ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

**KHOA KHOA HỌC MÁY TÍNH**



**BÁO CÁO ĐỒ ÁN**

**MÔN CÁC HỆ CƠ SỞ TRI THỨC**

**ĐỀ TÀI:**

**HỆ THỐNG TRUY HỒI THÔNG TIN**

**VỀ LĨNH VỰC THUẬT TOÁN**

**Lớp: CS217.I21**

**GVHD: Huỳnh Thị Thanh Thương**

**Nhóm thực hiện: Anarithms**

**Trần Nguyên Khánh - 15520363**

**Nguyễn Trọng Nhân - 15520570**

**Lê Văn Hạnh - 15520197**

**Nguyễn Tuyết Nhi - 15520585**

Thành phố Hồ Chí Minh, ngày 3 tháng 7 năm 2018



**Nhận xét của giảng viên**

**LỜI CẢM ƠN**

Để hoàn thành tốt đề tài này chúng em xin trân trọng cảm ơn giảng viên môn Hệ cơ sở tri thức cô **Huỳnh Thị Thanh Thương** đã tận tình giúp đỡ, chỉ bảo hướng dẫn chúng em trong suốt quá trình thực hiện đồ án. Vì thời gian và năng lực còn có hạn chế nên không thể tránh khỏi những sai sót trong khi thực hiện đồ án nghiên cứu của mình. Rất mong được sự góp ý bổ sung của cô để đồ án của chúng em ngày càng hoàn thiện hơn.

Chúng em xin chân thành cảm ơn !

MỤC LỤC

[Chương 1 TỔNG QUAN ĐỀ TÀI 10](#_Toc518940447)

[1. Đặt vấn đề 10](#_Toc518940448)

[2. Mục tiêu nghiên cứu 11](#_Toc518940449)

[3. Đối tượng nghiên cứu 12](#_Toc518940450)

[4. Giới hạn và phạm vi nghiên cứu 12](#_Toc518940451)

[5. Phương pháp nghiên cứu 12](#_Toc518940452)

[6. Nội dung thực hiện 13](#_Toc518940453)

[Chương 2 CƠ SỞ LÝ THUYẾT 14](#_Toc518940454)

[1. Lý thuyết mô hình ngôn ngữ N-gram 14](#_Toc518940455)

[2. Lý thuyết Minimum Edit Distance (MED) 17](#_Toc518940456)

[3. Mô hình không gian vector 17](#_Toc518940457)

[4. Mô hình Bag Of Words (BOW) 18](#_Toc518940458)

[5. Cấu trúc dữ liệu Trie 19](#_Toc518940459)

[6. Vector dạng rút gọn 21](#_Toc518940460)

[7. Hàm Heuristic để tính toán độ tương đồng (bề mặt) của 2 câu 22](#_Toc518940461)

[8. Thuật toán chuyển từ vector dạng rút gọn sang cây Trie 22](#_Toc518940462)

[9. Thuật toán tính cosine giữa cây Trie và vector 23](#_Toc518940463)

[10. Thuật toán so khớp giữa 2 cây cú pháp 24](#_Toc518940464)

[11. Thuật toán gom cụm Kmeans 26](#_Toc518940465)

[12. Thuật toán Kmeans với các điểm trong không gian không biết số chiều 28](#_Toc518940466)

[13. Greedy String Tiling 30](#_Toc518940467)

[14. So khớp N-Gram 31](#_Toc518940468)

[15. Earley Parsing 32](#_Toc518940469)

[Giới thiệu 32](#_Toc518940470)

[Giải thuật Earley cơ bản 33](#_Toc518940471)

[Khởi tạo 33](#_Toc518940472)

[Thuật toán 33](#_Toc518940473)

[Dự đoán 33](#_Toc518940474)

[Duyệt 33](#_Toc518940475)

[Hoàn thiện 34](#_Toc518940476)

[16. Thuật toán tách từ Maximum Matching: 34](#_Toc518940477)

[17. Thuật toán gán nhãn từ loại Transform Based Learning (TBL) 35](#_Toc518940478)

[Chương 3 THIẾT KẾ CƠ SỞ TRI THỨC 36](#_Toc518940479)

[1. THU THẬP VÀ PHÂN LOẠI TRI THỨC 36](#_Toc518940480)

[Miền tri thức thu thập 36](#_Toc518940481)

[Nguồn thu thập 36](#_Toc518940482)

[Phương pháp thu thập: 36](#_Toc518940483)

[Kết quả thu thập 36](#_Toc518940484)

[2. MÔ HÌNH TRI THỨC CỦA ỨNG DỤNG 37](#_Toc518940485)

[Vai trò và ý nghĩa của mô hình 37](#_Toc518940486)

[Thành phần của mô hình 37](#_Toc518940487)

[3. TỔ CHỨC CƠ SỞ TRI THỨC TRÊN MÁY 40](#_Toc518940488)

[4. CÁC VẤN ĐỀ CƠ BẢN TRÊN MÔ HÌNH 40](#_Toc518940489)

[Chương 4 CHƯƠNG IV: THIẾT KẾ BỘ TÌM KIẾM 57](#_Toc518940490)

[1. Thư viện tách từ vnTokenizer 58](#_Toc518940491)

[2. Thư viện gán nhãn từ loại vnTagger 58](#_Toc518940492)

[3. Thư viện phân tích cú pháp Earley Parser 58](#_Toc518940493)

[4. Xây dựng bộ máy tìm kiếm 59](#_Toc518940494)

[Tổng quan về bộ máy tìm kiếm 59](#_Toc518940495)

[5. Tiền xử lý dữ liệu 60](#_Toc518940496)

[6. Chấm điểm và lựa chọn tài liệu 60](#_Toc518940497)

[7. Xếp hạng các tài liệu 61](#_Toc518940498)

[Chương 5 ỨNG DỤNG – HỆ THỐNG TÌM KIẾM THUẬT TOÁN 61](#_Toc518940499)

[1. Bộ kiểm lỗi chính tả: 65](#_Toc518940500)

[2. Bộ tiền xử lý 71](#_Toc518940501)

[3. Bộ mở rộng query 71](#_Toc518940502)

[4. Bộ so khớp: 72](#_Toc518940503)

[5. Bộ máy tìm kiếm 73](#_Toc518940504)

[Tính điểm: 73](#_Toc518940505)

[Xếp hạng: 74](#_Toc518940506)

[Chương 6 KẾT LUẬN 75](#_Toc518940507)

[1. Kết quả của đề tài 75](#_Toc518940508)

[2. Hạn chế của đề tài 75](#_Toc518940509)

[3. Hướng phát triển 75](#_Toc518940510)

[Chương 7 TÀI LIỆU THAM KHẢO 75](#_Toc518940511)

**DANH MỤC CÁC HÌNH ẢNH**

|  |  |  |
| --- | --- | --- |
| **SỐ** | **TÊN HÌNH ẢNH** | **TRANG** |
| 1 | Hình 1: Ví dụ về cây Trie 1 | 17 |
| 2 | Hình 2: Ví dụ về cây Trie 2 | 20 |
| 3 | Hình 3: Cây cú pháp câu Cơn mưa ngang qua | 21 |
| 4 | Hình 4: Cây cú pháp câu Cơn mừa vừa ngang qua | 22 |
| 5 | Hình 5: Tổng quan về bộ máy tìm kiếm | 55 |
| 6 | Hình 6: Mô tả luồng dữ liệu bước tiền xử lý | 57 |
| 7 | Hình 7: Mô tả luồng dữ liệu bước tìm kiếm query | 58 |
| 8 | Hình 8: Mô tả luồng dữ liệu bước tính điểm và xếp hạng | 59 |
| 9 | Hình 9: Mô tả cách tổ chức dữ liệu | 60 |

**HỢP ĐỒNG NHÓM**

**\*\*\*\*\*\*\*\*\*\*\*\***

## TÊN NHÓM: Anarithms

## KHẨU HIỆU: ĐOÀN KẾT CÙNG TIẾN

## DANH SÁCH THÀNH VIÊN

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và Tên | MSSV | Vai trò |
| 1 | Nguyễn Trọng Nhân | 15520570 | Trưởng nhóm |
| 2 | Trần Nguyên Khánh | 15520363 | Thành viên |
| 3 | Lê Văn Hạnh | 15520197 | Thành viên |
| 4 | Nguyễn Tuyết Nhi | 15520585 | Thành viên |

## MỤC TIÊU THÀNH LẬP

**•** Xây dựng tinh thần hợp tác, tạo mối quan hệ gắn kết giữa các thành viên trong nhóm nhằm nâng cao kỹ năng làm việc nhóm cũng như các kỹ mềm khác để hoàn thiện bản thân.

• Cùng nhau phấn đấu và đạt kết quả cao trong môn học, cùng trao dồi các kiến thức về môn Phân tích và Thiết kế thuật toán và một số kiến thức khác.

• Hoàn thành môn “Các hệ cơ sở tri thức” với điểm trung bình 10.0.

## CÁC QUY TẮC LÀM VIỆC NHÓM

### Những điều thành viên PHẢI làm:

* Điều 1: Luôn đúng giờ.
* Điều 2: Đặt lợi ích của nhóm lên trên lợi ích cá nhân.
* Điều 3: Tôn trọng ý kiến của mỗi cá nhân, tránh xảy ra mâu thuẫn.
* Điều 4: Cố gắng hoàn thành công việc được giao.
* Điều 5: Có ý thức tự giác, có trách nhiệm với nhóm.
* Điều 6: Biết quan tâm, lắng nghe, bày tỏ quan điểm cá nhân và chia sẻ với nhau.
* Những điều thành viên **KHÔNG ĐƯỢC** làm:
* Điều 1: Trễ giờ.
* Điều 2: Không hoàn thành công việc được giao.
* Điều 3: Gây mâu thuẫn, chia rẻ nội bộ và ảnh hưởng đến công việc chung của nhóm.
* Điều 4: Im lặng trước những mâu thuẫn.
* TIÊU CHÍ ĐÁNH GIÁ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Thang điểm  Tiêu chí | Hoàn hảo  (10) | Tốt  (9) | Tạm được  (8) | Kém  (5) |
| Trách nhiệm | Hoàn thành rất tốt công việc trong thời gian được giao, chủ động chịu trách nhiệm và sửa sai về phần việc mình được giao (nếu cần), nhiệt tình giúp đỡ mọi người trong nhóm cùng hoàn thành tốt phần việc của mình | Hoàn thành tốt công việc được phân công, đề ra | Hoàn thành công việc một cách sơ sài, không chú trọng chất lượng | Không hoàn thành công việc |
| Kỷ luật | Luôn luôn tuân thủ đúng quy định của nhóm đã đề ra, biết đóng góp cải thiện những quy định chung của cả nhóm để ngày càng tốt hơn, nhắc nhở động viên các bạn trong nhóm tuân thủ đúng quy định đã đề ra nhằm nâng cao chất lượng công việc chung của cả nhóm | Tuân thủ tốt quy định của nhóm đã đề ra | Tuân theo quy định đã đề ra một cách đối phó | Vô kỷ luật, không tuân theo những quy định đã đề ra, gây cản trở, phá hoại những quy định chung của nhóm |
| Tinh thần hợp tác | Biết hợp tác với nhau tìm ra hướng giải quyết vấn đề tốt nhất cho nhóm, đề cao sự đồng thuận của nhiều người. Biết hi sinh lợi ích cá nhân vì lợi ích chung của nhóm. Chủ động giao tiếp và phối hợp với nhau để làm tốt công việc được phân công | Biết hi sinh 1 số lợi ích cá nhân để hợp tác làm việc cùng nhau, đề cao lợi ích của nhóm | Hợp tác với 1 vài người mà mình thích mà không cần quan tâm đến nhóm | Chỉ quan tâm đến ý kiến cá nhân, không quan tâm đến người cùng nhóm, gây mâu thuẫn trong nội bộ |
| Cống hiến | Biết ra sức trao dồi kiến thức và kĩ năng cá nhân, từ đó tích cực dùng những gì mình có giúp đỡ mọi người, đưa ra ý kiến đóng góp về công việc được giao cũng như công việc chung của nhóm giúp chất lượng làm việc của nhóm tốt hơn, có sự phản hồi đối với công việc được giao để rút kinh nghiệm cá nhân và tập thể cả nhóm | Biết đưa ra ý tưởng, ý kiến của mình về việc được giao, sau khi hoàn thành công việc biết đưa ra nhận xét cũng như phản hồi đóng góp rút kinh nghiệm cá nhân và tập thể | Có đóng góp ý kiến về công việc được giao và công việc chung của nhóm. | Chỉ làm theo sự phân công của nhóm trưởng một cách sơ sài, không có phản hồi đóng góp ý kiến |

## CAM KẾT CỦA CÁC THÀNH VIÊN

Sau khi đọc điều khoản trong hợp đồng, các thành viên sẽ tuân theo và thực hiện đúng những yêu cầu đã đặt ra.

## PHÂN CÔNG CÔNG VIỆC

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **TÊN THÀNH VIÊN** | **NHIỆM VỤ** | **MỨC ĐỘ ĐÓNG GÓP** |
| 1 | Trần Nguyên Khánh | Báo cáo + Ontology + Giao diện | 100% |
| 2 | Nguyễn Trọng Nhân | Báo cáo + Kiểm lỗi chính tả, Vector Space Model | 100% |
| 3 | Nguyễn Tuyết Nhi | Báo cáo + Chuẩn bị query | 100% |
| 4 | Lê Văn Hạnh | Báo cáo + Xây dựng thư viện + Bộ máy tìm kiếm | 100% |

# TỔNG QUAN ĐỀ TÀI

## Đặt vấn đề

Ở Việt Nam, thời điểm Internet ra đời (cuối năm 1997) cũng chính là lúc chuyển mình giữa hai thế hệ trẻ, thế hệ Millennials (1984-1998) và thế hệ Z (sau năm 1998). Họ là chủ nhân tương lai của đất nước cũng như sẽ là lực lượng lao động chính trong xã hội ở hiện tại và trong tương lai gần. Hơn ai hết, họ là những người hiểu và cảm nhận rất rõ về lợi ích, tác hại, cũng như tầm quan trọng của Internet vì được sinh ra và lớn lên trong thời đại mà công nghệ thông tin nổi lên như vũ bão. Internet cung cấp cho họ rất nhiều tiện ích liên quan đến giải trí, liên lạc, và một trong những chức năng quan trọng chính là tìm kiếm thông tin. Thông tin là kho dữ liệu quan trọng để tạo thành tri thức cho con người qua các thế hệ. Vì thế việc tìm kiếm thông tin một cách hiệu quả dần trở thành bài toán lớn khiến nhiều công ty, doanh nghiệp, và cá nhân phải ra sức nghiên cứu và tìm hướng giải quyết. Hiện nay có trên thị trường có các ông lớn như Google, Bing, Yahoo,…đã và đang nắm giữ phần lớn thị trường trong lĩnh vực search engine. Bên cạnh đó còn có rất nhiều đối thủ cạnh tranh đang đe dọa vị trí độc tôn của các ông lớn này. Bởi lẽ còn có rất nhiều vấn đề về tìm kiếm mà những thương hiệu lớn như Google chưa thể giải quyết được, nên thị trường search engine vẫn rất màu mỡ cho tất cả các cá nhân hoặc doanh nghiệp nào tìm được cách giải quyết tốt hơn cho bất cứ bài toán tìm kiếm để thỏa mãn được nhu cầu người sử dụng. Đấy là thị trường thế giới, lấy tiếng anh là ngôn ngữ chính. Đối với thị trường Việt Nam, với ngôn ngữ chính là Tiếng Việt lại càng ít được hỗ trợ, ít những công cụ hiệu quả giải quyết những vấn đề tìm kiếm bằng Tiếng Việt hơn. Vì thế, nhóm nhận thấy rằng có 2 vấn đề nổi bật trong lĩnh vực truy hồi thông tin rất cần được đầu tư và nghiên cứu. Thứ nhất là cần xây dựng và nghiên cứu nhiều hơn những công cụ hỗ trợ tiếng Việt trong xử lý ngôn ngữ tự nhiên. Thứ hai là phát triển hệ thống tìm kiếm theo những lĩnh vực cụ thể.

Đối với vấn đề thứ nhất, chúng ta biết rằng xử lý ngôn ngữ đóng một vai trò quan trọng trong việc đẩy mạnh sự phát triển của CNTT Việt Nam đến với thế giới. Tuy nhiên, ông ta ta xưa nay vẫn nói: *Phong ba bão táp không bằng ngữ pháp Việt Nam*. Người Việt Nam đôi khi còn gặp khó khăn khi sử dụng tiếng Việt, ấy thế mà, giờ chúng ta cần phải dạy cho máy có thể dùng tiếng Việt như con người. Điều này quả thật rất khó khăn. Ở Việt Nam cũng đã có một số nhóm bắt đầu nghiên cứu về xử lý ngôn ngữ, bước đầu đã có những kết quả nhất định, như: các nhóm nghiên cứu đến từ các trường đại học, đại học Bách Khoa, đại học Công Nghệ, đại học Khoa Học Tự Nhiên, Viện Nghiên Cứu Công Nghệ Thông Tin, .... Nhóm Nghiên Cứu VLNP, Bên ngoài Việt Nam cũng có các nghiên cứu đến từ các nghiên cứu sinh Việt Nam tại JAIST. Phần lớn các sản phẩm hiện nay đang tập trung vào lĩnh vực dịch máy, một sản phẩm khó, và yêu cầu làm dài hạn. Ngoài ra còn có các bài toán phân loại chia nhóm văn bản.... Và đa phần chỉ dừng lại ở đề tài tiến sĩ, thạc sĩ, mang tính chất tìm hiểu, chưa hệ thống và định hướng rõ ràng. Hiện tại, đã xuất hiện cũng khá những những công cụ hỗ trợ xử lý tiếng Việt như vntk, pyvi, vnTokenizer, DongDu,… Nhưng thực tế việc xử lý ngôn ngữ ở Việt Nam vẫn còn gặp nhiều khó khăn. Vì thế nhóm em mong muốn, thông qua quá trình làm đồ án môn “hệ cơ sở tri thức” sẽ có cơ hội trải nghiệm và nghiên cứu sâu hơn về những hạn chế mà chúng ta đang đối mặt, để từ đó có thể cùng nhau tìm ra cách giải quyết cho những khó khăn này.

Xét vấn đề thứ hai, là phát triển hệ thống tìm kiếm trên một lĩnh vực cụ thể. Hiện tại chưa có bất cứ hệ thống tìm kiếm nào được xây dựng chỉ được sử dụng để tìm kiếm trên một lĩnh vực, cụ thể là lĩnh vực thuật toán. Nhưng theo thực trạng thì có thể đánh giá tình hình chung trong lĩnh vực tìm kiếm thông tin như sau:

Ngày nay, khi muốn tìm kiếm bất cứ thứ gì chúng ta đều nghĩ ngay đến Google. Đó là bởi vì Google Search đã chiếm trên 70% tổng thị phần trên toàn thế giới. Tuy nhiên, ở một số quốc gia và cộng đồng, Google Search không phải là một lựa chọn hàng đầu.Ví dụ: công cụ tìm kiếm mặc định của người dân Trung Quốc là Baidu còn ở Nga là Yandex. Ở Việt Nam, tuy Google Search vẫn dẫn đầu thị trường, nhưng một bộ phận không nhỏ người dùng lại ưa chuộng Cốc Cốc - công cụ tìm kiếm cây nhà lá vườn của người Việt. Như vậy, dễ thấy nhóm đang xây dựng một hệ thống tìm kiếm mà gần như là không có đối thủ cạnh tranh. Vì có lẽ không có bất cứ ai nghĩ rằng nên tìm kiếm cụ thể tập trung chỉ vào một lĩnh vực, như vậy kiến thức sẽ bị hạn chế và giới hạn. Nhưng nhóm khi xây dựng hệ thống này, lại mong muốn rằng, hệ thống sẽ giúp người sử dụng tiết kiệm thời gian tìm kiếm. Họ sẽ tìm được chính xác thông tin trong lĩnh vực mà họ quan tâm, không phải lướt mắt qua những loại thông tin không thuộc phàm trụ họ muốn tìm hiểu. Ví dụ như khi mình muốn tìm hiểu về “Zalo” thay vì kết quả chỉ trả về những thông tin như lịch sử của công ty Zalo, sản phẩm của công ty thì google còn trả về các bài báo với nội dung thị phi như “Cao Toàn Mỹ nắm giữ bao nhiêu phần trăm cổ phần của công ty Zalo”.

