub\_new` `Publication`

WHERE

`Publication`.`id` = `Author`.`pub\_id` AND (`Publication`.`title` = ? )) Author

INNER JOIN

(SELECT `Author`.\*

FROM `dblp\_author\_ref\_new` `Author`,`dblp\_pub\_new` `Publication`

WHERE

`Publication`.`id` = `Author`.`pub\_id` AND (`Publication`.`title` = ? )) Author1 ON Author.author = Author1.author

**Điều kiện truy vấn (WHERE):**

* Nếu thực thể là một table và có giá trị, ta dựa vào thuộc tính đại diện của table đó. Ta tạm gọi thuộc tính đại diện đó là presentation\_field. Ta có cấu trúc sau:

table[table-alias].presentation\_fied[field\_name] = giá\_trị

* Nếu thực thể là một thuộc tính:
* Nếu một thuộc tính có related\_table và có mapping table(quan hệ n-n), ta có thêm điều kiện sau(dựa vào thông tin tại XML):

table[table-alias].table[primary\_key] = mapping\_table[mapping-table-name]. mapping\_table[table-key]

AND mapping\_table[mapping-table-name].mapping\_table[related-table-key] = related-table[table-alias].related-table[primary-key]

+ Nếu thuộc tính có related-table đồng thời thuộc tính đó có giá trị, ta làm tương tự với thực thể là table, ta tạm gọi thuộc tính đại diện cho related-table đó là related\_presentation\_field, ta có thêm điều kiện sau:

related-table[table-alias].related\_presentation\_field[field-name] = giá\_trị\_thuộc\_tính

* Nếu một thuộc tính chỉ có mỗi related\_table (quan hệ n-1) , ta có điều kiện:

table[table-alias].field[field\_name] = related\_table[table\_alias].related-table[primary-key]

* Nếu thuộc tính đó đã có giá trị, ta chỉ cần thêm điều kiện:

table[table-alias].field[field\_name] = giá\_trị\_thuộc\_tính

Nếu mapping\_table[mapping-table-name] trùng với bất kì table[table-name] thì mapping\_table[mapping-table-name] trong câu truy vấn sẽ được thay bằng table[table-alias] của table đó.

Ở ví dụ trên, với câu hỏi : Who write “Active Database Systems”?

Với các bộ 3 đã được nhận diện < Who:*Author* , write, “Active Database Systems” :*Publication* >

, “Active Database Systems” là thuộc tính publication của author và có mapping-table-name là ref\_author\_book, có related-table là publication, đồng thời thuộc tính đại diện của publication là publication.title có giá trị là “Active Database Systems”, ta sinh được câu truy vấn sau:

SELECT DISTINCT `Author`.\*

FROM

`dblp\_author\_ref\_new` `Author`,`dblp\_pub\_new` `Publication`

WHERE

`Publication`.`id` = `Author`.`pub\_id` AND `Publication`.`title` = `Active Database Systems `

Ví dụ: Với câu hỏi “Who published “Active Database Systems”

Các bộ ba được tìm thấy <“Active Database Systems” : *Publication*,publish, Who : *Publication.publisher*. Ta có câu truy vấn:

SELECT Publication.publisher

FROM

`dblp\_pub\_new` `Publication`

WHERE

(`Publication`.`title` = “Active Database Systems” )

Sau khi kết hợp 3 phần, ta có được câu truy vấn hoàn chỉnh và lấy dữ liệu từ DBLP. Như mô hình hệ thống ở mục sau, thành phần xử lý câu trả lời được nêu ra chủ yếu để trình bày nội dung kết quả trên giao diện web. Ví dụ: thông tin trả về là các bài báo thì người dùng có thể nhấp chuột vào các bài báo để xem thông tin chi tiết (đường link).

* 1. **Một số ví dụ**

Các ví dụ được giải quyết với thông tin database trên dblp.

***Ví dụ 1:*** Câu hỏi “Who published “Active Database Systems”?

* Bước 1: Tiền xử lý câu hỏi
  + Bước 1.1: Định dạng lại thực thể đặt tên

Đầu vào : Câu hỏi.