## Mục tiêu nghiên cứu

Trong phạm vi đồ án. Nhóm không mong muốn xây dựng được một hệ thống tìm kiếm tốt hơn những hệ thống tìm kiếm đang có trên thị trường. Nhóm chỉ tập trung vào việc hiện thực những kiến thức đã được học trên lớp để xây dựng được một chương trình thử nghiệm. Từ chương trình thử nghiệm ấy, nhóm sẽ có được cái nhìn rõ ràng chính xác, cặn kẽ, và thực tế hơn về hệ thống truy hồi thông tin. Qua đó nhóm cũng sẽ hiểu được những khó khăn và thuận lợi mà chúng ta đang phải đối mặt trong lĩnh vực này. Song song với 2 vấn đề chính thì nhóm cũng tự đặt ra cho mình 2 mục tiêu chính sau: Một là tập trung vào giải quyết các vấn đề xử lý tiếng Việt trong quá trình xây dựng chương trình. Hai là xây dựng hệ thống tìm kiếm chỉ tập trung vào lĩnh vực thuật toán. Ngoài ra còn có một mục tiêu phụ là trau dồi và nâng cao những kĩ năng mềm cần thiết. Mỗi nhiệm vụ sẽ được thực hiện cụ thể thông qua việc giải quyết các vấn đề lớn sau:

1. Xử lý ngôn ngữ tiếng Việt: Crawl dữ liệu, và phân tích nội dung văn bản.
2. Xây dựng hệ thống tìm kiếm: Lưu chỉ mục, và thuật toán xếp hạng.
3. Kĩ năng mềm: Teamwork, viết báo cáo, trình bày ý tưởng.

## Đối tượng nghiên cứu

Đề tài của nhóm là “Xây dựng hệ thống tìm kiếm thông tin lĩnh vực thuật toán”. Vì thế đối tượng nghiên cứu của nhóm sẽ xoay quanh các phương pháp, giải thuật, và kĩ thuật đề giải quyết bài toán xây dựng hệ thống trên. Cụ thể là những vấn đề chính sau:

* Xử lý câu query
* Tổ chức lưu trữ dữ liệu
* Thu thập tài liệu về lĩnh vực thuật toán
* Lập chỉ mục
* Tìm kiếm
* Giao diện người dùng

## Giới hạn và phạm vi nghiên cứu

Nhóm thực hiện đồ án “xây dựng hệ thống truy hồi thông tin lĩnh vực thuật toán” trong phạm vi môn học “Các hệ cơ sở tri thức” nên đồ án của nhóm sẽ còn nhiều giới hạn, cụ thể như sau:

1. Lĩnh vực tìm kiếm: Chỉ tìm kiếm trên lĩnh vực thuật toán.
2. Query: Chỉ tìm kiếm dựa trên keyword trong query mà người dùng nhập vào. Nhóm vẫn xử lý được những query sai chính tả.
3. Bộ dữ liệu: Chỉ lấy được một lượng dữ liệu rất ít từ internet (khoảng 200 bài viết).
4. Ontology: Cấu trúc hệ cơ sở tri thức do nhóm tự thiết kế.
5. Thời gian nghiên cứu: 1 học kì

## Phương pháp nghiên cứu

Nhóm sẽ nghiên cứu bằng các đọc các tài liệu tham khảo từ internet, từ sách, từ việc trao đổi với các thầy cô. Trong suốt quá trình tìm kiếm thông tin từ những nguồn trên, nhóm sẽ thử xây dựng lại thuật toán và kiểm tra tính hiệu quả đối với bộ dữ liệu của nhóm , thảo luận với nhóm để chọn ra phương pháp tốt nhất. Cụ thể phương pháp của nhóm để giải quyết những mục tiêu nghiên cứu như sau:

* Xử lý câu query: tách từ, gán nhãn cho từ, phân tích cú pháp, mở rộng câu query.
* Tổ chức lưu trữ dữ liệu: ontology tự xây dựng.
* Thu thập tài liệu về lĩnh vực thuật toán: crawl dữ liệu.
* Lập chỉ mục: tf-idf.
* Tìm kiếm: theo mô hình Bag Of Words kết hợp việc xử lý ngữ nghĩa dựa vào gán nhãn và phân tích cú pháp
* Giao diện người dùng: Xây dựng giao diện bằng ngôn ngữ lập trình Python.

## Nội dung thực hiện

Nhìn chung nhóm cần chú thực hiện và giải quyết các vấn đề tổng quát lần lượt như sau:

* Nghiên cứu lý thuyết IR.
* Tìm hiểu và lựa chọn kĩ thuật cho từng bài toán lớn trong mô hình.
* Xây dựng cơ sở tri thức cho mô hình IR.
* Xây dựng các module để giải quyết cho bài toán truy xuất và tìm kiếm.
* Xây dựng giao diện trực quan.
* Viết báo cáo.

# CƠ SỞ LÝ THUYẾT

## Lý thuyết mô hình ngôn ngữ N-gram

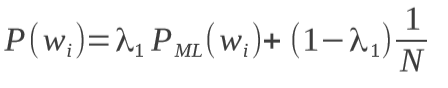
* N-gram là mô hình ngôn ngữ cho phép dự đoán được từ tiếp theo nếu cho trước một dãy N từ trước đó. Trong ứng dụng này nhóm chọn mô hình bigram kết hợp unigram(sẽ giải thích rõ hơn ở dưới).
* Mô hình ngôn ngữ với N-gram sẽ cho biết trong ngôn ngữ đó có những N-gram nào tương ứng với xác suất của nó. Dựa vào mô hình ngôn ngữ có thể phát hiện ra lỗi của văn bản.
* Nhiệm vụ của mô hình ngôn ngữ là cho biết xác suất của một câu ww...w là bao nhiêu.

Theo công thức Bayes**: P(AB) = P(B|A) \* P(A),** thì:

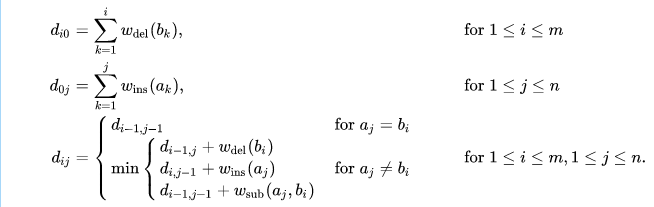
P(ww…w) = P(w) \* P(w|w) \* P(w|ww) \*…\* P(w|ww…w)

* Theo công thức này, mô hình ngôn ngữ cần phải có một lượng bộ nhớ vô cùng lớn để có thể lưu hết xác suất của tất cả các chuỗi độ dài nhỏ hơn m. Rõ ràng, điều này là không thể khi m là độ dài của các văn bản ngôn ngữ tự nhiên (m có thể tiến tới vô cùng). Để có thể tính được xác suất của văn bản với lượng bộ nhớ chấp nhận được, ta sử dụng xấp xỉ Markov bậc n:

P(w|w,w,…, w) = P(w|w,w, …,w)

* Nếu áp dụng xấp xỉ Markov, xác suất xuất hiện của một từ (w) được coi như chỉ phụ thuộc vào n từ đứng liền trước nó (ww…w) chứ không phải phụ thuộc vào toàn bộ dãy từ đứng trước (ww…w). Như vậy, công thức tính xác suất văn bản được tính lại theo công thức:
* P(ww…w) = P(w) \* P(w|w) \* P(w|ww) \*…\* P(w|ww …w)\* P(w|ww…w)
* Với công thức này, ta có thể xây dựng mô hình ngôn ngữ dựa trên việc thống kê các cụm có ít hơn n+1 từ. Mô hình ngôn ngữ này gọi là mô hình ngôn ngữ N-gram.
* Một cụm N-gram là 1 dãy con gồm N phần tử liên tiếp nhau của 1 dãy các phần tử cho trước. Với Unigram thì N là 1, còn Bigram thì N tương ứng sẽ là 2.
* **Unigram:** Là mô hình N-gram mà ở đó ngữ cảnh bị bỏ qua, chỉ quan tâm xác suất của mỗi từ đang xét, không quan tâm đến các từ lân cận nó.
* 
* Nhưng công thức trên có hạn chế khi gặp từ không có trong tài liệu thì xác suất sẽ bằng 0 vì thế ta có công thức cải tiến:
* 
* Trong đó:
* : Là xác suất các từ đã biết (tự cho là 0.95)
* PML(wi): Là xác suất của từ wi trên toàn tập dữ liệu
* N: Là số lần xuất hiện của tất cả các từ trong tài liệu
* **Bigram:** Là mô hình N-gram mà ở đó ngoài từ cần xét ta còn quan tâm đến 1 từ phía trước của nó.
* 
* Tương tự như unigram công thức trên vẫn hạn chế khi tính các từ không có trong tài liệu, vì thế ta dùng công thức
* 
* Đặc biệt khi tính bigram cần chú ý thêm từ <s> vào đầu câu và </s> vào cuối câu để tính được các từ ở đầu và cuối trong chuỗi.
* Để xây dựng được một hình ngôn ngữ hiệu quả, chúng ta phải có cách để đánh giá chúng. Trong rất nhiều cách đánh giá, nhóm sẽ chọn **Entropy** nghĩa là độ đo thông tin để đánh giá mô hình:
* **Entropy:** Entropy là thước đo thông tin, có giá trị rất lớn trong xử lý ngôn ngữ. Nó thể hiện mức độ thông tin trong ngữ pháp, thể hiện sự phù hợp của một câu với một ngôn ngữ, và dự đoán được từ tiếp theo trong cụm N-gram. Entropy của một biến ngẫu nhiên X được tính theo công thức**:**
* 
* Để tăng tính chính xác khi sử dụng độ đo entropy thì câu kiểm tra cần phải đủ dài và tổng quát (phân tán rộng) để tránh tập trung vào các xác suất lớn (chỉ chứa các cụm thông dụng).

## Lý thuyết Minimum Edit Distance (MED)

* Khoảng cách Minimum Edit Distance là số lượng ít nhất các phép toán chèn, sửa hoặc xóa các kí tự trên chuỗi kí tự A sao cho nó trở thành chuỗi kí tự B.
* **Ví dụ:** Khoảng cách MED của 2 chuỗi A = *recieve* và B = *receive* là 2, vì ta có thể thay chữ i ở vị trí thứ 4 trong chuỗi A thành chữ e, sau đó thay chữ e ở vị trí thứ 5 trong chuỗi A thành chữ i. Ta sẽ biến chuỗi A thành chuỗi B. Ở đây, ta dùng 2 phép biến đổi là thay thế.
* Khoảng cách MED được tính theo công thức dưới đây:
* 
* Ở đây, tương ứng với chi phí cho việc thực hiện phép toán xóa, chèn và thay thế (thường thì những giá trị này là 1).

## Mô hình không gian vector

* Đây là một mô hình trong những mô hình phổ biến cho hệ truy xuất thông tin
* Nội dung của mỗi tài liệu sẽ được biểu diễn như một tập hợp các từ khóa/term xuất hiện trong toàn bộ tập văn bản và hình thành một không gian mà trong đó mỗi term riêng biệt đóng vai trò là một chiều trong không gian đó.
* Gán thêm cho mỗi term một trọng số cục bộ, và giá trị của mỗi term trong mỗi tài liệu phản ánh mức độ hữu ích, tầm quan trọng của term đó trong việc mô tả nội dung hay chủ đề mà tài liệu đang đề cập tới.
* Câu truy vấn được cung cấp bởi người sử được đặc tả dưới dạng ngôn ngữ tự nhiên. Sau đó, nó sẽ được xử lý như đối với một tài liệu và được chuyển đổi thành tập các term có gán trọng số.
* Sau khi đã biểu diễn tập tài liệu và câu truy vấn thành các vector trong không gian tài liệu, bước tiếp theo là tính toán độ tương quan (giống nhau) giữa chúng bằng cách sử dụng các độ đo (ví dụ như: cosine similarity).
* **Tf-idf:** Viết tắt của thuật ngữ tiếng Anh term frequency – inverse document frequency, tf-idf là trọng số của một từ trong văn bản thu được qua thống kê thể hiện mức độ quan trọng của từ này trong một văn bản, mà bản thân văn bản đang xét nằm trong một tập hợp các văn bản.
  + **Tf – term frequency:** Dùng để ước lượng tần suất xuất hiện của từ trong văn bản. Tuy nhiên với mỗi văn bản thì có độ dài khác nhau, vì thế số lần xuất hiện của từ có thể nhiều hơn . Vì vậy số lần xuất hiện của từ sẽ được chia độ dài của văn bản (tổng số từ trong văn bản đó).
* TF(t, d) = ( số lần từ t xuất hiện trong văn bản d) / (tổng số từ trong văn bản d)
  + **IDF- Inverse Document Frequency**: dùng để ước lượng mức độ quan trọng của từ đó như thế nào. Khi tính tần số xuất hiện tf thì các từ đều được coi là quan trọng như nhau. Tuy nhiên, có một số từ thường được được sử dụng nhiều nhưng không quan trọng để thể hiện ý nghĩa của đoạn văn, ví dụ :
* Từ nối: và, nhưng, tuy nhiên, vì thế, vì vậy, …
* Giới từ: ở, trong, trên, …
* Từ chỉ định: ấy, đó, nhỉ, …
* Vì vậy ta cần giảm đi mức độ quan trọng của những từ đó bằng cách sử dụng IDF :
* IDF(t, D) = log( Tổng số văn bản trong tập mẫu D/ Số văn bản có chứa từ t )
  + Cuối cùng công thức tf-idf là: tf-idf = tf \* idf

## Mô hình Bag Of Words (BOW)

* Đây cũng là một trong những mô hình truy xuất thông tin phổ biến. Dựa trên cơ sở là phương pháp Vector Space Model, nhưng chỉ khác nhau ở chỗ cách lưu trữ thông tin hơi khác. Với VSM, lưu một tài liệu sẽ phải lưu dạng term/value với term là từ trong từ điển và value là giá trị đặt trưng của term đó trong tài liệu (ví dụ là: tf-idf), với term nào có value bằng 0 trong tài liệu (tức là nó không xuất hiện trong tài liệu) vẫn được lưu với giá trị bằng 0. Cách lưu trữ này bị thừa. BOW là phiên bản cải tiến của VSM nên đã khống chế được bằng cách lưu các term/value nào mà có value khác 0. Tưởng tượng như nó là một túi chứa các từ. Còn việc tính độ tương đồng giữa hai túi từ với nhau, ví dụ ở đây là phương pháp tính cosine, thì cách tính cũng giống như ở VSM, nhưng tốc độ nó cũng được cải thiện nhờ chỉ tính những thành phần chung của hai túi từ. Dưới đây là ví dụ để hiểu thêm:
* B1 = {(cơn, 2), (mưa, 3), (ngang, 4), (qua, 1)}
* B2 = {(cơn, 3), (lốc, 3), (ngang, 5), (qua, 10), (mạnh, 1), (quá, 4)}
* Ta có: B = B1 ∩ B2 = {cơn, ngang, qua} 🡺 giá trị cosine chỉ tính dựa trên tập hợp giao này:

Áp dụng trong trường hợp này:

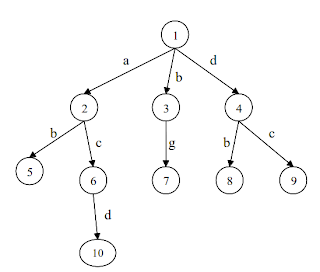
## Cấu trúc dữ liệu Trie

Trong khoa học máy tính, trie, hay cây tiền tố, là một cấu trúc dữ liệu sử dụng cây có thứ tự, dùng để lưu trữ một mảng liên kết của các xâu ký tự. Không như cây nhị phân tìm kiếm, mỗi nút trong cây không liên kết với một khóa trong mảng. Thay vào đó, mỗi nút liên kết với một xâu ký tự sao cho các xâu ký tự của tất cả các nút con của một nút đều có chung một tiền tố, chính là xâu ký tự của nút đó. Nút gốc tương ứng với xâu ký tự rỗng.

Trie là một cấu trúc dữ liệu dùng để quản lý một tập hợp các xâu. Trie cho phép:

1. Thêm một xâu vào tập hợp
2. Xóa một xâu khỏi tập hợp
3. Kiểm tra một xâu có tồn tại trong tập hợp hay không.

Trie gồm một gốc không chứa thông tin, trên mỗi cạnh lưu một ký tự, mỗi nút và đường đi từ gốc đến nút đó thể hiện 1 xâu, gồm các ký tự là các ký tự thuộc cạnh trên đường đi đó.



Hình 1: Ví dụ cây Trie 1

Trong hình vẽ trên, nút 1 là nút gốc, nút 7 thể hiện có 1 xâu là ‘bg’, nút 8 thể hiện có 1 xâu là ‘db’, nút 9 thể hiện có 1 xâu là ‘dc’, nút 10 thể hiện có 1 xâu là ‘acd’, nút 5 thể hiện là có 1 xâu là ‘ab’.

Tuy nhiên, có thể thấy, đối với một số nút, chẳng hạn nút 4, ta không biết nó là thể hiện kết thúc 1 xâu hay chỉ là 1 phần của đường đi từ nút 1 đến nút 9. Vì vậy, khi cài đặt, thông thường, tại nút U ta cần lưu thêm thông tin nút U có là kết thúc của 1 xâu hay không, hoặc nút U là kết thúc của bao nhiêu xâu, tuỳ theo yêu cầu bài toán.

Một số ưu điểm của trie:

* Cài đặt đơn giản, dễ nhớ
* Tiết kiệm bộ nhớ: Khi số lượng khóa lớn và các khóa có độ dài nhỏ, thông thường trie tiết kiệm bộ nhớ hơn do các phần đầu giống nhau của các khoá chỉ được lưu 1 lần. Ưu điểm này có ứng dụng rất lớn, chẳng hạn trong từ điển.
* Thao tác tìm kiếm: O(m) với m là độ dài khóa. Với Binary search tree (cân bằng): là O(logN). Khi số lượng khóa cần tìm lớn và độ dài mỗi khóa tương đối nhỏ, logN xấp xỉ m, và như đã biết, để cài được Binary search tree cân bằng không phải là một việc đơn giản. Hơn nữa, các thao tác trên trie rất đơn giản và thường chạy nhanh hơn trên thực tế.
* Dựa vào tính chất của cây trie, có thể thực hiện một số liên quan đến thứ tự từ điển như sắp xếp, tìm một khóa có thứ tự từ điển nhỏ nhất và lớn hơn một khóa cho trước...; và một số thao tác liên quan đến tiền tố, hậu tố.

## Vector dạng rút gọn

Vector dạng rút gọn là vector không biết trước số chiều. Nó chỉ lưu các thông tin cần thiết. Còn gọi là vector thưa. Theo mô hình Vector Space Model thì nó sẽ tạo ra một bộ từ điển các term. Sau đó, lưu các tài liệu theo kiểu là một vector tf-idf với số chiều cố định bằng với số từ trong từ điển. Tiếp theo, sẽ tính cosine giữa hai vector. Theo cách này, độ phức tạp không gian sẽ là O(n) với (n là số từ trong từ điển), độ phức tạp thời gian tính cosine giữa 1 tài liệu và một query là: O(n). Với m tài liệu thì:

Độ phức tạp không gian và thời gian là: O(n\*m)

Nếu số tài liệu là 1000, số term trong từ điển chắc cũng sẽ trên dưới 1,000,000 vậy thì độ phức tạp không gian và thời gian là không phù hợp cho hệ tìm kiếm.

Bằng cách lưu thông tin theo kiểu vector dạng rút gọn, ta có thể tránh tối đa việc lãng phí bộ nhớ, đồng thời giảm tốc độ tính toán như mô hình Vector Space Model.

Độ phức tạp đối với phương pháp này:

Độ phức tạp không gian chính bằng: O(k), với k là số từ khác nhau trong tài liệu

Độ phức tạp thời gian: O(length), length là tổng của tất cả các từ trong câu query. Cách làm để đạt được độ phức tạp này được giới thiệu ở phần **Thuật toán tính cosine giữa cây Trie và vector**.

Độ phức tạp với m tài liệu:

Độ phức tạp không gian: O(m\*k)

Độ phức tạp thời gian: O(m \* length)

Trên thực tế, k nhỏ hơn rất nhiều so với n. Vậy nên có thể nói, cách tổ chức này khá tốt.

## Hàm Heuristic để tính toán độ tương đồng (bề mặt) của 2 câu

Bằng quan sát và dựa trên lý thuyết: Ta thấy, hai câu càng tương đồng với nhau thì:

* + - 1. Giá trị MED càng nhỏ
      2. Giá trị NM càng lớn
      3. Giá trị GST càng lớn

Từ đó, hàm heuristic được nhóm đề xuất như sau:

Ở đây:

## Thuật toán chuyển từ vector dạng rút gọn sang cây Trie

Bài toán: Nhận vào vector tf-idf dạng rút gọn, trả về cây Trie của vector đó.

Mô hình hóa bài toán:

Input: Vector dạng rút gọn

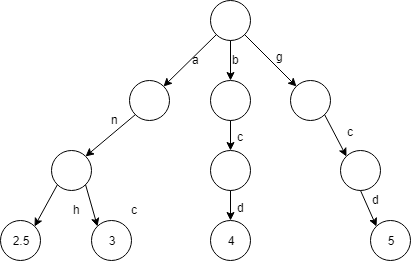
Output: Cây Trie ứng với vector đó

Thuật toán: Chạy qua từng thành phần của vector, thêm thành phần đó vào cây Trie

Ví dụ:

Vector = [(‘anh’ : 2.5), (‘anc’ : 3), (‘bcd’ : 5), (‘gcd’ : 4)]

Sẽ chuyển thành cây Trie tương ứng là:



Hình 2: Ví dụ về cây Trie 2

## Thuật toán tính cosine giữa cây Trie và vector

Bài toán: Nhận vào một vector tf-idf dạng rút gọn và một cây Trie, trả về giá trị cosine của cây Trie và vector này (bằng với giá trị cosine giữa 2 vector bình thường).

Mô hình hóa bài toán:

Input: Vector tf-idf, cây Trie

Output: Giá trị cosine

Thuật toán:

B1: Chạy qua từng cặp (từ, giá trị) đặt nó là (w1, val1) trong vector:

B1.1: Tìm kiếm từ này xem có trong cây Trie hay không? Nếu có thì đặt cặp giá trị đó là: (w1, val2)

B1.1.1: Tính giá trị tích của val1 \* val2

B1.2: Tính tổng tất cả các giá trị tích trong bước B1.1.1 đặt tổng này là s

B2: Tính giá trị cosine = s / (size(Trie) \* size(vector))

B3: Trả về giá trị cosine

Độ phức tạp thuật toán này là:

- Độ phức tạp cho việc tìm kiếm 1 từ trong cây: bằng chiều dài từ đó

- Độ phức tạp cho việc tìm kiếm tất cả các từ trong vector vào trong cây: bằng tổng số chiều dài của các từ có trong vector.

Cụ thể: O(n), n = sum(len(w)) với w thuộc vector.

## Thuật toán so khớp giữa 2 cây cú pháp

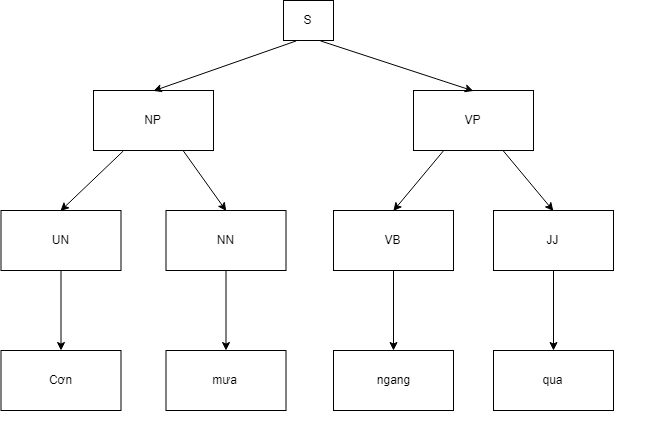
Để so sánh 2 cây cú pháp, nhóm đề xuất cách làm là so sánh dựa trên các cụm của cây cú pháp đó, ví dụ:

Ta có 2 câu:

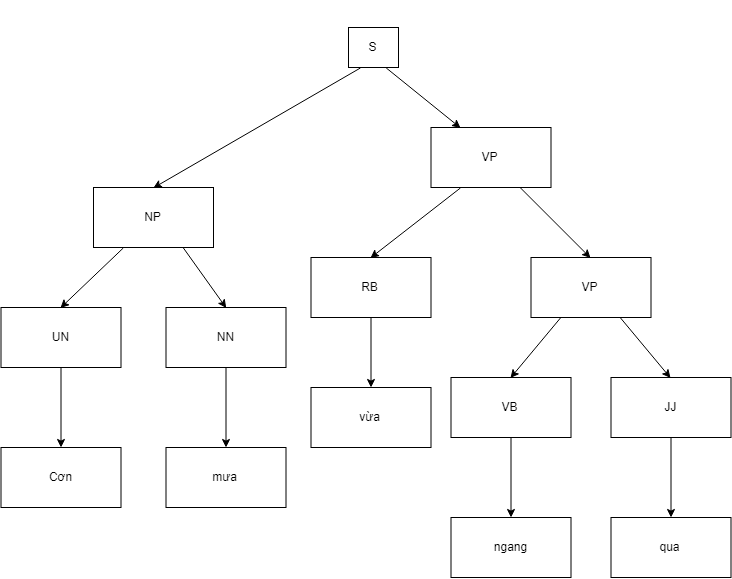
Câu 1: Cơn mưa ngang qua

Câu 2: Cơn mưa vừa ngang qua

Ta có 2 cây cú pháp tương ứng:



Hình 3: Cây cú pháp câu Cơn mưa ngang qua



Hình 4: Cây cú pháp câu Cơn mưa vừa ngang qua

Theo thuật toán, ta tách thành các cụm:

Câu 1 có các cụm:

* Set1 = {
  + - * + UN(cơn),
        + NN(mưa),
        + NP(UN(cơn)NN(mưa),
        + VB(ngang),
        + JJ(qua),
        + VP(VB(ngang)JJ(qua)),
        + S(NP(UN(cơn)NN(mưa))VP(VB(ngang)JJ(qua)))}
* len(Set1) = 7

Câu 2 tương tự có các cụm:

* Set2 = {
  + - * + UN(cơn),
        + NN(mưa),
        + NP(UN(cơn)NN(mưa),
        + VB(ngang),
        + JJ(qua),
        + RB(vừa),
        + VP(VB(ngang)JJ(qua)),
        + VP(RB(vừa),
        + VP(VB(ngang)JJ(qua)), S(NP(UN(cơn)NN(mưa))VP(RB(vừa)VP(VB(ngang)JJ(qua))))}
* Len(set2) = 9

Sau đó, ta tiến hành giao 2 cập hợp này lại: ta được tập hợp mới:

* Common set = {
  + - * + UN(cơn),
        + NN(mưa),
        + NP(UN(cơn)NN(mưa),
        + VB(ngang),
        + JJ(qua),
        + VP(VB(ngang)JJ(qua)),
      * len(Common set) = 6

maximumLengthOfCommon = len(VP(VB(ngang)JJ(qua)))) = 5 (tính theo số token)

Sau đó ta tính độ tương đồng theo công thức:

Similarity = 0.5 \* maximumLengthOfCommon / maxLen(Set1, Set2) +

0.5 \* len(Common) / min(len(Set1), len(Set2))

🡺 Similarity = 0.5 \* (5 / 14) + 0.5 \* (6 / 7) = 60,7%

## Thuật toán gom cụm Kmeans

Cho một tập hợp các phần tử S = {x1, x2, …, xn}.

B1) Chọn k phần tử bất kỳ trong S làm k cụm ban đầu.

Thông thường chọn các phần tử có khoảng cách xa nhất. Gọi gi là trọng tâm của cụm Si

B2) Với mỗi phần tử xj còn lại, chọn cụm Si có khoảng cách đến gi nhỏ nhất.

B3) Tính lại trọng tâm gi của mỗi cụm Si, bằng trung bình cộng của tất cả phần tử xj trong cụm Si

B4) Thực hiện lại bước B2 cho tới khi các phần tử xj trong các cụm Si không thay đổi, hoặc đến số lần thực hiện thứ N.

Ví dụ: gom cụm các điểm sau thành 2 cụm:

1 (1, 1); 2 (1.5, 2); 3 (3, 4); 4(5, 7); 5(3.5, 5);

6 (4.5, 5); 7(3.5, 4.5);

Cụm S1 = {1}, g1 = (1,1);

Cụm S2 = {4}, g2 = (5, 7)

Xét điểm 2 (1.5, 2):

- Khoảng cách đến S1 : d1 = 1.11

- Khoảng cách đến S2 : d2 = 6.1

→ S1 = {1, 2}, g1 = (1.25, 1.5); S2 = {4}, g2 = (5, 7)

Xét điểm 3(3,4):

- Khoảng cách đến S1 : d1 = 3.05

- Khoảng cách đến S2 : d2 = 3.6

→ S1 = {1, 2,3}, g1 = (1.83, 2.33); S2 = {4}, g2 = (5, 7)

Xét điểm 7 (3.5, 4.5):

- Khoảng cách đến S1 : d1 = 2.73

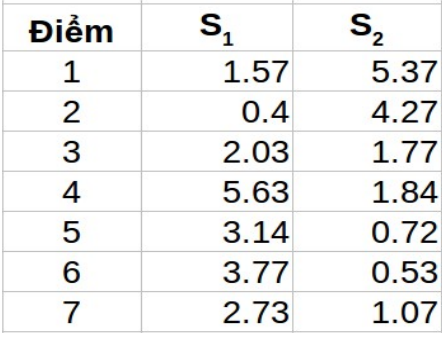
- Khoảng cách đến S2 : d2 = 1.43

→ S1 = {1, 2,3}, g1 = (1.83, 2.33); S2 ={4,5,6,7}, g2 =(4.12,5.37)

Với cụm S1 = {1, 2,3}, g1 = (1.83, 2.33); và

S2 ={4,5,6,7}, g2 =(4.12,5.37)

Xét lại khoảng cách các điểm

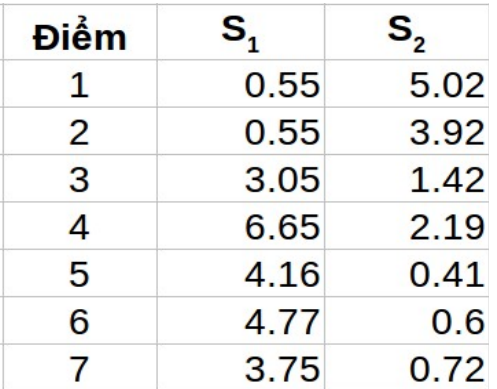


Cập nhật lại cụm:

S1 = {1, 2}, g1 = (1.25, 1.5);

và S2 ={3,4,5,6,7}, g2 =(3.9, 5.1)

Xét lại khoảng cách các điểm



Phần tử các cụm không đổi, vậy hai cụm được xác định là

S1 = {1, 2}, g1 = (1.25, 1.5); và

S2 ={3,4,5,6,7}, g2 =(3.9, 5.1)

## Thuật toán Kmeans với các điểm trong không gian không biết số chiều

Thuật toán Kmeans có 2 giai đoạn:

- Tính khoảng cách giữa các điểm và điểm trung tâm

- Tìm điểm trung tâm

Thì cụ thể điểm khác biệt giữa **không biết số chiều** và **biết số chiều** đó là: cách tính khoảng cách hơi khác.

Ta có công thức tính khoảng cách giữa 2 điểm với số chiều nhất định là:

Khoảng cách e-clit giữa hai điểm n chiều là:

Trong trường hợp nếu vector không biết số chiều như vector tf-idf dạng rút gọn thì công thức này sẽ được tính theo thuật toán dưới đây:

Bài toán: Nhận vào 2 vector tf-idf dạng rút gọn, tính khoảng cách giữa 2 vector đó.

Thuật toán:

B1: Gộp hai vector này lại thành một vector **v** duy nhất.

B2: Sắp xếp các phần tử trong vector **v** này theo thứ tự tăng dần của các từ (word)

B3:

B3.1: Gán tổng = 0;

B3.2: Chạy qua từng phần tử trong **v**:

nếu v[i].word == v[i-1].word thì:

tổng += (v[i].value - v[i-1].value) ^ 2

trường hợp khác:

tổng += v[i].value ^ 2

B4: Trả về giá trị là

Đồ án này, nhóm đề xuất cách tính khoảng cách như trên để áp dụng trong thuật toán Kmeans.

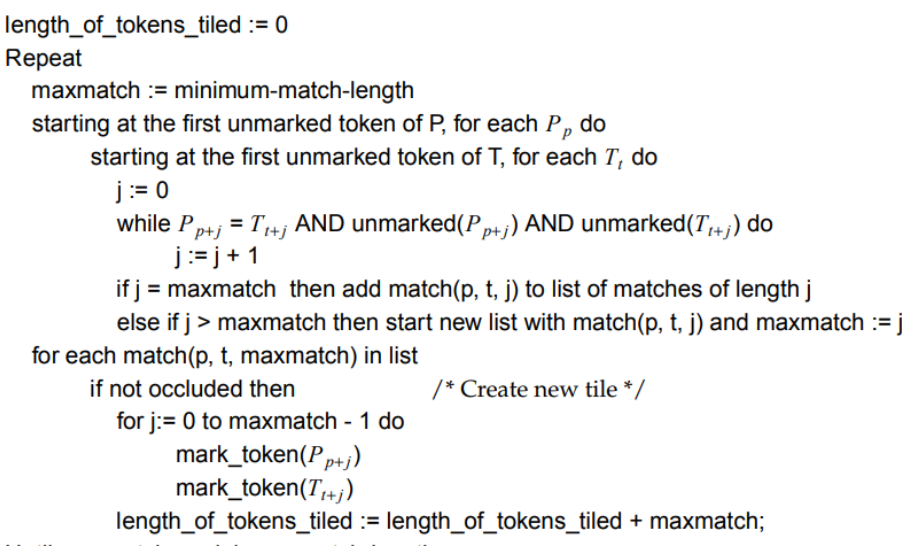
## Greedy String Tiling

Nhiệm vụ: Tính số lượng trùng khớp N các cụm từ trong hai văn bản A và B.

- N càng lớn thì A và B càng giống nhau.

- Để chuẩn hóa về đoạn [0, 1]:

Sim(A, B) = N / min(|A|,|B|)



Ví dụ:

Xác định số lượng trùng khớp giữa hai đoạn

văn bản sau:

P: c a a b a a d

T: b a a d c a a a a b a a

Giả sử minimum-match-length = 2:

\* Các match xác định được trong vòng lặp Repeat thứ

nhất:

match(4, 0, 3), match(1, 7, 5)

P: c a a b a a d

T: b a a d c a a a a b a a

\* Không tìm được các cụm giống nhau có chiều dài

lớn hơn minimum-match-length trong vòng lặp Repeat

thứ hai → kết thúc

→ length-of-token-tiled = 5

Ví dụ:

A: chú rể tặng cô dâu một bó hoa

B: cô dâu được chú rể tặng một bó hoa

Với số lượng khớp tối thiểu là 2 từ,

Lần 1:

A: chú rể tặng cô dâu một bó hoa

B: cô dâu được chú rể tặng một bó hoa

Lần 2:

A: chú rể tặng cô dâu một bó hoa

B: cô dâu được chú rể tặng một bó hoa

Số lượng trùng khớp: N = 8

- Sim(A, B) = 8 / min(8, 9) = 1

## So khớp N-Gram

Nhiệm vụ: Phân tách các đoạn văn bản A, B thành các n-gram:

- Tính số lượng n-gram N giữa A và B

- Độ tương đồng giữa A và B được chuẩn hóa về đoạn [0, 1]

Sim(A, B) = N / min(|An-gram|,|Bn-gram|)

Ví dụ:

A: chú rể tặng cô dâu một bó hoa

B: cô dâu được chú rể tặng một bó hoa

Giả sử dùng 2-gram:

A2-gram : chú rể, rể tặng, tặng cô, cô dâu, dâu một,

một bó, bó hoa

B2-gram : cô dâu, dâu được, được chú, chú rể, rể tặng,

tặng một, một bó, bó hoa

- Số 2-gram trùng khớp: N = 5

- Sim(A, B) = 5 / min(7, 8) = 5/7

## Earley Parsing

### Giới thiệu

Giải thuật Earley là một trong những giải thuật được sử dụng phổ biến trong việc xây dựng các hệ thống phân tích cú pháp. Giải thuật này sử dụng chiến lược phân tích kiểu trên xuống (top-down), bắt đầu với một ký hiệu không kết thúc đại diện cho câu và sử dụng các luật khai triển cho đến khi thu được câu vào. Hạn chế của cách tiếp cận này là không chú trọng nhiều đến các từ đầu vào. Vì vậy trong quá trình phân tích, giải thuật Earley sản sinh ra rất nhiều luật dư thừa. Ngoài ra, giải thuật Earley được xây dựng cho tiếng Anh nên khi áp dụng cho tiếng Việt sẽ có hạn chế. Mỗi câu vào tiếng Anh chỉ có một cách tách từ, trong khi với tiếng Việt, mỗi câu vào có thể có nhiều cách tách từ khác nhau. Với đặc điểm đầu vào của giải thuật Earley chỉ là một câu với một cách tách, bộ phân tích cú pháp sẽ phải thực hiện lặp đi lặp lại giải thuật này cho từng trường hợp tách từ đối với tiếng Việt. Để giải quyết vấn đề này, chúng tôi nhận thấy trong các cách tách từ Việt tồn tại các cặp cách tách giống nhau ở danh sách các từ loại đầu tiên và chỉ khác nhau ở phần đuôi của chúng. Chúng tôi sẽ tận dụng đặc điểm này trong việc cải tiến giải thuật Earley cho phân tích tiếng Việt.