Đầu ra: Câu hỏi mà thực thể đặt tên đã được định dạng lại

Kết quả: “Who published NER001?

(Thông tin NER001 tương ứng với “Active Database Systems” đã được lưu vết lại.)

* + Bước 1.2: Xây dựng danh sách Pos Tagger

Đầu vào: “Who published NER001?

Đầu ra: Danh sách Pos Tagger:

Kết quả: WP/Who VBD/published NNS/“Active Database Systems”.

* + Bước 1.3: Tối ưu hóa Pos Taggers

Đầu vào: Danh sách Pos Tagger ở bước 1.2

Đầu ra: Danh sách Pos Tagger đã tối ưu

Kết quả: WP/Who VBZ/publish NNS/“Active Database Systems”.

* Bước 2: Liệt kê các bộ 3 quan hệ về từ

Đầu vào: danh sách Pos Tagger bước 1.3

Đầu ra: Tập các bộ ba quan hệ về từ và từ loại tương ứng

Kết quả: <Who,publish, “Active Database Systems”>

* Bước 3: Nhận diện thực thể
  + Bước 3.1: Nhận diện các thực thể dựa vào từ quan hệ

Đầu vào: Bộ 3 quan hệ từ ở bước 2

Đầu ra: Bộ ba quan hệ <thực thể, loại quan hệ, thực thể>.

Kết quả: <“Active Database Systems”:publication,be publish,publisher>

* Bước 4: Nhận dạng quan hệ ẩn

Đầu vào: Kết quả bước 3.1

Đầu ra: Bộ 3 quan hệ ẩn

Kết quả: <publication, has title, “Active Database Systems”: title>

* Bước 5: Sinh câu truy vấn  
  Đầu vào: Danh sách các bộ 3 quan hệ

Đầu ra: Câu truy vấn SQL

*Với ví dụ trên đầu vào:*

<publication, has title, “Active Database Systems”: title>và < publication, is published, Who:publisher>

*Đối tượng truy vấn :*select dblp\_pub\_new.publisher

*Nguồn truy vấn:*From dblp\_pub\_new

*Điều kiện:* Where dblp\_pub\_new.title = “Active Database Systems”

*Câu truy vấn:*

Select dblp\_pub\_new.publisher

From dblp\_pub\_new

Where dblp\_pub\_new.title = “Active Database Systems”

***Ví dụ 2:*** Câu hỏi “Who wrote “Active Database Systems”?

* Bước 1: Tiền xử lý câu hỏi.
  + Bước 1.1: Định dạng lại thực thể đặt tên

Đầu vào : Câu hỏi.

Đầu ra: Câu hỏi mà thực thể đặt tên đã được định dạng lại

Kết quả: “Who published NER001?

(Thông tin NER001 tương ứng với “Active Database Systems” đã được lưu vết lại.)

* + Bước 1.2: Xây dựng danh sách Pos Tagger

Đầu vào: Kết quả bước 1.1

Đầu ra: Danh sách Pos Tagger

Kết quả: WP/Who VBD/wrote NNS/“Active Database Systems”.

* + Bước 1.3: Tối ưu hóa Pos Taggers

Đầu vào: Danh sách Pos Tagger ở bước 1.2

Đầu ra: Danh sách Pos Tagger đã tối ưu

Kết quả: WP/Who VBZ/write NNS/“Active Database Systems”.

* Bước 2: Liệt kê các bộ 3 quan hệ về từ

Đầu vào: danh sách Pos Tagger bước 1.3

Đầu ra: Tập các bộ ba quan hệ về từ và từ loại tương ứng

Kết quả: <Who,write, “Active Database Systems”>

* Bước 3: Nhận diện thực thể
  + Bước 3.1: Nhận diện tất cả các thực thể dựa vào từ quan hệ

Đầu vào: Bộ 3 quan hệ từ ở bước 2

Đầu ra: Bộ ba <thực thể, loại quan hệ, thực thể>.

Kết quả:

<“Active Database Systems”:publication,write,Who:publication.author>

* Bước 4: Nhận dạng quan hệ ẩn

Đầu vào: Kết quả bước 3

Đầu ra: Bộ 3 quan hệ ẩn

Kết quả: <publication, has title, “Active Database Systems”: title>

* Bước 5: Sinh câu truy vấn  
  Đầu vào: Danh sách các bộ 3 quan hệ

Đầu ra: Câu truy vấn SQL

*Với ví dụ trên đầu vào:*

<publication, has title, “Active Database Systems”: title> và < publication, is written, Who:author>

*Đối tượng truy vấn :*select distinct dblp\_author\_pub\_ref.author

(Vì author là thuộc tính có mapping\_table và không có related-table)

*Nguồn truy vấn*:From dblp\_pub\_new, dblp\_author\_pub\_ref

*Điều kiện*: Where publication.title = “Active Database Systems”

and dblp\_pub\_new.id = dblp\_author\_pub\_ref.pub\_id

*Câu truy vấn:*

Select select distinct dblp\_author\_pub\_ref.author

From dblp\_pub\_new, dblp\_author\_pub\_ref

Where publication.title = “Active Database Systems”

and dblp\_pub\_new.id = dblp\_author\_pub\_ref.pub\_id

# CHƯƠNG 3: XÂY DỰNG HỆ THỐNG HỎI ĐÁP ĐỂ TÌM KIẾM BÀI BÁO KHOA HỌC

1. **Tổng quan về hệ thống**
   1. **Mô hình hệ thống:**

Hệ thống được xây dựng gồm hai luồng chính: người dùng nhập vào từ khóa để tìm bài báo và người dùng nhập câu hỏi đễ tìm kiếm.



Hình 5 – Mô hình hệ thống tìm kiếm bài báo

Tìm kiếm theo từ khóa: Người dùng nhập vào các từ khóa để tìm kiếm. Từ khóa có thể là tên bài báo, tên nhà xuất bản và nguồn của bài báo. Kết quả trả về là các bài báo có thuộc tính chứa từ khóa mà người dùng đã nhập.

Dùng câu hỏi để tìm kiếm: Thay vì nhập từ khóa. Hệ thống cho phép người dùng nhập vào một câu hỏi tự nhiên để tìm kiếm. Kết quả trả về sẽ là các bài báo phù hợp với các thông tin trong câu hỏi.

* 1. **Các chức năng trong hệ thống**

|  |  |  |
| --- | --- | --- |
| **Chức năng** | **Đầu vào** | **Đầu ra** |
| Tìm kiếm chung theo  từ khóa | Từ khóa là tên tiêu đề bài báo  (hoặc tên nguồn hoặc tên nhà xuất bản) | Danh sách các bài báo liên quan có tiêu đề (tên nguồn, tên nhà xuất bản) chứa từ khóa. |
| Tìm kiếm theo tên  tác giả | Tên tác giả | Danh sách các tác giả có tên tương tự, click chọn tác giả sẽ hiển thị các bài báo mà tác giả đó đã viết |
| Tìm kiếm nâng cao | Các thuộc tính mà người dùng có thể nhập: Năm (lớn hơn, nhỏ hơn, bằng), nhà xuất bản, nguồn, loại. | Danh sách các bài báo liên quan |
| Đa ngôn ngữ |  | Hiển thị đa ngôn ngữ |
| Tìm kiếm  hỏi đáp | Câu hỏi dạng ngôn ngữ tự nhiên | Các bài báo liên quan đến nội dung câu hỏi |