### Giải thuật Earley cơ bản

Giải thuật Earley cơ bản được phát biểu như sau:

Đầu vào: Văn phạm G = (N, T, S, P), trong đó:

* N: tập kí hiệu không kết thúc
* T: tập kí hiệu kết thúc
* S: kí hiệu không kết thúc bắt đầu
* P: tập luật cú pháp Xâu vào w = a1a2 ... an

Đầu ra: Phân tích đối với w hoặc "sai"

Kí hiệu

* α, β, γ biểu diễn xâu chứa các kí hiệu kết thúc, không kết thúc hoặc rỗng
* X, Y, Z biểu diễn các kí hiệu không kết thúc đơn
* a biểu diễn kí hiệu kết thúc

Earley sử dụng cách biểu diễn luật thông qua dấu chấm “• ” X→ α • β có nghĩa :

* Trong P có một luật sản xuất X→ α β
* α đã được phân tích
* β đang được chờ phân tích
* Khi dấu chấm “• ” được chuyển ra sau β có nghĩa đây là một luật hoàn thiện. Thành phần X đã được phân tích đầy đủ, ngược lại nó là một luật chưa hoàn thiện.

Đối với mỗi từ thứ j của xâu đầu vào, bộ phân tích khởi tạo một bộ có thứ tự các trạng thái S(j). Mỗi bộ tương ứng với một cột trong bảng phân tích. Mỗi trạng thái có dạng (X → α • β, i), thành phần sau dấu phẩy xác định rằng luật này được phát sinh từ cột thứ i.

### Khởi tạo

* S(0) được khởi tạo chứa ROOT → • S.
* Nếu tại bộ cuối cùng ta có luật (ROOT → S•, 0) thì có nghĩa xâu vào được phân tích thành công.

### Thuật toán

Thuật toán phân tích thực hiện 3 bước: Dự đoán (Predictor), Duyệt (Scanner), và Hoàn thiện (Completer) đối với mỗi bộ S(j).

### Dự đoán

Với mọi trạng thái trong S(j): (X → α • Y β, i), ta thêm trạng thái (Y → • γ, j) vào S(j) nếu có luật sản xuất Y → γ trong P.

### Duyệt

Nếu a là kí hiệu kết thúc tiếp theo.

Với mọi trạng thái trong S(j): (X → α • a β, i), ta thêm trạng thái (X → α a • β, i) vào S(j+1).

### Hoàn thiện

Với mọi trạng thái trong S(j): (X → γ• , i), ta tìm trong S(i) trạng thái (Y → α • X β, k), sau đó thêm (Y → α X • β, k) vào S(j).

Ở mỗi bộ S(j) phải kiểm tra xem trạng thái đã có chưa trước khi thêm vào để tránh trùng lặp

## Thuật toán tách từ Maximum Matching:

Phương pháp Maximum Matching:

- Dùng một từ điển D chứa toàn bộ từ trong ngôn ngữ đang xét.

- Chọn từ có độ dài lớn nhất trong câu cần tách theo thứ tự từ trái qua phải (hoặc từ phải qua trái tùy theo đặc điểm ngôn ngữ).

- Nếu từ đó xuất hiện trong từ điển thì chọn, ngược lại giảm độ dài từ và kiểm tra lại.

- Tiếp tục thực hiện việc chọn từ kế tiếp.

Ví dụ: cho từ điển D = {chưa, hiện, nhưng, phù hợp, sự, sự thực, thực hiện, vẫn còn}

Tách từ cho câu sau: “nhưng sự thực hiện vẫn còn chưa phù hợp”

Theo thống kê, từ tiếng Việt thường xuất hiện có độ dài cao nhất là 4. Vì thế:

i) Xét từ bắt đầu bằng chữ nhưng:

- w1 = nhưng sự thực hiện, w1∉D.

- w1 = nhưng sự thực, w1∉ D.

- w1 = nhưng sự, w1 ∉ D.

- w1 = nhưng, w1 ∈ D. vậy từ thứ nhất là “nhưng”

ii) Xét từ bắt đầu bằng chữ “sự”:

- w2 = sự thực hiện vẫn, w2 ∉ D.

- w2 = sự thực hiện, w2 ∉ D.

- w2 = sự thực, w2 ∈ D. vậy từ thứ hai là “sự thực”

iii) Xét từ bắt đầu bằng chữ “hiện”:

- w3 = hiện vẫn còn chưa, w3 ∉ D.

- w3 = hiện vẫn còn, w3 ∉ D.

- w3 = hiện vẫn, w3 ∉ D.

- w3 = hiện, w3 ∈ D. vậy từ ba là “hiện”

iv) Xét từ bắt đầu bằng chữ “vẫn”:

- w4 = vẫn còn chưa phù, w4 ∉ D.

- w4 = vẫn còn chưa, w4 ∉ D.

- w4 = vẫn còn, w4 ∈D. Vậy từ thứ tư là “vẫn còn”

v) Xét từ bắt đầu bằng chữ “chưa”:

- w5 = chưa phù hợp, w5 ∉D.

- w5 = chưa phù, w5 ∉D.

- w5 = chưa, w5 ∈ D. Vậy từ thứ tư là “chưa”

vi) Xét từ bắt đầu bằng chữ “phù”:

- w6 = phù hợp, w6 ∈ D. Vậy chữ thứ sáu là “phù hợp”

Vậy, kết quả tách từ là: nhưng sự\_thực hiện vẫn\_còn chưa phù\_hợp

## Thuật toán gán nhãn từ loại Transform Based Learning (TBL)

Phương pháp học luật biến đổi (Transformation Based

Learning – TBL): gồm 2 pha

- Pha áp dụng luật: sử dụng lần lượt từng luật biến đổi để chuyển từ loại của mỗi từ về dạng đúng.

- Pha học luật: chọn theo thứ tự những luật nào được áp dụng nhiều nhất trên bộ dữ liệu huấn luyện. Để học luật, cần phải xác định các dạng luật có thể sử

dụng.

VD: tôi/PRN đã/RB báo/NN anh/NN

=> tôi/PRN đã/RB báo/VB anh/NN

Luật biến đổi có dạng:

if (điều kiện) && POS[wi] = From then POS[wi] = To

Trong đó:

+ wi là từ thứ i trong dữ liệu.

+ From là từ loại ban đầu của từ.

+ To là từ loại đúng của từ.

Pha áp dụng: được thực hiện theo 2 bước

B1 - Gán nhãn từ loại của mỗi từ trong dữ liệu theo từ loại có tần suất cao nhất của nó.

B2 - Lần lượt áp dụng theo thứ tự từng luật trên dữ liệu đã được gán nhãn để tạo bộ dữ liệu mới. Lặp lại bước 2 đến khi không còn có thể áp dụng được luật nào.

# THIẾT KẾ CƠ SỞ TRI THỨC

## THU THẬP VÀ PHÂN LOẠI TRI THỨC

### Miền tri thức thu thập

Thu thập tri thức về lĩnh vực Thuật toán. Thu thập các từ vựng trong lĩnh vực Thuật toán, quan hệ giữa các từ vựng và thông tin của các từ vựng.

### Nguồn thu thập

Nguồn thu thập: các bài viết về thuật toán trên trang web Diễn đàn Tin học và Thuật toán vnoi.info. Thông tin của các từ vựng được lấy trên trang web wikipedia.com.

### Phương pháp thu thập:

Crawl dữ liệu các bài viết trên trang web, tách từ vựng từ nội dung các trang web, sau đó loại bỏ các stop word thu được tập các từ vựng.

Với mỗi từ vựng, thực hiện tìm kiếm từ vựng đó trên wikipedia thu được thông tin các từ vựng.

### Kết quả thu thập

Phụ lục

## MÔ HÌNH TRI THỨC CỦA ỨNG DỤNG

### Vai trò và ý nghĩa của mô hình

Mô hình tri thức của ứng dụng lưu trữ danh sách các từ vựng, thông tin về các từ vựng và mối quan hệ giữa các từ vựng với nhau. Dựa vào đó, mô hình tri thức giúp ứng dụng mở rộng câu truy vấn và tìm kiếm ngữ nghĩa

### Thành phần của mô hình

Mô hình tri thức của ứng dụng gồm 4 thành phần:

O := (T, C, Relation, Data, Rules )

T: là tập hợp các từ vựng trong lĩnh vực thuật toán đã thu thập được

T := {“thuật\_toán”, ”đồ\_thị”, …}

C: là tập các lớp, với mỗi lớp là tập các từ vựng cùng với các thông tin của từ vựng thuộc lớp đó

C := (Object, Entity, Other)

* Object: các từ thuộc lớp Object thường là các từ chỉ khái niệm. Mỗi từ này có duy nhất một thuộc tính (thuộc thành phần Data) biểu diễn định nghĩa của nó.
* Entity: các từ thuộc lớp Entity thường là các từ chỉ một đối tượng cụ thể.
* Other: từ thuộc lớp Other là các từ còn lại, không thuộc 2 lớp trên. Các từ thuộc lớp này không có các thuộc tính trong thành phần Data

Relation: là tập các quan hệ giữa các từ vựng, mỗi tập r thuộc R là một tập hợp gồm các cặp (a,b) với a và b thuộc tập T

Relation := (isTheSameAs, isSynonymOf, isRelationOf, isLessRelationOf)

* isTheSameAs là một quan hệ 2 ngôi trên tập các từ vựng T, do đó isTheSameAs là tập con của tập T x T. isTheSameAs là một tập các cặp (a, b) với a, b thuộc tập từ vựng T thỏa mãn a là từ viết tắt hoặc là 1 cách viết khác của b và ngược lại.
* Ví dụ: isTheSameAs := {(“cntt”, “công\_nghệ\_thông\_tin”), (“thuật\_giải”, “giải\_thuật”) }
* isSynonymOf là một quan hệ 2 ngôi trên tập các từ vựng T do đó isSynonymOf là tập con của tập T x T. isSynonymOf là một tập các cặp (a, b) với a, b thuộc tập từ vựng T thỏa mãn a là từ đồng nghĩa hoặc có liên quan về nghĩa với từ b hoặc ngược lại
* Ví dụ: isSynonymOf := {(“thuật\_toán”, “thuật\_giải”), (“khoa\_học\_máy\_tính”, “computer\_science”), …}
* isRelationOf là một quan hệ 2 ngôi trên tập các từ vựng T, do đó isRelationOf là tập con của tập T x T. isRelationOf là một tập các cặp (a, b) với a, b thuộc tập từ vựng T thỏa mãn a là từ có quan hệ về nghĩa (mối quan hệ về nghĩa này nhỏ hơn so với isSynonymOf) hoặc cũng thuộc một lĩnh vực với b, ngược lại.
* Ví dụ: isRelationOf := {(“google”, “tìm\_kiếm”), (“quick\_sort”, “thuật\_toán”), …}
* isLessRelationOf là một quan hệ 2 ngôi trên tập các từ vựng T, do đó isLessRelationOf là tập con của tập T x T. isLessRelationOf là một tập các cặp (a, b) với a, b thuộc tập từ vựng T thỏa mãn a là từ có mối quan hệ về nghĩa (nhỏ hơn so với isRelationOf) với b
* Ví dụ: isLessRelationOf := {(“thuật\_giải”, “hiệu\_quả”), (“thuật\_toán”, “tìm\_kiếm”), …}

Data: là tập các thông tin của các từ vựng

Data := (Common Data, Person Data, Book Data)

Common Data: tập các thông tin chung của các từ

Common Data := {Name, Info}

* + Name: là cách viết đầy đủ của các từ. Các từ lưu trong mô hình dưới dạng các từ không có ký tự hoa, khoảng trắng giữa các tiếng trong từ được thay bằng dấu “\_”, do đó Name lưu cách viết “bình thường” của từ đó trong tự nhiên. Ví dụ: thuật\_toán : Name= “Thuật toán”
  + Info: lưu lại khái niệm của các từ thuộc lớp Object hoặc thông tin sơ lược của các từ thuộc lớp Entity

Person Data: tập các thông tin của các từ vựng chỉ người

Person Data := { Birthday, "DateOfDeath", "Education", "Field", "Nationality" }

* + - * Birthday: ngày sinh
      * DateOfDeath: ngày mấy
      * Education: trình độ học vấn
      * Field: lĩnh vực nghiên cứu
      * Nationality: quốc tịch

Book Data: tập thông tin của các từ vựng chỉ sách

Book Data := {PublicationDate, Author}

* + - * PublicationDate: ngày xuất bản
      * Author: tác giả

Rule: là tập hợp các luật của cơ sở tri thức. Các luật được xây dựng dựa trên mô hình Hệ luật dẫn.

Các sự kiện của cơ sở tri thức:

Rela = (a, b): từ a có quan hệ Rela với từ b, với Rela là một phần từ thuộc tập Relation

a = b: từ a giống với từ b

Tập luật:

Rule := { if isTheSameAs = (a, b) 🡪 a = b;

If isSynonymOf = (a, b) AND isSynonymOf = (b, c) 🡪 isSynonymOf = (a, c);

If isRelationOf = (a, b) AND isRelationOf = (b, c) 🡪 isLessRelationOf = (a, c)

}

* if isTheSameAs = (a, b) 🡪 a = b;
* Nếu 2 từ vựng có quan hệ isTheSameAs với nhau thì 2 từ đó giống nhau, cùng thuộc một class và có các thuộc tính data giống nhau
* Ví dụ: isTheSameAs = (“thuật giải”, “giải\_thuật”) 🡪 “thuật\_giải” = “giải\_thuật”
* If isSynonymOf = (a, b) AND isSynonymOf = (b, c) 🡪 isSynonymOf = (a, c)
* Nếu từ a có quan hệ isSynonymOf với từ b và từ b có quan hệ isSynonym với từ c thì từ a có quan hệ isSynonymOf với từ c
* Ví dụ: If isSynonymOf = (“graph”, “đồ\_thị”) AND isSynonymOf = (“đồ\_thị”, “đồ\_thị\_vô\_hướng”) 🡪 isSynonymOf = (“graph”, “đồ\_thị\_vô\_hướng”)
* If isRelationOf = (a, b) AND isRelationOf = (b, c) 🡪 isLessRelationOf = (a, c)
* Nếu từ a có quan hệ isRelationOf với từ b và từ b có quan hệ isRelationOf với từ c thì từ a có quan hệ isLessRelaionOf với từ c
* Ví dụ: If isRelationOf = (“thuật\_giải”, “tìm\_kiếm\_nhị\_phân”) AND isRelationOf = (“tìm\_kiếm\_nhị\_phân”, “tìm\_kiếm”) 🡪 isLessRelationOf = (“thuật\_giải”, “tìm\_kiếm”)

## TỔ CHỨC CƠ SỞ TRI THỨC TRÊN MÁY

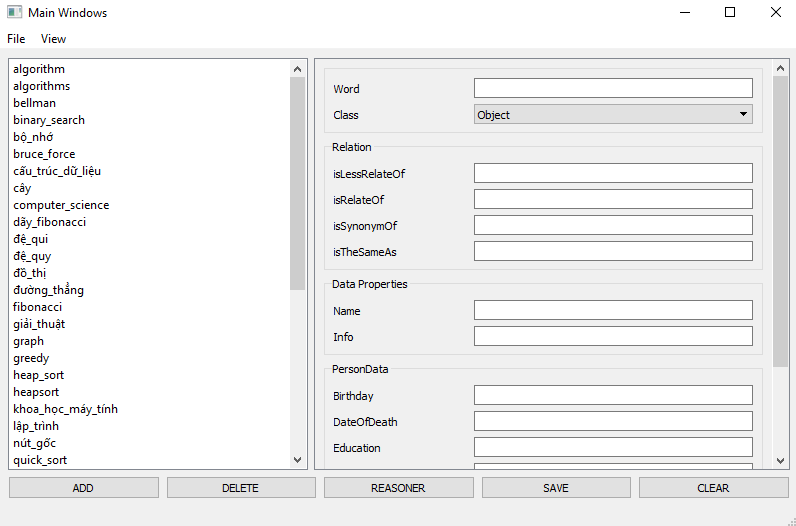
Cơ sở tri thức được xây dựng thành class Ontology và được lưu trữ trên máy bằng các file .csv. File csv chỉ là file raw lưu trữ các từ vựng cùng các thông tin của từ vựng. Khi hoạt động, ứng dụng sẽ xây dựng lại cơ sở tri thức từ các file csv.

## CÁC VẤN ĐỀ CƠ BẢN TRÊN MÔ HÌNH

Vấn đề truy cập, lưu trữ, thêm/xóa/sửa cơ sở tri thức:

Cơ sở tri thức được xây dựng thành class Ontology. Class Ontology hỗ trợ các chức năng cơ bản trên cơ sở tri thức như thêm/xóa/sửa cơ sở tri thức, xem thông tin của các từ vựng, tìm kiếm mối quan hệ giữa 2 từ vựng.

Từ class Ontology, xây dựng tool trong ứng dụng để truy cập và chỉnh sửa cơ sở tri thức.



Tool hỗ trợ xem danh sách các từ có trong cơ sở tri thức; chỉnh sửa các vựng trong cơ sở tri thức; chạy các luật (thông qua button REASONER); lưu cơ sở tri thức.

Bảng phụ lục:

|  |  |
| --- | --- |
| Word | Thông tin |
| dslk\_đơn | Một Danh sách liên kết (Linked List) là một dãy các cấu trúc dữ liệu được kết nối với nhau thông qua các liên kết (link). Hiểu một cách đơn giản thì Danh sách liên kết |
| dslk | Một Danh sách liên kết (Linked List) là một dãy các cấu trúc dữ liệu được kết nối với nhau thông qua các liên kết (link). Hiểu một cách đơn giản thì Danh sách liên kết là một cấu trúc dữ liệu bao gồm một nhóm các nút (node) tạo thành một chuỗi. Mỗi nút gồm dữ liệu ở nút đó và tham chiếu đến nút kế tiếp trong chuỗi. |
| mã\_băm | hay hàm băm là một hàm mã hóa trong máy tính. Hàm này được dùng dùng để mã hóa các dữ liệu với dung lượng bất kỳ về một loại dữ liệu với dung lượng nhất định. Hàm hash thường được dùng phổ biến với kiểu dữ liệu cơ bản như kiểu chuỗi, kiểu số, mảng, Object... nhưng nó có thể dùng với bất cứ loại dữ liệu nào thậm chỉ cả kiểu dữ liệu binary. |
| linear | Thuật toán Linear Regression mô tả dữ liệu và thể hiện mối quan hệ giữa một biến phụ thuộc với một hoặc nhiều biến độc lập. Tức là nếu biến độc lập thay đổi thì ảnh hưởng đến biến phụ thuộc như thế nào. |
| acm | ACM, viết tắt từ Association for Computing Machinery, thành lập năm 1947, là một hiệp hội quốc tế về nghiên cứu, giáo dục ngành Khoa học máy tính và Tin học uy tín nhất thế giới với hơn 100.000 hội viên, tính đến năm 2011. |
| alan | Alan Mathison Turing (23 tháng 6 năm 1912 – 7 tháng 6 năm 1954) là một nhà toán học, logic học và mật mã học người Anh thường được xem là cha đẻ của ngành khoa học máy tính. Phép thử Turing (Turing test) là một trong những cống hiến của ông trong ngành trí tuệ nhân tạo: thử thách này đặt ra câu hỏi rằng máy móc có khi nào đạt được ý thức và có thể suy nghĩ được hay không. Ông đã hình thức hóa khái niệm thuật toán và tính toán với máy Turing, đồng thời đưa ra phiên bản của "Turing", mà ngày nay được đông đảo công chúng chấp nhận, về luận đề Church-Turing, một luận đề nói rằng tất cả những gì tính được bằng thuật toán đều có thể tính được bằng máy Turing |
| ascii | ASCII (American Standard Code for Information Interchange - Chuẩn mã trao đổi thông tin Hoa Kỳ), thường được phát âm là át-xơ-ki, là bộ ký tự và bộ mã ký tự dựa trên bảng chữ cái La Tinh được dùng trong tiếng Anh hiện đại và các ngôn ngữ Tây Âu khác. Nó thường được dùng để hiển thị văn bản trong máy tính và các thiết bị thông tin khác. Nó cũng được dùng bởi các thiết bị điều khiển làm việc với văn bản. |
| bảo\_mật | Bảo mật là sự hạn chế khả năng lạm dụng tài nguyên và tài sản. Bảo mật trở nên đặc biệt phức tạp trong quản lý, vận hành những hệ thống thông tin có sử dụng các công cụ tin học, nơi có thể xảy ra và lan tràn nhanh chóng việc lạm dụng tài nguyên (các thông tin di chuyển vô hình trên mạng hoặc lưu trữ hữu hình trong các vật liệu) và lạm dụng tài sản (các máy tính, thiết bị mạng, thiết bị ngoại vi, các phần mềm của cơ quan hoặc người sở hữu hệ thống). Hạn chế ở đây có ý rằng không thể triệt phá hết ngay việc lạm dụng, cho nên cần sẵn sàng đề phòng mọi khả năng xấu với các phương cách thích hợp và chuẩn bị xử lý các sự cố nếu có việc lạm dụng xảy ra. |
| bộ\_nhớ | Bộ nhớ máy tính (tiếng Anh: Computer data storage), thường được gọi là ổ nhớ (storage) hoặc bộ nhớ (memory), là một thiết bị công nghệ bao gồm các phần tử máy tính và lưu trữ dữ liệu, được dùng để duy trì dữ liệu số. Nó là một linh kiện cơ bản có chức năng cốt lõi của các máy tính. |
| bruce\_force | Brute Force là một thuật toán vét cạn, thuật toán này sẽ chạy tất cả các trường hợp có thể có để giải quyết một vấn đề nào đó (Bao gồm cả trường hợp đúng và các trường hợp sai hay còn gọi là trường hợp dư thừa). |
| cây | Cây (tiếng Anh: Tree) là khái niệm quan trọng trong lý thuyết đồ thị, cấu trúc dữ liệu và giải thuật. Cây là một đồ thị mà trong đó hai đỉnh bất kì đều được nối với nhau bằng đúng một đường đi. Nói cách khác, đồ thị liên thông bất kỳ không có chu trình là một cây. |
| đồ\_thị | Cho đồ thị G=(X, E). Nếu chúng ta không phân biệt thứ tự của cặp đỉnh liên kết với mỗi cạnh thì sẽ có được đồ thị vô hướng. Đồ thị vô hướng G=(X, E) được định nghĩa bởi: tập hợp X ≠ ∅ được gọi là tập các đỉnh của đồ thị; tập hợp E là tập các cạnh của đồ thị. mỗi cạnh e∈E được liên kết với một cặp đỉnh {i, j} ⊆ X không phân biệt thứ tự. |
| đồ\_thị\_vô\_hướng | Cho đồ thị G=(X, E). Nếu chúng ta không phân biệt thứ tự của cặp đỉnh liên kết với mỗi cạnh thì sẽ có được đồ thị vô hướng. Đồ thị vô hướng G=(X, E) được định nghĩa bởi: tập hợp X ≠ ∅ được gọi là tập các đỉnh của đồ thị; tập hợp E là tập các cạnh của đồ thị. mỗi cạnh e∈E được liên kết với một cặp đỉnh {i, j} ⊆ X không phân biệt thứ tự. |
| chuỗi | Chuỗi ký tự là một dãy gồm các ký tự hoặc một mảng các ký tự được kết thúc bằng ký tự '\0' (còn được gọi là ký tự NULL trong bảng mã Ascii). Các hằng chuỗi ký tự được đặt trong cặp dấu nháy kép "". |
| liên\_tục | Có nhiều định nghĩa về hàm liên tục. Định nghĩa theo giới hạn của hàm là: Cho dãy {\displaystyle (x\_{n})\_{n\in \mathbb {N} }} {\displaystyle (x\_{n})\_{n\in \mathbb {N} }} bất kì trên miền xác định hội tụ về {\displaystyle c} c, thì tương ứng dãy {\displaystyle (f(x\_{n}))\_{n\in \mathbb {N} }} {\displaystyle (f(x\_{n}))\_{n\in \mathbb {N} }} hội tụ về {\displaystyle f(c)} {\displaystyle f(c)} |
| dataset | DataSet là một đối tượng có thể chứa nhiều DataTable cùng với mối liên hệ giữa chúng (relationship) và kể các ràng buộc (constraint) được lưu hoàn toàn trong bộ nhớ để làm việc offline. Qua bài viết này, bạn có thể hiểu cấu trúc của DataSet, DataTable cũng như nạp dữ liệu, tạo relation, constraint và thao tác dữ liệu trên các đối tượng dữ liệu này. |
| vertex | đỉnh trong đồ thị |
| donald\_knuth | Donald Ervin Knuth (sinh ngày 10 tháng 1, năm 1938) là một nhà khoa học máy tính nổi tiếng hiện đang là giáo sư danh dự tại Đại học Stanford. Knuth được biết đến nhiều nhất là tác giả của bộ sách Nghệ thuật lập trình máy tính (The Art of Computer Programming, TAOCP), một trong những sách tham khảo được coi trọng nhất trong ngành khoa học máy tính. Ông đã tạo ra ngành phân tích thuật toán và đã đem lại nhiều cống hiến nền tảng cho ngành khoa học máy tính lý thuyết. Ông đã tạo ra hệ thống sắp chữ TEX và hệ thống phát họa phông chữ METAFONT, và cũng là người khởi xướng khái niệm lập trình học thức (literate programming) |
| dung\_lượng | Dung lượng là một thuật ngữ đo lường để chỉ dung lượng lưu trữ máy tính. |
| trực\_quan | Dùng những vật cụ thể hay ngôn ngữ cử chỉ làm cho học sinh có được hình ảnh cụ thể về nhữngđiều được học. |
| halmiton | Đường đi Hamilton có nguồn gốc từ bài toán: "Xuất phát từ một đỉnh của khối thập nhị diện đều hãy đi dọc theo các cạnh của khối đó sao cho đi qua tất cả các đỉnh khác, mỗi đỉnh đúng một lần sau đó quay về đỉnh xuất phát." là gọi theo tên của William Rowan Hamilton phát biểu vào năm 1859. |
| bfs | duyệt theo chiều rộng |
| dfs | duyệt theo chiều sâu |
| tham\_lam | Giải thuật tham lam (tiếng Anh: Greedy algorithm) là một thuật toán giải quyết một bài toán theo kiểu metaheuristic để tìm kiếm lựa chọn tối ưu địa phương ở mỗi bước đi với hy vọng tìm được tối ưu toàn cục. |
| greedy | Giải thuật tham lam (tiếng Anh: Greedy algorithm) là một thuật toán giải quyết một bài toán theo kiểu metaheuristic để tìm kiếm lựa chọn tối ưu địa phương ở mỗi bước đi với hy vọng tìm được tối ưu toàn cục. |
| nhị\_phân | Hệ nhị phân (hay hệ đếm cơ số hai) là một hệ đếm dùng hai ký tự để biểu đạt một giá trị số, bằng tổng số các lũy thừa của 2. Hai ký tự đó thường là 0 và 1; chúng thường được dùng để biểu đạt hai giá trị hiệu điện thế tương ứng (có hiệu điện thế, hoặc hiệu điện thế cao là 1 và không có, hoặc thấp là 0). Do có ưu điểm tính toán đơn giản, dễ dàng thực hiện về mặt vật lý, chẳng hạn như trên các mạch điện tử, hệ nhị phân trở thành một phần kiến tạo căn bản trong các máy tính đương thời. |
| embed | Hệ thống nhúng (tiếng Anh: embedded system) là một thuật ngữ để chỉ một hệ thống có khả năng tự trị được nhúng vào trong một môi trường hay một hệ thống mẹ. Đó là các hệ thống tích hợp cả phần cứng và phần mềm phục vụ các bài toán chuyên dụng trong nhiều lĩnh vực công nghiệp, tự động hoá điều khiển, quan trắc và truyền tin. Đặc điểm của các hệ thống nhúng là hoạt động ổn định và có tính năng tự động hoá cao. |
| heuristic | Heuristic là các kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, học hỏi hay khám phá nhằm đưa ra một giải pháp mà không được đảm bảo là tối ưu. Với việc nghiên cứu khảo sát không có tính thực tế, các phương pháp heuristic được dùng nhằm tăng nhanh quá trình tìm kiếm với các giải pháp hợp lý thông qua các suy nghĩ rút gọn để giảm bớt việc nhận thức vấn đề khi đưa ra quyết định. Ví dụ của phương pháp này bao gồm sử dụng một luật ngón tay cái, giả thuyết, phán đoán trực giác, khuôn mẫu hay nhận thức thông thường. |
| java | Java (phiên âm Tiếng Việt: "Gia-va") là một ngôn ngữ lập trình hướng đối tượng (OOP) và dựa trên các lớp (class)[9]. Khác với phần lớn ngôn ngữ lập trình thông thường, thay vì biên dịch mã nguồn thành mã máy hoặc thông dịch mã nguồn khi chạy, Java được thiết kế để biên dịch mã nguồn thành bytecode, bytecode sau đó sẽ được môi trường thực thi (runtime environment) chạy. Trước đây, Java chạy chậm hơn những ngôn ngữ dịch thẳng ra mã máy như C và C++, nhưng sau này nhờ công nghệ "biên dịch tại chỗ" - Just in time compilation, khoảng cách này đã được thu hẹp, và trong một số trường hợp đặc biệt Java có thể chạy nhanh hơn. Java chạy nhanh hơn những ngôn ngữ thông dịch như Python, Perl, PHP gấp nhiều lần. Java chạy tương đương so với C#, một ngôn ngữ khá tương đồng về mặt cú pháp và quá trình dịch/chạy[10][11] Cú pháp Java được vay mượn nhiều từ C & C++ nhưng có cú pháp hướng đối tượng đơn giản hơn và ít tính năng xử lý cấp thấp hơn. Do đó việc viết một chương trình bằng Java dễ hơn, đơn giản hơn, đỡ tốn công sửa lỗi hơn. Trong Java, hiện tượng rò rỉ bộ nhớ hầu như không xảy ra do bộ nhớ được quản lý bởi Java Virtual Machine (JVM) bằng cách tự động "dọn dẹp rác". Người lập trình không phải quan tâm đến việc cấp phát và xóa bộ nhớ như C, C++. Tuy nhiên khi sử dụng những tài nguyên mạng, file IO, database (nằm ngoài kiểm soát của JVM) mà người lập trình không đóng (close) các streams thì rò rỉ dữ liệu vẫn có thể xảy ra. |
| computer\_science | Khoa học máy tính (tiếng Anh: computer science) là ngành nghiên cứu các cơ sở lý thuyết về thông tin và tính toán cùng sự thực hiện và ứng dụng của chúng trong các hệ thống máy tính. |
| khoa\_học\_máy\_tính | Khoa học máy tính (tiếng Anh: computer science) là ngành nghiên cứu các cơ sở lý thuyết về thông tin và tính toán cùng sự thực hiện và ứng dụng của chúng trong các hệ thống máy tính. |
| euclidean | Khoảng cách Euclidean (Euclidean Distance): Khoảng cách giữa 2 điểm là chiều dài của đường thẳng nối chúng. Trong mặt phẳng, khoảng cách giữa 2 điểm (x1, y1)  và (x2, y2) được cho bởi định lý Pythagorean như sau: |
| lập\_trình\_hướng\_đối\_tượng | Lập trình hướng đối tượng (tiếng Anh: Object-oriented programming, viết tắt: OOP) là một mẫu hình lập trình dựa trên khái niệm "công nghệ đối tượng", mà trong đó, đối tượng chứa đựng các dữ liệu, trên các trường, thường được gọi là các thuộc tính; và mã nguồn, được tổ chức thành các phương thức. Phương thức giúp cho đối tượng có thể truy xuất và hiệu chỉnh các trường dữ liệu của đối tượng khác, mà đối tượng hiện tại có tương tác (đối tượng được hỗ trợ các phương thức "this" hoặc "self"). Trong lập trình hướng đối tượng, chương trình máy tính được thiết kế bằng cách tách nó ra khỏi phạm vi các đối tượng tương tác với nhau.[1][2] Ngôn ngữ lập trình hướng đối tượng khá đa dạng, phần lớn là các ngôn ngữ lập trình theo lớp, nghĩa là các đối tượng trong các ngôn ngữ này được xem như thực thể của một lớp, được dùng để định nghĩa một kiểu dữ liệu. |
| lập\_trình | Lập trình máy tính hay lập chương trình máy tính thường gọi tắt là lập trình (tiếng Anh: Computer programming, thường gọi tắt là programming) là việc lập ra chương trình làm việc cho máy có bộ xử lý, nói riêng là máy tính, để thực thi nhiệm vụ xử lý thông tin nào đó. |
| fibonacci | Leonardo Pisano Bogollo, còn được biết đến với tên Leonardo của Pisa, Leonardo Pisano, Leonardo Bonacci, Leonardo Fibonacci, hay, phổ biến nhất, chỉ là Fibonacci, là một nhà toán học người Ý, được một số người xem là "nhà toán học tài ba nhất thời Trung Cổ. |
| lý\_thuyết\_trò\_chơi | Lý thuyết trò chơi là một nhánh của Toán học ứng dụng. Ngành này nghiên cứu các tình huống chiến thuật trong đó các đối thủ lựa chọn các hành động khác nhau để cố gắng làm tối đa kết quả nhận được. Ban đầu được phát triển như là một công cụ để nghiên cứu hành vi kinh tế học, ngày nay Lý thuyết trò chơi được sử dụng trong nhiều ngành khoa học, từ Sinh học tới Triết học. Lý thuyết trò chơi đã có sự phát triển lớn từ khi John von Neumann là người đầu tiên hình thức hóa nó trong thời kỳ trước và trong Chiến tranh Lạnh, chủ yếu do áp dụng của nó trong chiến lược quân sự, nổi tiếng nhất là khái niệm đảm bảo phá hủy lẫn nhau (mutual assured destruction). Bắt đầu từ những năm 1970, Lý thuyết trò chơi bắt đầu được áp dụng cho nghiên cứu về hành vi động vật, trong đó có sự phát triển của các loài qua chọn lọc tự nhiên. Do các trò chơi hay như Song đề tù nhân(prisoner's dilemma), trong đó lợi ích cá nhân làm hại cho tất cả mọi người, Lý thuyết trò chơi đã bắt đầu được dùng trong Chính trị học, Đạo đức họcvà triết học. Cuối cùng, Lý thuyết trò chơi gần đây đã thu hút được sự chú ý của các nhà Khoa học máy tính do ứng dụng của nó trong Trí tuệ nhân tạo và Điều khiển học. |
| google | MATLAB là phần mềm cung cấp môi trường tính toán số và lập trình, do công ty MathWorksthiết kế. MATLAB cho phép tính toán số với ma trận, vẽ đồ thị hàm số hay biểu đồ thông tin, thực hiện thuật toán, tạo các giao diện người dùng và liên kết với những chương trình máy tính viết trên nhiều ngôn ngữ lập trình khác |
| matlab | MATLAB là phần mềm cung cấp môi trường tính toán số và lập trình, do công ty MathWorksthiết kế. MATLAB cho phép tính toán số với ma trận, vẽ đồ thị hàm số hay biểu đồ thông tin, thực hiện thuật toán, tạo các giao diện người dùng và liên kết với những chương trình máy tính viết trên nhiều ngôn ngữ lập trình khác |
| microsoft | Microsoft Windows (hoặc đơn giản là Windows) là tên của một họ hệ điều hành dựa trên giao diện người dùng đồ hoạ được phát triển và được phân phối bởi Microsoft. Nó bao gồm một vài các dòng hệ điều hành, mỗi trong số đó phục vụ một phần nhất định của ngành công nghiệp máy tính. Các dòng Windows hiện tại gồm Windows NT, Windows Embedded và Windows Phone; chúng có thể bao gồm các phân họ, VD: Windows Embedded Compact (Windows CE) hoặc Windows Server. Các dòng Windows đã bị ngừng gồm Windows 9x và Windows Mobile. |
| windows | Microsoft Windows (hoặc đơn giản là Windows) là tên của một họ hệ điều hành dựa trên giao diện người dùng đồ hoạ được phát triển và được phân phối bởi Microsoft. Nó bao gồm một vài các dòng hệ điều hành, mỗi trong số đó phục vụ một phần nhất định của ngành công nghiệp máy tính. Các dòng Windows hiện tại gồm Windows NT, Windows Embedded và Windows Phone; chúng có thể bao gồm các phân họ, VD: Windows Embedded Compact (Windows CE) hoặc Windows Server. Các dòng Windows đã bị ngừng gồm Windows 9x và Windows Mobile. |
| train | Một tập huấn luyện được sử dụng trong trí tuệ nhân tạo, cùng với phương pháp học có giám sát, và nó bao gồm một vec-tơ đầu vào và một vec-tơ kết quả. |
| ngẫu\_nhiên | Ngẫu nhiên nghĩa là thiếu một khuôn mẫu hay khả năng dự báo trong các sự kiện.[1] Sự ngẫu nhiên cho thấy một sự vô thứ tự hoặc không gắn kết trong một chuỗi các ký hiệu hoặc bước, nấc; sao cho không có một khuôn mẫu hoặc sự kết hợp nào minh bạch dễ hiểu. |
| kernel | Nhân hệ điều hành (tiếng Anh: Kernel) là thành phần trung tâm của hầu hết các hệ điều hành máy tính. Nó có nhiệm vụ quản lý các tài nguyên hệ thống (liên lạc giữa các thành phần phần cứng và phần mềm).[1] Thông thường, với vai trò một thành phần cơ bản của một hệ điều hành, nhân có thể cung cấp các tầng trừu tượngmức thấp nhất cho các tài nguyên máy tính (đặc biệt là bộ nhớ, CPU, và các thiết bị vào ra mà phần mềm ứng dụng cần điều khiển để thực hiện các chức năng của mình. Nhân hệ điều hành thường cung cấp các tiện ích xử lý này cho các tiến trình của các phần mềm ứng dụng qua các cơ chế liên lạc giữa các tiến trình (inter-process communication) và các hàm hệ thống (system call). |
| offline | Offline có nghĩa là ẩn, ngoại tuyến. Nghĩa là tình trạng chưa hoặc không kết nối với mạng Internet (chế độ tắt mạng), trái nghĩa với offline chính là online (trực tuyến). |
| online | Online chỉ có 1 nghĩa duy nhất đó là trực tuyến, tức là bạn đang kết nối với mạng Internet. Trái nghĩa với Offline. Để hiểu rõ hơn thì bạn hãy đọc kỹ lại phần ví dụ của Offline, bạn sẽ thấy Online và Offline có mối quan hệ mật thiết với nhau. |
| luật | Phương pháp biểu diễn tri thức bằng luật sinh được phát minh bởi Newell và Simon trong lúc hai ông đang cố gắng xây dựng một hệ giải bài toán tổng quát. Đây là một kiểu biểu diễn tri thức có cấu trúc. Ý tưởng cơ bản là tri thức có thể được cấu trúc bằng một cặp điều kiện – hành động : "NẾU điều kiện xảy ra THÌ hành động sẽ được thi hành". Chẳng hạn : NẾU đèn giao thông là đỏ THÌ bạn không được đi thẳng, NẾU máy tính đã mở mà không khởi động được THÌ kiểm tra nguồn điện, … |