Bảng 5 – bảng các chức năng trong hệ thống

* Chức năng tìm kiếm: cung cấp chức năng tìm kiếm bài báo dựa vào từ khóa. Chức năng cho phép người dùng nhập vào từ khóa có liên quan đến tiêu đề, nguồn, nhà xuất bản.
* Chức năng tìm kiếm nâng cao: Cho phép người dùng nhập nhiều dữ liệu hơn để cho kết quả chính xác như mong muốn. Các thuộc tính mà người dùng có thể nhập: Năm (lớn hơn, nhỏ hơn hoặc bằng), nhà xuất bản, nguồn, loại.
* Chức năng tìm kiếm theo tên tác giả: Nhập vào tên tác giả, chương trình trả về các tác giả có tên tương tự. Khi chọn vào một tác giả sẽ hiển thị các bài báo mà tác giả đó đã viết. Các kết quả nhóm theo từng năm, giao diện hiển thị các bài báo mà tác giả đã viết trong mỗi năm một cách trực quan ( tương tự DBLP).
* Đa ngôn ngữ: Hỗ trợ giao diện hiển thị đa ngôn ngữ tiếng Anh-Tiếng Việt
* Chức năng hỏi đáp: Đây là một chức năng mới của chương trình. Người dùng nhập vào một câu hỏi tự nhiên duy nhất để tìm kiếm. Dựa trên phân tích câu hỏi chuyển thành câu truy vấn xuống cơ sở dữ liệu DBLP, chương trình trả về thông tin hoặc bài báo liên quan câu đến câu hỏi. Ngoài ra, người dùng được cung cấp một số câu hỏi mẫu có sẵn trên giao diện chức năng.

Ví dụ:

*-Câu hỏi:* How many publications were written by *Philip K. Chan*?

-Kết quả:

|  |
| --- |
| 1. [Experiments on Multi-Strategy Learning by Meta-Learning](http://localhost:8080/QAWeb/showPubDetail.do?id=107298)  2. [Learning Useful System Call Attributes for Anomaly Detection](http://localhost:8080/QAWeb/showPubDetail.do?id=234312)  3. [Incrementally Learning Rules for Anomaly Detection](http://localhost:8080/QAWeb/showPubDetail.do?id=235040)  …  …  36. [Systems for Knowledge Discovery in Databases](http://localhost:8080/QAWeb/showPubDetail.do?id=2138381) |

*-Câu hỏi:* Who wrote paper “[*Incrementally Learning Rules for Anomaly Detection*](http://localhost:8080/QAWeb/showPubDetail.do?id=235040)*.”*?

-Kết quả:

[*Denis Petrussenko*](http://localhost:8080/QAWeb/showPubsByAuthor.do?authorName=Denis%20Petrussenko)*,*

[*Philip*](http://localhost:8080/QAWeb/showPubsByAuthor.do?authorName=Denis%20Petrussenko)K. Chan,

1. **Thiết kế cài đặt hệ thống**
2. **Môi trường xây dựng hệ thống**

* Hệ điều hành: WindowXP/ Vista/ 7
* Chương trình chạy trên nền java 1.6
* Server: Glassfish 3.0
* Database: MySQL
* Framework MVC: Strut 1.3 theo mô hình MVC (Model – Controller- View)
* Các công cụ sử dụng: Wordnet, WNSimilarity, OpenNLP Postagger,

Hibernate 3.

1. **Cấu trúc các lớp**

* ***com.myapp.struts:*** gói chứa các lớp ngôn ngữ (gồm tiếng Anh và tiếng Việt).
* ***uit.qadbpss.core.search:*** nhân chứa các lớp xử lý tìm kiếm theo từ khóa, lớp thao tác với Wordnet.
* ***uit.qadbpss.core.wordnet:*** nhân chứa các lớp thao tác với bộ từ điển Wordnet và lớp thao tác các so sánh độ tương đồng ngữ nghĩa.
* ***uit.qadbpss.dbconfig:*** gói chứa các lớp cấu hình, lớp đại diện cho các thuộc tính trong bảng của cơ sở dữ liệu, lớp thao tác với file cấu hình XML.
* ***uit.qadbpss.model:*** lớp đối tượng đại diện cho một đối tượng dưới cơ sở dữ liệu ( dùng để truy xuất Hibernate).
* ***uit.qadbpss.preprocess:*** các lớp tiền xử lý câu hỏi của người dùng.
* ***uit.qadbpss.extracttriple:*** các lớp hỗ trợ rút các bộ ba trong câu hỏi.
* ***uit.qadbpss.entityrecog:*** Lớp nhận diện các đối tượng trong bộ ba.
* ***uit.qadbpss.generatequery:*** Lớp tạo câu truy vấn SQL từ các bộ ba rút ra được từ các lớp trên.
* ***uit.qadbpss.util:*** Lớp toàn cục cho cả hệ thống*.*Các lớp về kho ký tự CharPool, String Pool.