| bellman | Richard Ernest Bellman là một nhà toán học ứng dụng người Mỹ, được ghi nhớ vì phát minh ra quy hoạch động vào năm 1953, và nhiều đóng góp quan trọng trong nhiều lĩnh vực toán học khác. |
| richard\_bellman | Richard Ernest Bellman là một nhà toán học ứng dụng người Mỹ, được ghi nhớ vì phát minh ra quy hoạch động vào năm 1953, và nhiều đóng góp quan trọng trong nhiều lĩnh vực toán học khác. |
| quick\_sort | Sắp xếp nhanh (Quicksort), còn được gọi là sắp xếp kiểu phân chia (part sort) là một thuật toán sắp xếp phát triển bởi C.A.R. Hoare |
| quicksort | Sắp xếp nhanh (Quicksort), còn được gọi là sắp xếp kiểu phân chia (part sort) là một thuật toán sắp xếp phát triển bởi C.A.R. Hoare |
| heap\_sort | Sắp xếp vun đống (Heapsort) là thuật toán sắp xếp dựa trên một cấu trúc dữ liệu được gọi là đống nhị phân (binary heap), gọi đơn giản là đống. |
| heapsort | Sắp xếp vun đống (Heapsort) là thuật toán sắp xếp dựa trên một cấu trúc dữ liệu được gọi là đống nhị phân (binary heap), gọi đơn giản là đống. |
| suy\_diễn\_lùi | Suy diễn lùi là quá trình xuất phát từ sự kiện cần chứng minh và thay vào đó là những sự kiện ở vế trái của 1 luật có vế phải là sự kiện cần chứng minh. Quá trình này được thực hiện cho đến khi đưa về các sự kiện là tập sự kiện con của tập sự kiện giả thiết. |
| suy\_diễn\_tiến | Suy diễn tiến là quá trình suy diễn bắt đầu từ tập sự kiện đã biết, rút ra những sự kiện mới và cứ như vậy cho đến khi có được sự kiện cần chứng minh hoặc không có luật nào sinh ra các sự kiện mới (tập sự kiện đúng là cực đại). |
| algorithm | Thuật toán (tiếng Anh: Algorithm) , còn gọi là giải thuật, là một tập hợp hữu hạn của các chỉ thị hay phương cách được định nghĩa rõ ràng cho việc hoàn tất một số sự việc từ một trạng thái ban đầu cho trước; khi các chỉ thị này được áp dụng triệt để thì sẽ dẫn đến kết quả sau cùng như đã dự đoán trước. Nói cách khác, thuật toán là một bộ các quy tắc hay quy trình cụ thể nhằm giải quyết một vấn đề trong một số bước hữu hạn, hoặc nhằm cung cấp một kết quả từ một tập hợp của các dữ kiện đưa vào. |
| algorithms | Thuật toán (tiếng Anh: Algorithm) , còn gọi là giải thuật, là một tập hợp hữu hạn của các chỉ thị hay phương cách được định nghĩa rõ ràng cho việc hoàn tất một số sự việc từ một trạng thái ban đầu cho trước; khi các chỉ thị này được áp dụng triệt để thì sẽ dẫn đến kết quả sau cùng như đã dự đoán trước. Nói cách khác, thuật toán là một bộ các quy tắc hay quy trình cụ thể nhằm giải quyết một vấn đề trong một số bước hữu hạn, hoặc nhằm cung cấp một kết quả từ một tập hợp của các dữ kiện đưa vào. |
| giải\_thuật | Thuật toán (tiếng Anh: Algorithm) , còn gọi là giải thuật, là một tập hợp hữu hạn của các chỉ thị hay phương cách được định nghĩa rõ ràng cho việc hoàn tất một số sự việc từ một trạng thái ban đầu cho trước; khi các chỉ thị này được áp dụng triệt để thì sẽ dẫn đến kết quả sau cùng như đã dự đoán trước. Nói cách khác, thuật toán là một bộ các quy tắc hay quy trình cụ thể nhằm giải quyết một vấn đề trong một số bước hữu hạn, hoặc nhằm cung cấp một kết quả từ một tập hợp của các dữ kiện đưa vào. |
| thuật\_giải | Thuật toán (tiếng Anh: Algorithm) , còn gọi là giải thuật, là một tập hợp hữu hạn của các chỉ thị hay phương cách được định nghĩa rõ ràng cho việc hoàn tất một số sự việc từ một trạng thái ban đầu cho trước; khi các chỉ thị này được áp dụng triệt để thì sẽ dẫn đến kết quả sau cùng như đã dự đoán trước. Nói cách khác, thuật toán là một bộ các quy tắc hay quy trình cụ thể nhằm giải quyết một vấn đề trong một số bước hữu hạn, hoặc nhằm cung cấp một kết quả từ một tập hợp của các dữ kiện đưa vào. |
| thuật\_toán | Thuật toán (tiếng Anh: Algorithm) , còn gọi là giải thuật, là một tập hợp hữu hạn của các chỉ thị hay phương cách được định nghĩa rõ ràng cho việc hoàn tất một số sự việc từ một trạng thái ban đầu cho trước; khi các chỉ thị này được áp dụng triệt để thì sẽ dẫn đến kết quả sau cùng như đã dự đoán trước. Nói cách khác, thuật toán là một bộ các quy tắc hay quy trình cụ thể nhằm giải quyết một vấn đề trong một số bước hữu hạn, hoặc nhằm cung cấp một kết quả từ một tập hợp của các dữ kiện đưa vào. |
| dijsktra | Thuật toán Dijkstra, mang tên của nhà khoa học máy tính người Hà Lan Edsger Dijkstra vào năm 1956 và ấn bản năm 1959[1], là một thuật toán giải quyết bài toán đường đi ngắn nhất nguồn đơn trong một đồ thị có hướng không có cạnh mang trọng số âm. Thuật toán thường được sử dụng trong định tuyến với một chương trình con trong các thuật toán đồ thị hay trong công nghệ Hệ thống định vị toàn cầu (GPS). |
| overflow | Thuộc tính overflow xác định điều gì sẽ xảy ra nếu một thành phần box tràn nội dung. |
| thông\_minh | Trí thông minh hay trí năng được định nghĩa theo nhiều cách khác nhau bao gồm khả năng logic, trừu tượng, sự hiểu biết, tự nhận thức, học tập, có trí tuệ xúc cảm, trí nhớ, kế hoạch, và giải quyết vấn đề. Trí thông minh được nghiên cứu rộng rãi ở loài người, nhưng cũng được quan sát ở động vậtvà thực vật. Trí tuệ nhân tạo là sự mô phỏng trí thông minh ở máy móc. |
| compiler | Trình biên dịch, còn gọi là phần mềm biên dịch, compiler, là một chương trình máy tính làm công việc dịch một chuỗi các câu lệnh được viết bằng một ngôn ngữ lập trình (gọi là ngôn ngữ nguồn hay mã nguồn), thành một chương trình tương đương nhưng ở dưới dạng một ngôn ngữ máy tính mới (gọi là ngôn ngữ đích) và thường là ngôn ngữ ở cấp thấp hơn, như ngôn ngữ máy. Chương trình mới được dịch này gọi mã đối tượng. |
| tối\_ưu\_hóa | Trong khoa học máy tính và toán học, bài toán tối ưu hóa là bài toán tìm kiếm lời giải tốt nhất trong tất cả các lời giải khả thi. Bài toán tối ưu hóa có thể được chia thành hai loại tùy thuộc vào việc các biến là liên tục hay rời rạc. |
| cấu\_trúc\_dữ\_liệu | Trong khoa học máy tính, cấu trúc dữ liệu là một cách lưu dữ liệu trong máy tính sao cho nó có thể được sử dụng một cách hiệu quả. Trong thiết kế nhiều loại chương trình, việc chọn cấu trúc dữ liệu là vấn đề quan trọng. |
| ngăn\_xếp | Trong khoa học máy tính, một ngăn xếp (còn gọi là bộ xếp chồng, tiếng Anh: stack) là một cấu trúc dữ liệu trừu tượng hoạt động theo nguyên lý "vào sau ra trước" (Last In First Out (LIFO). |
| liên\_thông | Trong lý thuyết đồ thị, một thành phần liên thông của một đồ thị vô hướng là một đồ thị con trong đó giữa bất kì hai đỉnh nào đều có đường đi đến nhau, và không thể nhận thêm bất kì một đỉnh nào mà vẫn duy trì tính chất trên. |
| quy\_hoạch\_động | Trong ngành khoa học máy tính, quy hoạch động là một phương pháp giảm thời gian chạy của các thuật toán thể hiện các tính chất của các bài toán con gối nhau (overlapping subproblem) và cấu trúc con tối ưu (optimal substructure). |
| phân\_hoạch | Trong số học, sự phân tích một số nguyên dương n là cách viết số đó dưới dạng tổng của các số nguyên dương. Hai cách phân tích có các số hạng giống nhau được coi là một cách phân tích. Số lượng cách phân tích số n được tính bởi hàm phân tích, ký hiệu là p(n). |
| tiền\_tố | Trong toán học thì các biểu thức thường được biểu diễn dưới dạng trung tố (các toán tử nằm giữa các toán hạng) cho dễ hiểu, đó là đối với con người. Còn đối với máy tính thì nó khó hiểu đối với chúng. Vì vậy, để hiểu được máy tính phải đưa biểu thức dạng trung tố về dạng tiền tố hoặc hậu tố. Trong bài viết này tôi trình bày cách đưa biểu thức dạng trung tố về dạng tiền tố và tính toán biểu thức tiền tố. |
| hằng\_số | Trong vật lý và toán học, hằng số là đại lượng có giá trị không đổi. Hằng số thường được ký hiệu là const, viết tắt chữ tiếng Anh constant. |
| euler | Trong lý thuyết đồ thị, một đường đi trong đồ thị G=(X,E) được gọi là đường đi Euler nếu nó đi qua tất cả các cạnh của đồ thị, mỗi cạnh đúng một lần. Đường đi Euler có đỉnh cuối cùng trùng với đỉnh xuất phát gọi là chu trình Euler. Khái niệm chu trình Euler xuất phát từ bài toán bảy cây cầu do Euler giải quyết vào khoảng năm 1737. Đường đi Euler có thể tìm thấy trong các bài toán vui vẽ một nét (vẽ một hình nào đó mà không nhấc bút khỏi mặt giấy, không tô lại cạnh nào hai lần). |
| trung\_vị | Trong lý thuyết xác suất và thống kê, số trung vị (tiếng Anh: median) là một số tách giữa nửa lớn hơn và nửa bé hơn của một mẫu, một quần thể, hay một phân bố xác suất. Nó là giá trị giữa trong một phân bố, mà số các số nằm trên hay dưới con số đó là bằng nhau. Điều đó có nghĩa rằng 1/2 quần thể sẽ có các giá trị nhỏ hơn hay bằng số trung vị, và một nửa quần thể sẽ có giá trị bằng hoặc lớn hơn số trung vị. |
| lý\_thuyết\_đồ\_thị | Trong toán học và tin học, lý thuyết đồ thị nghiên cứu các tính chất của đồ thị. Một cách không chính thức, đồ thị là một tập các đối tượng được gọi là các đỉnh (hoặc nút) nối với nhau bởi các cạnh (hoặc cung). Cạnh có thể có hướng hoặc vô hướng. Đồ thị thường được vẽ dưới dạng một tập các điểm (các đỉnh nối với nhau bằng các đoạn thẳng (các cạnh). Đồ thị biểu diễn được rất nhiều cấu trúc, nhiều bài toán thực tế có thể được biểu diễn bằng đồ thị. Ví dụ, cấu trúc liên kết của một website có thể được biểu diễn bằng một đồ thị có hướng như sau: các đỉnh là các trang web hiện có tại website, tồn tại một cạnh có hướng nối từ trang A tới trang B khi và chỉ khi A có chứa 1 liên kết tới B. Do vậy, sự phát triển của các thuật toán xử lý đồ thị là một trong các mối quan tâm chính của khoa học máy tính. Cấu trúc đồ thị có thể được mở rộng bằng cách gán trọng số cho mỗi cạnh. Có thể sử dụng đồ thị có trọng số để biểu diễn nhiều khái niệm khác nhau. Ví dụ, nếu đồ thị biểu diễn một mạng đường giao thông, các trọng số có thể là độ dài của mỗi con đường. Một cách khác để mở rộng đồ thị cơ bản là quy định hướng cho các cạnh của đồ thị (như đối với các trang web, A liên kết tới B, nhưng B không nhất thiết cũng liên kết tới A). Loại đồ thị này được gọi là đồ thị có hướng. Một đồ thị có hướng với các cạnh có trọng số được gọi là một lưới. Các lưới có nhiều ứng dụng trong khía cạnh thực tiễn của lý thuyết đồ thị, chẳng hạn, phân tích lưới có thể dùng để mô hình hoá và phân tích mạng lưới giao thông hoặc nhằm "phát hiện" hình dáng của Internet - (Xem thêm các ứng dụng đưới đây. Mặc dù vậy, cũng nên lưu ý rằng trong phân tích lưới, thì định nghĩa của khái niệm "lưới" có thể khác nhau và thường được chỉ ra bằng một đồ thị đơn giản.) |
| operator | Trong toán học, một toán tử (tiếng Anh operator, phân biệt với operation - phép toán) là một hàm, thông thường có một vai trò quan trọng trong một lĩnh vực nào đấy. Chẳng hạn trong đại số tuyến tính có "toán tử tuyến tính" (linear operator). Trong giải tích có "toán tử vi phân" (differential operator)... Thông thường, một "toán tử" là một hàm tác động lên các hàm khác; hoặc nó có thể là tổng quát hóa của một hàm, như trong đại số tuyến tính. |
| toán\_tử | Trong toán học, một toán tử (tiếng Anh operator, phân biệt với operation - phép toán) là một hàm, thông thường có một vai trò quan trọng trong một lĩnh vực nào đấy. Chẳng hạn trong đại số tuyến tính có "toán tử tuyến tính" (linear operator). Trong giải tích có "toán tử vi phân" (differential operator)... Thông thường, một "toán tử" là một hàm tác động lên các hàm khác; hoặc nó có thể là tổng quát hóa của một hàm, như trong đại số tuyến tính. |
| ánh\_xạ | Trong toán học, ánh xạ là khái quát của khái niệm hàm số. hàm số lại xuất phát từ khái niệm tương quan giữa các đại lượng vật lý. Chẳng hạn trong một chuyển động đều, độ dài quãng đường đi được bằng tích của tốc độ với thời gian. Nếu tốc độ là 5cm/s thì quãng đường đi được trong t giây là s= 5t |
| không\_gian\_vector | Trong toán học, không gian vectơ là một tập hợp mà trên đó hai phép toán, phép cộng vectơ và phép nhân vectơ với một số, được định nghĩa và thỏa mãn các tiên đề được liệt kê dưới đây. |
| truy\_hồi | Truy hồi thông tin (hay còn gọi là truy vấn thông tin) là hoạt động thu thập các nguồn thông tin liên quan đến một thông tin cần tìm kiếm, có thể dựa trên siêu dữ liệu (metadata) và trên việc đánh chỉ mục toàn văn (hoặc dựa trên nội dung khác). Những hệ thống truy hồi thông tin tự động được sử dụng để giảm thiểu "quá tải thông tin". Nhiều các trường đại học và thư viện công cộng sử dụng hệ thống truy hồi thông tin để cung cấp truy cập đến sách báo và các tài liệu khác. Các máy tìm kiếm trên web được coi là những ứng dụng truy hồi thông tin dễ nhận thấy nhất. |
| tuple | Tuple là một container cũng được sử dụng rất nhiều trong các chương trình Python không thua kém gì List. Một Tuple gồm các yếu tố sau: Được giới hạn bởi cặp ngoặc (), tất cả những gì nằm trong đó là những phần tử của Tuple; Các phần tử của Tuple được phân cách nhau ra bởi dấu phẩy ; Tuple có khả năng chứa mọi giá trị, đối tượng trong Python. |
| rời\_rạc | Ứng dụng phổ biến của rời rạc hóa là đưa các vùng dữ liệu lớn về vùng dữ liệu nhỏ để dễ xử lý, sao cho vẫn thỏa mãn yêu cầu của bài toán đặt ra. |
| lỗi\_cú\_pháp | Ứng với mỗi ngôn ngữ lập trình, sẽ có bộ cú pháp riêng. Nhưng dù có sử dụng ngôn ngữ nào thì lập trình viên vẫn luôn luôn phải đối mặt với lỗi sai phổ biến nhất là lỗi cú pháp. |
| biên\_toàn\_cục | Vị trí biến đặt bên ngoài tất cả các hàm, cấu trúc…Các biến có ảnh hưởng tới toàn bộ chương trình |
| hậu\_tố | Việc tính giá trị của một biểu thức toán học ở dạng trung tố trong máy tính thông thường sẽ được chuyển sang dạng ký pháp nghịch đảo Ba Lan (hậu tố) để việc tính toán được dễ dàng. Bạn có thể xem lại thuật toán chuyển đổi từ trung tố sang hậu tố trong bài viết “Chuyển biểu thức trung tố sang tiền tố và hậu tố bằng Stack” của tôi, và tải project mẫu tại đây. Trong bài viết này, tôi sẽ trình bày phương pháp tính giá trị của một biểu thức tiền tố và hậu tố bằng Stack. Bạn có thể sẽ cần áp dụng kiến thức dưới đây nếu muốn tạo ra một cây biểu thức (expression tree) từ các dạng biểu thức này. |
| protocol | Việc trao đổi thông tin dù là đơn giản nhất cũng phải tuân theo những nguyên tắc nhất định. Đơn giản như hai người nói chuyện với nhau, muốn cho cuộc nói chuyện có kết quả thì ít nhất cả hai người phải ngầm tuân thủ quy ước : Khi một người nói thì người kia phải biết lắng nghe và ngược lại. Việc truyền thông trên mạng cũng vậy. Cần có các quy tắc, quy ước truyền thông về nhiều mặt : khuôn dạng cú pháp của dữ liệu, các thủ tục gửi, nhận dữ liệu, kiểm soát hiệu quả nhất chất lượng truyền thông tin. Tập hợp những quy tắc, quy ước truyền thông đó được gọi là giao thức của mạng (protocol). |
| lặp | Vòng lặp là một loại câu lệnh điều khiển dòng xử lý của máy tính. Vòng lặp for là để thực hiện lặp đi lặp lại một tập hợp các câu lệnh nào đó với điều kiện cụ thể. |
| natural\_language\_processing | Xử lý ngôn ngữ tự nhiên (natural language processing - NLP) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ-công cụ hoàn hảo nhất của tư duy và giao tiếp. |
| nlp | Xử lý ngôn ngữ tự nhiên (natural language processing - NLP) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ-công cụ hoàn hảo nhất của tư duy và giao tiếp. |
| classification | Ý tưởng của bài toán phân loại bắt đầu như sau: Chúng ta có một danh sách các quan sát (observation). Một quan sát là một khái niệm chung chung. Với Machine Learning, một quan sát thường là sự mô tả một đối tượng trong một trạng thái nào đó. Để hiểu rõ hơn, hãy xét một ví dụ rất phổ biến trong Machine Learning: hoa diên vĩ (Iris flowers). Giả sử một quan sát là sự mô tả một bông hoa. Với mỗi bông hoa, chúng ta có bốn thông số: độ dày và chiều dài của đài hoa (sepal) và cánh hoa (petal). Trên thực tế, ví dụ về hoa diên vĩ nổi tiếng với cái tên the Iris dataset và đây là bộ dữ liệu được sử dụng rất rộng rãi trong giới nghiên cứu Machine Learning để tìm hiểu về các thuật toán phân loại. |
| sự\_kiện |  |
| đầu\_vào |  |
| công\_cụ |  |
| framework |  |
| Admin |  |
| Ai |  |
| AI |  |
| artificial\_intelligence |  |
| backtracking |  |
| bảng\_băm |  |
| Bayes |  |
| biên\_dịch |  |
| biến\_số |  |
| binary\_search |  |
| Bit |  |
| Bít |  |
| Bool |  |
| Boolean |  |
| cấp\_phát |  |
| cấu\_trúc |  |
| chiến\_lược |  |
| Class |  |
| complexity |  |
| Ctdl |  |
| cú\_pháp |  |
| đảo\_ngược |  |
| đáp\_án |  |
| dãy\_fibonacci |  |
| đệ\_qui |  |
| đệ\_quy |  |
| debug |  |
| độ\_phức\_tạp |  |
| dữ\_liệu |  |
| đường\_thẳng |  |
| function |  |
| giải\_phóng |  |
| giao\_diện |  |
| giao\_thức |  |
| graph |  |
| hàm\_băm |  |
| hiệu\_năng |  |
| hiệu\_quả |  |
| hoán\_vị |  |
| huấn\_luyện |  |
| input |  |
| insert |  |
| kế\_thừa |  |
| khai\_báo |  |
| kĩ\_thuật |  |
| kích\_thước |  |
| lập\_trình\_viên |  |
| leaf |  |
| lời\_giải |  |
| loop |  |
| lớp |  |
| mã\_hóa |  |
| matrix |  |
| mâu\_thuẫn |  |
| neighbor |  |
| nhúng |  |
| node |  |
| nút\_gốc |  |
| Object\_oriented\_programming |  |
| oop |  |
| optimize |  |
| parallel |  |
| phần\_tử |  |
| postfix |  |
| prefix |  |
| query |  |
| random |  |
| randomized |  |
| recursion |  |
| rời\_rạc\_hóa |  |
| root |  |
| sắp\_xếp |  |
| search |  |
| số\_nguyên |  |
| song\_song |  |
| sort |  |
| string |  |
| swap |  |
| tài\_liệu |  |
| tham\_số |  |
| theorem\_graphic |  |
| theory |  |
| thiết\_kế |  |
| thuật\_ngữ |  |
| tìm\_kiếm |  |
| tìm\_kiếm\_nhị\_phân |  |
| toán\_học |  |
| trạng\_thái |  |
| tree |  |
| trí\_tuệ\_nhân\_tạo |  |
| trình\_biên\_dịch |  |
| trình\_duyệt |  |
| truy\_nhập |  |
| turing |  |
| var |  |
| variance |  |
| vét\_cạn |  |
| xác\_suất |  |
| xóa\_đỉnh |  |

# CHƯƠNG IV: THIẾT KẾ BỘ TÌM KIẾM

Để thiết kế bộ tìm kiếm, nhóm thực hiện các bước sau:

1. Xây dựng bộ thư viện để tiện cho việc xử lý ngôn ngữ tiếng Việt
2. Xây dựng bộ máy tìm kiếm

Bộ thư viện để tiện cho việc xử lý ngôn ngữ tiếng Việt

Gồm có:

- Thư viện tách từ: vnTokenizer

- Thư viện gán nhãn từ loại: vnTagger

- Thư viện phân tích cú pháp: Earley Parser

## Thư viện tách từ vnTokenizer

Nhóm xây dựng thư viện này ban đầu dựa trên bài báo [1], sau khi cài đặt xong và chạy thử trên bộ dữ liệu nhỏ thì kết quả chưa tốt. Nhưng chạy trên bộ dữ liệu lớn hơn thì không có thời gian và máy móc đủ mạnh. Vậy nên, nhóm đành phải cân nhắc các thuật toán cổ điển hơn như: Maximum Matching (MM) và Weighted Finite State Tranducer (WFST). Tuy nhiên, cả hai phương pháp này đều không dựa trên ngữ cảnh xung quanh của từ trong câu đang xét (MM thì chọn lựa dựa trên bộ từ điển, WFST thì lựa chọn dựa trên xác suất các từ trên bộ dữ liệu mẫu). Nên, việc cân nhắc lựa chọn giữa thuật toán này là như nhau. Cuối cùng, nhóm quyết định chọn phương pháp MM vì nó có ưu điểm là nhanh và đơn giản trong cài đặt hơn. Trên mạng đã có nhiều tool được share public để làm việc này tốt hơn. Việc cài đặt lại thư viện này có vẽ như không được khuyến khích, tuy nhiên, vì mục tiêu học hỏi nên nhóm đã chọn cách làm này.

## Thư viện gán nhãn từ loại vnTagger

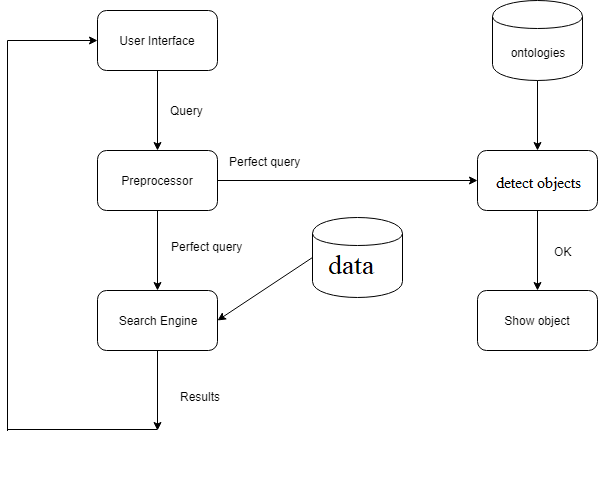
Nhóm xây dựng thư viện này dựa trên phương pháp học máy: Transformation Based Learning (TBL). Nhóm chọn phương pháp này vì: phương pháp này không phải là quá khó trong việc cài đặt (nếu không muốn nói là đơn giản) so với những phương pháp khác, tuy nhiên mang lại hiệu quả cao. Thứ cần thiết để phương pháp này trở nên tốt tương đương với độ tốt của bộ dữ liệu. Dữ liệu để xây dựng thư viện này, nhóm thu thập ở trên mạng, chủ yếu là trang [2]. Cũng như vnTokenizer, việc cài đặt lại thư viện này chỉ để mục đích học hỏi, vì trên mạng đã có tool làm việc này tốt hơn.

## Thư viện phân tích cú pháp Earley Parser

Nhóm xây dựng thư viện này dựa trên phương pháp Earley Parsing. Lý do nhóm chọn phương pháp này là vì nó đơn giản, dễ cài đặt nhưng lại hiệu quả với tốc độ và có thể tránh được đệ quy trái (nhập nhằn trong xử lý ngôn ngữ tự nhiên). Tuy nhiên, độ chính xác vẫn chưa cao vì nhóm chưa làm được phần đánh giá độ hiệu quả của hai cây cú pháp có thể phân tích được từ 1 câu. Và cũng vì chưa có thời gian để xây dựng được bộ ngữ pháp đẹp hơn, nên có nhiều câu trong tài liệu cũng không thể phân tích được.

## Xây dựng bộ máy tìm kiếm

### Tổng quan về bộ máy tìm kiếm



Hình 5: Tổng quan bộ máy tìm kiếm

User Interface: Là phần giao diện

Preprocessor: Là bộ tiền xử lý

Search Engine: Nơi đây cho phép truy xuất các tài liệu liên quan và trả về cho người dùng

Để có thể xây dựng được bộ máy tìm kiếm như trên, phải giải quyết khá nhiều bài toán con. Dưới đây là các bài toán con trọng yếu:

* + - 1. Tiền xử lý dữ liệu
      2. Chấm điểm và lựa chọn tài liệu
      3. Xếp hạng các tài liệu

Dưới đây, nhóm xin trình bày lần lượt các phần này. Chi tiết cài đặt sẽ được đề cập vào phần sau nữa.

## Tiền xử lý dữ liệu

Dữ liệu sau khi crawl từ các trang web về đa số không được định dạng đẹp, và tiện. Điều này làm ảnh hưởng đến một dãy các quá trình tính toán về sau. Vì vậy nên, nhiệm vụ của bước này là: Nhận vào một tài liệu thô (vừa crawl về từ trang web), thực hiện có lựa chọn các chức năng sau:

* Xóa kí tự đặc biệt
* Tách từ cho tài liệu
* Xóa các từ stopwords
* Tách câu cho tài liệu

## Chấm điểm và lựa chọn tài liệu

Sau khi nhận một câu query, và tiền xử lý nó, thì việc áp dụng vào tìm kiếm có lẽ sẽ không tốt bằng việc ta thêm một bước ở đây nữa là: *Kiểm lỗi chính tả* của câu query.

Sau đó, theo phương pháp Vector Space Model, ta cũng chưa thể dùng câu query đó ngay được mà phải: Tiến hành mô hình hóa câu query ấy thành vector. Tuy nhiên, sau khi mô hình hóa vector xong, việc áp dụng vào tìm kiếm có lẽ vẫn chưa tốt nếu ta thực hiện thêm một chức năng nhỏ nữa. Đó là: Thêm các từ đồng nghĩa, gần nghĩa vào vector vừa xây dựng. Nhưng, giá trị tf-idf của chúng sẽ là bao nhiêu? Theo đề xuất của nhóm, các số này được lấy từ giá trị tf-idf của từ gốc và nhân với một giá trị phán đoán.

Vậy là đã xong, tiếp theo, tiến hành đi tính các giá trị độ tương đồng của vector của câu query này với vector của các tài liệu khác. Việc tính giá trị này dựa trên cây Trie nên độ phức tạp khá ổn (tính giá trị cosine trên 200 tài liệu trong vòng sấp xỉ 0,002s). Sau khi tính điểm xong, cần lựa chọn số ít các tài liệu có liên quan. Để làm việc này, nhóm đề xuất cách làm là: sắp xếp giảm dần độ lớn của giá trị cosine, sau đó chọn số nguyên dương k tài liệu có độ lớn cao nhất. Cụ thể, nhóm chọn k = 10.

Cuối cùng, theo những nhận định trên, các bài toán con cần giải quyết trong đây sẽ là:

* Kiểm lỗi chính tả
* Mô hình hóa câu query thành vector
* Mở rộng vector query
* Tính giá trị cosine của vector biểu diễn query và vector biểu diễn tài liệu

## Xếp hạng các tài liệu

Sau khi đã có được số ít các tài liệu liên quan nhất (theo cosine) ở bước *chấm điểm và lựa chọn tài liệu*. Nếu ta trả về luôn kết quả này cho người dùng, thì có thể thứ tự sắp xếp các tài liệu liên quan sẽ không được người dùng đánh giá cao. Vì, người dùng muốn, tài liệu liên quan nhất phải được xếp đầu tiên. Và cứ thế…

Nhóm đề xuất các làm sau để xếp hạng các tài liệu này:

Thiết kế một hàm Heuristic cho phép tính độ tương đồng chung của hai chuỗi bất kì. Áp dụng hàm này trong việc tính độ tương đồng giữa:

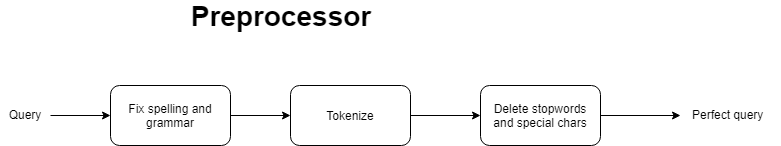
* Query và tiêu đề của tài liệu
* Query và nội dung của tài liệu (sau khi đã được gán nhãn từ loại)

Tiếp theo, để tăng phần chính xác, nhóm đề xuất tiếp việc xử dụng cây cú pháp. Cụ thể, nhóm sẽ thiết kế thuật toán so khớp giữa hai cây cú pháp, thuật toán này sẽ trả về một số thực (nằm trong khoảng [0, 1]) là độ tương quan cú pháp giữa 2 cây cú pháp. Áp dụng thuật toán này trên: Cây cú pháp được xây dựng từ câu query và các cây cú pháp được xây dựng dựa trên các câu trong các tài liệu (đây cũng là lí do mà vì sao lại có chức năng tách câu trong bước *tiền xử lý dữ liệu*). Chi tiết thuật toán sẽ được mô tả trong phần tiếp theo.

# ỨNG DỤNG – HỆ THỐNG TÌM KIẾM THUẬT TOÁN

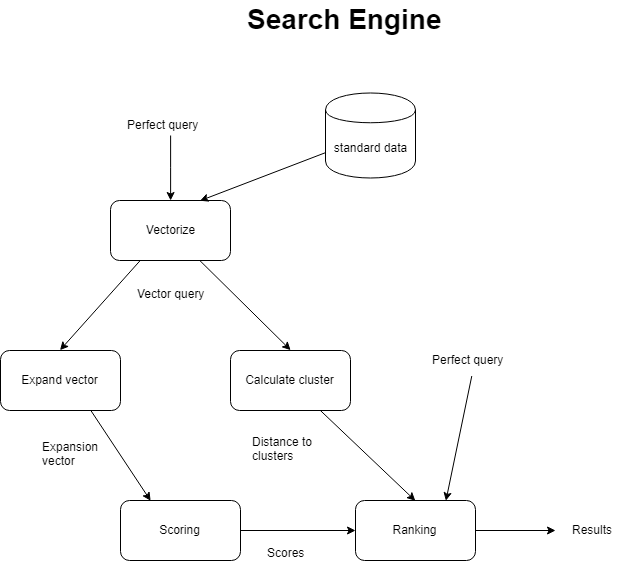
Trong ứng dụng này, nhóm thiết kế hệ thống như Hình 5.

Bộ tiền xử lý được mô tả như sau:



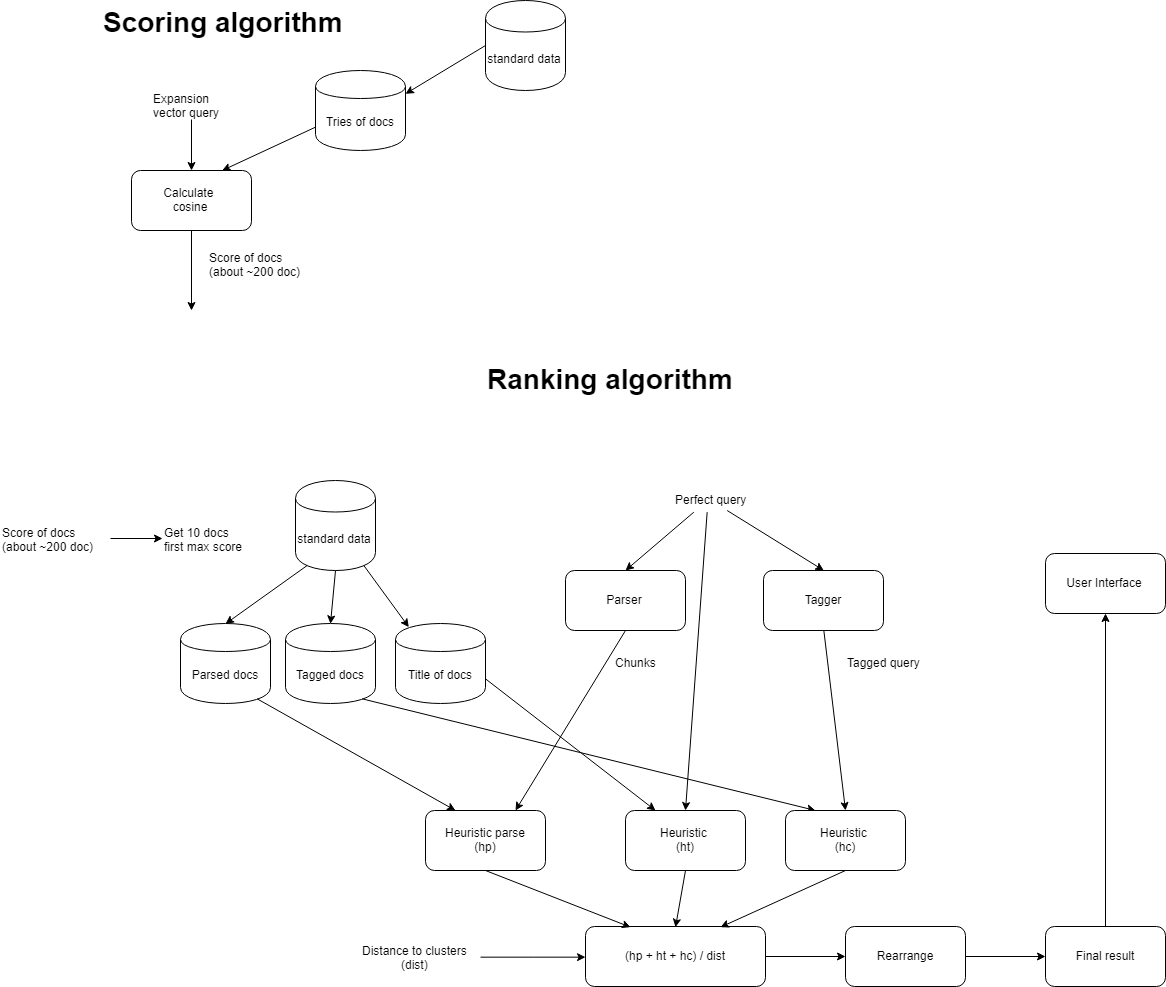
Hình 6: Mô tả luồng dữ liệu bước tiền xử lý

Bộ máy tìm kiếm được mô tả như dưới đây:



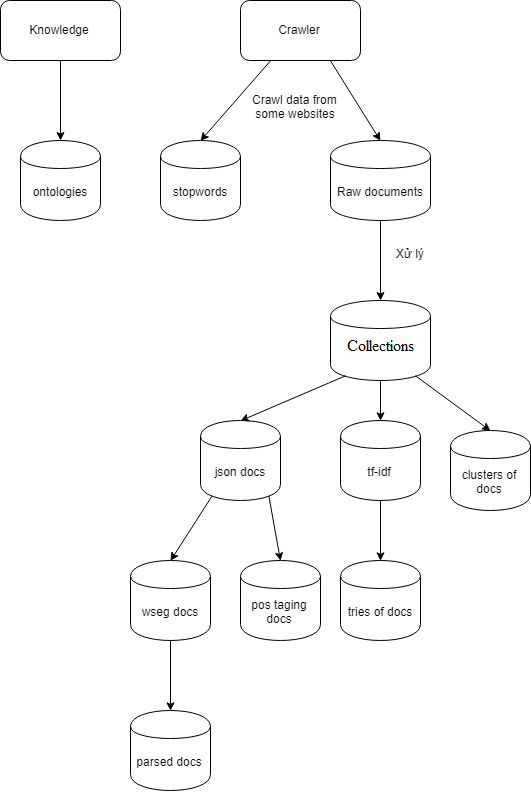
Hình 7: Mô tả luồng dữ liệu của bước tìm kiếm query

Bộ chấm điểm và xếp hạng được mô tả như sau:



Hình 8: Mô tả luồng dữ liệu của bước tính điểm và xếp hạng

Bộ dữ liệu được mô tả như sau:



Hình 9: Mô tả cách bố trí dữ liệu

Để xây dựng ứng dụng, nhóm đã phải giải quyết các vấn đề sau:

* Kiểm lỗi chính tả
* Tiền xử lý
* Mở rộng query
* So khớp
* Tính khoảng cách đến các cụm
* Xây dựng bộ máy tìm kiếm với 2 chức năng con: Chấm điểm và xếp hạng

## Bộ kiểm lỗi chính tả:

Bài toán: Đưa một câu query dạng chuỗi vào. Những từ trong query đó có thể chứa:

1. Những từ chưa có trong từ điển tiếng Việt như: noio, toi…
2. Những từ có trong từ điể–n tiếng Việt nhưng nguyên câu query nối lại thì chẳng có ý nghĩa gì cả

Làm sao để chương trình sửa đổi bổ sung câu query đó theo cho nó đúng nghĩa nhất có thể.