1. **Cơ sở dữ liệu DBLP**

Dữ liệu của đề tài lấy từ nguồn dữ liệu DBLP. Cấu trúc các bảng được trình bày như bên hình dưới:

****

* **Table Publication (dblp\_pub\_new)**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Loại** | **Giải thích** |
| id | int(8) | Khóa toàn cục trong CSDL |
| dblp\_key | varchar(150) | Khóa trong file xml (một dạng khóa khác tổ chức trong dữ liệu xml) |
| title | longtext | Tiêu để bài báo |
| source | varchar(150) | Nguồn bài báo |
| source\_id | varchar(50) | Tham chiếu đến nguồn bài báo (đường dẫn trong dblp\_key) |
| series | varchar(100) | Series của bài báo ( chỉ dành cho kiểu book và proceedings |
| year | int(4) | Năm công bố |
| type | varchar(20) | Kiểu bài báo (vd: article, proceedings, book …) |
| volume | varchar(50) | Khối lượng của nguồn, nơi tác phẩm được xuất bản |
| number | varchar(20) | Số lượng nguồn nơi mà tác phẩm được công bố |
| month | varchar(30) | Tháng xuất bản |
| pages | varchar(100) | Số trang |
| ee | varchar(200) | Đường dẫn tới phiên bản điện tử |
| ee\_PDF | varchar(200) | Đường dẫn tới phiên bản điện tử dạng PDF |
| url | varchar(150) | Địa chỉ tới trang dblp ( bắt đầu với db/…) |
| publisher | varchar(250) | Tên nhà xuất bản |
| isbn | varchar(25) | Mã ISBN |
| crossref | varchar(50) | Khóa tham chiếu chéo tới một bài báo khác |
| titleSignature | varchar(255) | Tiêu đề bài báo (không có khoảng trắng) |
| doi | varchar(255) | Digital Object Identifier |
| mdate | date | Ngày chỉnh sửa |

* **Table Author (dblp\_author\_ref\_new)**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Loại** | **Giải thích** |
| pub\_id | int(8) | Khóa tương ứng với id trong bảng dblp\_pub\_new |
| author | varchar(70) | Tên tác giả |
| editor | int(1) | Biến xác nhận tác giả có phải là người chỉnh sửa |
| author\_num | int(3) | Số thứ tự của tác giả |

* **Table Publication Reference (dblp\_ref\_new)**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Loại** | **Giải thích** |
| id | int(8) | Khóa tương ứng với id trong bảng dblp\_pub\_new |
| dblp\_key | varchar(150) | Khóa dblp\_key của những bài báo được trích dẫn |