Mô hình hóa bài toán:

Input: Câu query (có thể bị lỗi chính tả, ngữ pháp)

Output: Câu query đã được sửa lỗi chính tả (bề mặt)

Ví dụ:

Input1: hom nay tooi ddi hocj

Output1: hôm nay tôi đi học

Thuật toán giải quyết bài toán này:

* Ý tưởng: Ta sẽ sử dụng lý thuyết về ngram và minimum edit distance
* Đầu tiên ta tách từng từ trong câu query ra
* So từng từ đó với các từ trong bộ các tài liệu, nếu từ đó chưa từng xuất hiện thì xác định đó là từ cần sửa lại cho đúng
* Tiếp theo, trong tất cả các từ có trong tài liệu ta tính minimum edit distance của từng từ với từ cần sửa và chọn ra 1 danh sách các từ có cùng khoảng cách ngắn nhất để làm các từ có khả năng thay thế.
* Cuối cùng ta dùng bigram và độ đo entropy để chọn ra từ thích hợp nhất và thay thế nó cho từ bị sai.
* Các thuật giải:
* Minimum edit distance:
  + B1: Khởi tạo:
* + 1.1. Cho w và w' là hai chuỗi cần tính khoảng cách m = len(w), n = len(w').
* + 1.2. Tạo ma trận khoảng cách d[n+1, m+1] với các phần tử có giá trị là 0
  + B2: Gán các giá trị biên:
* + 2.1. Gán d[0,i] = i với 0 <= i <=m
* + 2.2. Gán d[i,0] = i với 0 <= i <= n
  + B3: Ta dùng công thức quy hoạch động tính ma trận khoảng cách:
* + for (i = 0; i <= n; i++)
* for (j = 0; j <= m; j++)
* d[i, j] = min(d[i-1, j] + ins(w'i ), d[i-1,j-1] + rep(w'i ), d[i, j-1] + ins(wi ));
* + Trong đó,
* - ins(wi) là chi phí thêm phần tử thứ i trong chuỗi w vào chuỗi còn lại tại vị trí đang xét.
* - rep(wi) là chi phí thay thế phần tử thứ i trong chuỗi w vào chuỗi còn lại tại vị trí đang xét .
* Theo Levenshtein, ins(wi ) = 1, rep(wi ) = 2 nếu wi ≠w'i , rep(wi ) = 0 nếu wi =w'i.
  + B4: Return d[n+1,m+1]
* Can\_Change: Tìm list các từ có thể có khả năng thay thế cao nhất của 1 từ word
  + B1. Khởi tạo list kq = {}
  + B2. for w in singleword: #tìm từng từ đơn trong tài liệu
    - 2.1.refterm1[w].append(minimumdistance(toTelex(w), toTelex(word)))
    - 2.2. refterm2[w].append(minimumdistance(toVni(w), toVni(word)))
* #Hàm toTelex/toVni chuyển 1 chữ sang kiểu gõ telex/vni. Ví dụ: thuật -> thuaajt/thua65t
  + - 2.3. refterm3[w].append(minimumdistance(w,word))
    - 2.4. Min = giá trị nhỏ nhất trong 3 list refterm1, refterm2, refterm3
    - 2.5. Thêm tất các các từ có giá trị bằng min vào kq (các từ chỉ xuất hiện 1 lần)
    - 2.6. Return kq
* Bigram:
* Ta chia làm 2 phần, phần 1 sẽ xây dựng mô hình bigram và unigram có trong tài liệu, phần 2 sẽ tính entropy
* Phần 1: Xây dựng mô hình bigram và unigram của toàn tài liệu
  + B1:Khởi tạo
    - 1.1. unigram = {}
    - 1.2. bigram = {}
  + B2: Tính bigram và unigram
    - 2.1. for each line in the training\_file:
      * 2.1.1. words = line.split();
      * 2.1.2. append “</s>” to the end and “<s>” to the beginning words;
      * 2.1.3. for each i in 0 to length(words)-1:
        + 2.1.3.1. key = words[i] + ‘ ‘ + words[i+1] #Tạo key cho bigram gồm 2 từ liên tiếp nhau
        + 2.1.3.2. if key in bigram.keys() #Nếu như key đã có trong bigram

2.1.3.2.1. bigram[key] += 1

2.1.3.2.2. Ngược lại key chưa tồn tại thì, bigram[key] =1

* + - * + 2.1.3.3. if words[i] in unigram.keys()

2.1.3.3.1. unigram[words[i]] +=1

2.1.3.3.2. Ngược lại, unigram[words[i]] = 1

* Phần 2: Tính entropy của câu sentence
  + B1: Khởi tạo:
    - 1.1. confidence\_uni = 0.95
    - 1.2. confidence\_bi = 0.95
  + B2: words = sentence.split()
  + B3: for i in 0 to length(words)-1:
    - 3.1. p1 = (1-confidence\_uni)/ nterm
    - 3.2. p2 = p1\*(1-confidence\_bi)
    - 3.3. key = words[i] + ‘ ‘ + words[i+1]
    - 3.4. if words[i] in unigram.keys():
* Then p1= confidence\_uni \* (unigram[words[i]] / nterm) + (1 - confidence\_uni) / nterm
  + - 3.5. if key in bigram.keys():
* Then p2= confidence\_bi \* (bigram[key] / unigram[words[i]]) + (1 - confidence\_bi)\*p1
  + B4. return p2
* Correct: Kết hợp các thuật giải bên trên lại để sửa chính tả cho câu sentence
  + B1: sentence = ‘<s>’ + sentence
  + B2: sentence = sentence + ‘</s>’
  + B3: Khởi tạo arr\_sentence = sentence.split()
  + B4: for i in 1 to length(arr\_sentence)
    - 4.1 if arr\_sentence[i] not in singlewords #Nếu từ chưa xuất hiện thì sửa lại
      * 4.1.1: canchange = Can\_Change(arr\_sentence[i])
      * 4.1.2: xs\_term = {}
      * 4.1.3: for w in canchange:
        + 4.1.3.1: lhs = arr\_sentence[i-1] + ‘ ‘ + w
        + 4.1.3.2: rhs = w + arr\_sentence[i+1]
        + 4.1.3.3: xs\_term[w] = -log(2,Entropy(lhs)\*Entropy(rhs))
      * 4.1.4: Chọn từ có giá trị lớn nhất trong xs\_term là từ thay thế vì từ này có độ đo entropy tốt nhất
* 4.2.1.3. Ví dụ minh họa
* Minimum edit distance:
* **Ví dụ:** Khoảng cách của 2 chuỗi recieve và receive, ta sẽ có ma trận khoảng cách

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | # | r | e | c | i | e | v | e |
| # | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| r | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| e | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| c | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 |
| e | 4 | 3 | 2 | 1 | 2 | 1 | 2 | 3 |
| i | 5 | 4 | 3 | 2 | 1 | 2 | 3 | 4 |
| v | 6 | 5 | 4 | 3 | 2 | 2 | 2 | 3 |
| e | 7 | 6 | 5 | 4 | 3 | 2 | 3 | 2 |

* Can\_Change: Ví dụ cần tìm các từ thay thế cho từ hoajch
* refterm1 sẽ chọn được từ “hoạch” có khoảng cách nhỏ nhất vì minimumdistance(toTelex(hoạch), toTelex(hoajch)) = 0
* refterm2 không có từ nào tìm được khoảng cách nhỏ hơn
* refterm3 không có từ nào tìm được khoảng cách nhỏ hơn
* Vậy danh sách trả về kq={hoạch}
* Bigram:
  + Phần 1: Tạo bigram và unigram
* Ví dụ: Ta có câu cần tính là “quy hoạch động” và nterm = 1000000
* Giả sử từ “quy” xuất hiện 500 lần, “hoạch” 300 lần, “động” 200 lần, “quy hoạch” xuất hiện 150 lần và “hoạch động” xuất hiện 100 lần
* Thì ta sẽ thu được:
* unigram ={quy = 500,hoạch = 300,động = 200}
* bigram={quy hoạch = 150,hoạch động = 100}
  + Phần 2: Tính entropy
* Ví dụ câu cần tính là “quy hoạch” với các số như trên
* Vì “quy” có trong unigram nên p1 = 0.95\*500/1000000 + (1-0.95)/1000000
* Vì “quy hoạch” có trong bigram nên p2 = 0.95\*150/500 + (1-0.95)\*p1
* Correct: Ví dụ cần sửa câu sentence = “quy hoajch do65ng”
  + B1: sentence = “<s> quy hoajch do65ng”
  + B2: sentence = “<s> quy hoajch do65ng </s>”
  + B3: arr\_sentence = {<s>,quy,hoajch,do65ng,</s>}
  + B4:
    - Ta bắt đầu xét từ từ “quy”, và từ này đã xuất hiện nên không sửa
    - Tiếp đến từ hoajch, từ này không có nên sửa. Ta gọi Can\_Change để tìm các từ có khả năng thay thế cho từ này cao nhất ta được list trả về là {hoạch}
    - Tiếp đến ta xs\_term cho từ “hoạch” bằng cách -log(2,Entropy(quy hoạch)\*Entropy(hoạch do65ng)
    - Vì list các từ thay thế chỉ có mỗi hoạch nên từ “hoajch” sẽ được thay bằng từ “hoạch”
    - Tương tự với các từ còn lại.

## Bộ tiền xử lý

Bài toán: Đưa câu query dạng chuỗi vào, làm sao để xóa những kí tự không cần thiết (vô nghĩa, có thể do bấm nhầm) và xóa các từ stopwords trong câu query đó được?

Mô hình hóa bài toán:

Base: Các từ stopwords, các kí tự không có nghĩa

Input: Câu query dạng chuỗi

Output: Câu query thỏa mãn yêu cầu

Ví dụ:

Stopwords là các từ: “bla”, “ble”,…

Các kí tự vô nghĩa là: “@”, “^”, …

Input: tôi yêu bla bla @ em

Output: tôi yêu em

Thuật toán: chỉ đơn giản là áp dụng tìm kiếm và thay thế chuỗi bằng regrex

## Bộ mở rộng query

Bài toán: Đưa vào một query dạng chuỗi, mô hình hóa query đó thành vector (tfIdf dạng rút gọn). Sau đó, mở rộng vector query này ra cho việc tìm kiếm ra thông tin phù hợp với người dùng hơn.

Mô hình hóa bài toán:

Base: Ontology có chứa từ đồng nghĩa, gần nghĩa

Input: câu query

Output: Vector query (dạng rút gọn) được mở rộng từ query đầu vào

Thuật toán:

B1: Tách câu query thành các token đặt tokens là danh sách các token

B2: Đếm số lần xuất hiện của các token

B3: Khởi tạo Vector rút gọn là rỗng

B4: Chạy qua từng token trong tokens:

B4.1: Dùng cây TfIdf để tìm ra giá trị tf-idf ứng với token. Ta có được cặp (token, tfidf\_of\_token)

B4.2: Truy xuất Ontology để lấy ra từ đồng nghĩa, gần nghĩa. Đánh trọng số cho nó ta được các cặp: (relation\_of\_token, weight \* tfidf\_of\_token)

B4.3: Thêm vào vector rút gọn

B5: Trả về Vector rút gọn đã xây dựng được ở bước 3

## Bộ so khớp:

Nhiệm vụ: Đưa vào 2 chuỗi kí tự, trả về độ tương đồng giữa 2 chuỗi kí tự đó (bề mặt, không xét về ngữ nghĩa)

Mô hình hóa bài toán:

Input: Hai chuỗi kí tự A và B

Output: Giá trị số thực mô tả độ tương đồng sym (0 <= sym <= 1)

Thuật toán:

B1: Dùng Greedy String Tiling để tính độ tương đồng giữa hai chuỗi, ta được giá trị là a

B2: Dùng Minimum Edit Distance tính độ tương đồng giữa hai chuỗi, ta được giá trị là b

B3: Dùng n-gram matching tính độ tương đồng ta được giá trị c

B4: Dùng một hàm heuristic (dựa trên cảm giác) đưa 3 tham số này vào và cho ra số thực đánh giá độ tương đồng: hàm được dùng ở đây là:

h(a, b, c) = (a \* c) / b, vì số có thể rất nhỏ (nhân lại không tránh khỏi sai số) nên ta logarit hóa nó cho tiện: h(a, b, c) = max(0, f(b) – (f(a) + f(c))), với f(x) = -log(x)

B5: Trả về giá trị hàm h ở bước 4

- Tính toán khoảng cách đến các cụm

Nhiệm vụ: Nhận vào một vector query (xem nó như một điểm trong không gian với số chiều không xác định) và trả về khoảng cách của nó tới các cụm khác

Mô hình hóa bài toán:

Base: Tọa độ của điểm trung tâm các cụm

Input: Vector query **v**

Output: Khoảng cách từ **v** tới các cụm

Thuật toán:

B1: Chạy qua từng cụm trong các cụm đã phân sẵn:

B1.1: Tính khoảng cách giữa vector query với điểm trung tâm của cụm

B1.2: Lưu lại khoảng cách đó

B2: Trả về các khoảng cách được tính ở bước 1.

## Bộ máy tìm kiếm

Nhiệm vụ: Đưa vào một câu query (dạng thô do người dùng trực tiếp nhập vào), đưa ra các tài liệu liên quan nhất đến query đó.

Mô hình hóa bài toán:

Input: Query dạng thô

Output: Các tài liệu liên quan

Ở bộ máy tìm kiếm, ta chia làm 2 bài toán con là Tìm kiếm và Xếp hạng.

### Tính điểm:

Nhiệm vụ: Đưa vector query vào và trả về n tài liệu liên quan nhất bằng cách dùng tính cosine giữa vector query và vector của các documents.

Mô hình hóa bài toán:

Input: Vector query (đã mở rộng), số nguyên dương k

Output: n tài liệu liên quan nhất (theo giá trị cosine)

Thuật toán:

B1: Chạy qua từng tài liệu:

B1.1: Dùng cây Trie đã lưu của tài liệu đang xét để tính cosine với vector query

B2: Sắp xếp các giá trị cosine này theo thứ tự giảm dần, rồi chọn k tài liệu đầu tiên

B3: Trả về thông tin ở bước 2

### Xếp hạng:

Nhiệm vụ: Nhận vào k tài liệu và điểm số của nó (điểm số càng cao thì càng liên quan đến query). Làm sao để sắp xếp lại theo thứ tự liên quan giảm dần k tài liệu này dựa trên ngữ nghĩa.

Mô hình hóa bài toán:

Input: K tài liệu, câu query, vector query

Output: K tài liệu đã sắp xếp

Thuật toán:

B1: Chạy qua từng tài liệu trong k tài liệu:

B1.1: Dùng bộ ba thuật toán so khớp (GST, n-grams, MED) để tính độ tương đồng của query với tiêu đề của tài liệu đang xét(dựa vào một hàm heuristic với ba giá trị này) được giá trị gọi là h\_title

B1.2: Tương tự như B1.1 nhưng so khớp giữa câu query (đã được gán nhãn) với nội dung của tài liệu đang xét (đã được gán nhãn). Đặt giá trị này là h\_content

B1.3: Dùng thuật toán **tính giá trị độ tương đồng giữa 2 cây cú pháp** để tính độ tương đồng giữa các cây cú pháp:

1. Cây cú pháp phân tích ra được từ câu query
2. Và các cây cú pháp từ các câu của tài liệu

Đặt giá trị này là h\_parsing

B1.4: Tính khoảng cách của vector query tới các cụm tài liệu đặt vào mảng: dist

B2: Sắp xếp lại k tài liệu theo thứ tự giảm dần dựa vào các giá trị của hàm h(i) tương ứng:

Giá trị hàm h được tính bằng:

h(i) = (h\_content + h\_title + h\_parsing) / dist(i)

# KẾT LUẬN

## Kết quả của đề tài

* Đã đáp ứng được những yêu cầu cơ bản của đề tài:
  + Crawl dữ liệu
  + Xây dựng được cơ sở tri thức đơn giản cho đề tài
  + Xây dựng được bộ tìm kiếm thông minh: Có khả năng chỉnh sửa chính tả cho query, tách từ/cụm từ tiếng Việt một cách tự động, sử dụng phân tích cú pháp câu để tìm kiếm.
* Giúp nâng cao các kỹ năng mềm của các thành viên:
  + Làm việc nhóm
  + Tự nghiên cứu đề tài mới
  + Viết báo cáo, trình bày ý tưởng

## Hạn chế của đề tài

* Bộ tài liệu thu thập được còn khá khiêm tốn
* Bộ tách từ tự động và sửa chính tả query độ chính xác chỉ dừng ở mức có thể chấp nhận được
* Giao diện chưa bắt mắt
* Cơ sở tri thức còn tổ chức khá đơn giản
* Kỹ năng viết báo cáo của các thành viên dù đã được cải thiện song vẫn chưa mấy khả quan

## Hướng phát triển

* Nâng cao độ chính xác của các chức năng sau: Tách từ, sửa lỗi, …
* Tổ chức cơ sở tri thức hoàn thiện hơn.
* Hoàn thiện giao diện cho đẹp mắt.
* Tìm được bộ tài liệu lớn hơn.

# TÀI LIỆU THAM KHẢO

* Chương 2, phần 2.2: Slide bài giảng “MỘT SỐ ỨNG DỤNG CỦA XỨ LÝ NGÔN NGỮ TỰ NHIÊN CHƯƠNG V” của thầy Nguyễn Trọng Chỉnh giảng viên ĐH CNTT
* Chương 2, phần 2.1: Slide “NLP Programming Tutorial 1 Unigram Language Models” và “NLP Programming Tutorial 2 Bigram Language Models” của thầy Graham Neubig thuộc trường Nara Institute of Science and Technology (NAIST)
* Tài liệu học tập của thầy Nguyễn Trọng Chỉnh