1. **Giao diện chương trình**

* **Giao diện chính khi khởi động chương trình**

****

* **Giao diện tìm kiếm theo từ khóa**

****

* **Giao diện chạy thực nghiệm**

**Chạy thực nghiệm với câu:** *Who is the author of "Working Models for Uncertain Data"*

**Kết quả:**

****

**Chạy thực nghiệm với câu:** *Which books were written by Rafiul Ahad and Amelia Carlson ?*

**Kết quả:**

****

* + **Giao diện chạy thực nghiệm với câu**

**Chạy thực nghiệm với câu:** *Which papers cited to "Foundations of Databases"?*

**Kết quả:**

**

1. **Đánh giá kết quả**

**CHƯƠNG 4: KẾT LUẬN**

# THAM KHẢO

## Tiếng Việt

1. Đỗ Thị Thanh Tuyền (2008), “XÂY DỰNG HỆ THỐNG TRA CỨU THƯ VIỆN ĐIỆN TỬ BẰNG NGÔN NGỮ TỰ NHIÊN”.
2. Lương Quý Tịnh Hà (2009), “Xây dựng công cụ tìm kiếm tài liệu học tập bằng các truy vấn ngôn ngữ tự nhiên trên kho học liệu mở tiếng Việt”.
3. Cao Duy Trường (2008), “ Dịch câu truy vấn có cấu trúc sang đồ thị ý niệm: cách tiếp cận ít phụ thuộc vào cú pháp”.

## Tiếng Anh

1. Li and Roth (2002), Learning Question Classifier.
2. LiXin, Huang Xuan Jing, Wu Lid-de (2006), Question Classification by Ensemble Learning.
3. Dell Zhang and Wee Sun Lee (2003), Question classification using support vector machines.
4. Wael Salloum (2009), “A Question Answering System based on Conceptual Graph Formalism”.
5. Shiyan Ou, Constantin Orasan, Dalila Mekhaldi and Laura Hasler (2008), “Automatic Question Pattern Generation for Ontology-based Question”,.
6. Meltem Serhatli and Ferda N. Alpaslan (2009), “An Ontology based Question Answering System on Software Test Document Domain”.
7. Lorand Dali, Delia Rusu, Blaz Fortuna, Dunja Mladenic and Marko Grobelnik (2009) “ Question Answering Based on Semantic Grahps”.
8. Delia Rusu, Lorand Dali, Blaz Fortuna, Marko Grobelnik, Dunja Mladenic (2007) “Triplet Extraction from Sentences”.

## Online

1. Digital Bibliography & Library Project

<http://www.informatik.uni-trier.de/~ley/db/>

1. Wiki Question Types

[http://en.wikipedia.org/wiki/Question#Types](http://en.wikipedia.org/wiki/Question)

1. English Club

<http://www.englishclub.com/grammar/verbs-questions_types.htm>

1. Oxford

[http://oxforddictionaries.com/view/entry/m\_en\_gb0680290#m\_en\_gb0680290](http://oxforddictionaries.com/view/entry/m_en_gb0680290)

1. Learn English

<http://www.learnenglish.de/grammar/questiontext.htm>

1. Ontology

<http://www.w3schools.com/RDF/rdf_owl.asp>, <http://www.phpvn.org/index.php?topic=133.0;wap2>

1. SPARQL

<http://en.wikipedia.org/wiki/SPARQL>

1. Named Entity

<http://en.wikipedia.org/wiki/Named_entity_recognition>

1. GATE tool

<http://gate.ac.uk/>

1. NSIR

<http://tangra.si.umich.edu/clair/NSIR/html/nsir.cgi>

1. The START Natuaral Language Question Answering System

[http://start.csail.mit.edu](http://start.csail.mit.edu/)

1. Ask.com

<http://www.ask.com>

1. RDF ([Resource Description Framework](http://www.w3.org/RDF/))

<http://www.xml.com/pub/a/2001/01/24/rdf.html?page=2>

1. Conceptual graph

<http://en.wikipedia.org/wiki/Conceptual_graph>

1. Wordnet

<http://wordnet.princeton.edu/>

1. Pos tagging

<http://en.wikipedia.org/wiki/Part-of-speech_tagging>

1. http://dblp.uni-trier.de/xml/ [↑](#footnote-ref-2)
2. http://trec.nist.gov/ [↑](#footnote-ref-3)
3. <http://www.vocw.edu.vn/> [↑](#footnote-ref-4)
4. <http://www.gutenberg.org/wiki/Main_Page> [↑](#footnote-ref-5)
5. http://opennlp.sourceforge.net/ [↑](#footnote-ref-6)
6. http://gate.ac.uk/gate/doc/plugins.html [↑](#footnote-ref-7)
7. http://sourceforge.net/projects/wn-similarity/ [↑](#footnote-ref-8)