ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**----------\*\*\*----------**

**Báo Cáo**

**Đồ Án Cuối Kì**

Môn: **Các Hệ Thống Hỏi Đáp**

Đề Tài: **Hệ Thống Hỏi Đáp Lịch Sử Việt Nam**

GVHD: Thầy **Nguyễn Trọng Chỉnh**

Lớp: CS323.G11

Sinh Viên Thực Hiện:

* Lý Nhật Huy – MSSV: 12520174
* Phan Trung Đông – MSSV: 12520564
* Nguyễn Anh Quân – MSSV: 12520337



**Mục Lục**

[**Lời Cám Ơn!** 4](#_Toc441096963)

[**Lời Nói Đầu** 5](#_Toc441096964)

[**Chương I: TÌM HIỂU VỀ HỆ THỐNG HỎI ĐÁP.** 6](#_Toc441096965)

[**I.** **Tổng Quan về Hệ Thống Hỏi Đáp Tự Động:** 6](#_Toc441096966)

[**II.** **Lịch Sử Phát Triển:** 7](#_Toc441096967)

[**III.** **Phân Loại Hệ Thống Hỏi Đáp:** 11](#_Toc441096968)

[**1.** **Phân loại theo miền ứng dụng (domain):** 11](#_Toc441096969)

[**2.** **Phân loại theo khả năng trả lời câu hỏi:** 12](#_Toc441096970)

[**3.** **Phân loại theo hướng tiếp cận:** 12](#_Toc441096971)

[**IV.** **Các Bước Chung của Hệ Thống Hỏi Đáp:** 13](#_Toc441096972)

[**Chương II: MÔ HÌNH HỆ THỐNG** 16](#_Toc441096973)

[**I.** **Tổng Quan về Đề Tài:** 16](#_Toc441096974)

[**1.** **Tên Đề Tài:** 16](#_Toc441096975)

[**2.** **Mục Tiêu Hệ Thống:** 16](#_Toc441096976)

[**3.** **Sơ Lược Về Hệ Thống:** 16](#_Toc441096977)

[**II.** **Nguồn Dữ Liệu:** 17](#_Toc441096978)

[**III.** **Giới Thiệu Tổng Quan Công Cụ Sử Dụng:** 18](#_Toc441096979)

[**1.** **Eclipse:** 18](#_Toc441096980)

[**2.** **Lucene:** 19](#_Toc441096981)

[**3.** **VnTagger:** 24](#_Toc441096982)

[**4.** **VnTokenizer:** 24](#_Toc441096983)

[**IV.** **Mô Hình Hệ Thống:** 25](#_Toc441096984)

[**V.** **Mô Tả Nguyên Tắc Hoạt Động Của Thành Phần Chính Trong Mô Hình:** 26](#_Toc441096985)

[**1.** **Quá Trình Lập Chỉ Mục (Lucene Indexing):** 26](#_Toc441096986)

[**2.** **Thành phần  lập chỉ mục Index:** 27](#_Toc441096987)

[**Chương III: Cài Đặt Chương Trình:** 30](#_Toc441096988)

[**I.** **Mô Tả Xây Dựng Hệ Thống Trong Java:** 30](#_Toc441096989)

[**1.** **Quá Trình Lập Chỉ Mục – Indexing:** 30](#_Toc441096990)

[**2.** **Quá Trình Phân Tích Câu Hỏi – Tạo Câu Query:** 31](#_Toc441096991)

[**3.** **Quá Trình Tìm Kiếm:** 35](#_Toc441096992)

[**4.** **Rút Trích Thông Tin Câu Trả Lời:** 38](#_Toc441096993)

[**II.** **Giới Thiệu Thư Viện Sử Dụng:** 41](#_Toc441096994)

[**Chương IV: Thử Nghiệm và Đánh Giá Chương Trình:** 41](#_Toc441096995)

[**I.** **Giao Diện Chương Trình:** 41](#_Toc441096996)

[**II.** **Đánh Giá Hệ Thống:** 42](#_Toc441096997)

[**1.** **Dữ Liệu Sử Dụng Để Đánh Giá:** 42](#_Toc441096998)

[**2.** **Kết quả Đánh Giá:** 45](#_Toc441096999)

[**Chương V. Tổng Kết:** 52](#_Toc441097000)

[**I.** **Kết Luận – Đánh Giá:** 52](#_Toc441097001)

[**1.** **Kết Quả Đạt Được:** 52](#_Toc441097002)

[**2.** **Hướng Phát Triển:** 53](#_Toc441097003)

[**II.** **Tài Liệu Tham Khảo:** 53](#_Toc441097004)

# **1212569_21823227 copyLời Cám Ơn!**

Sau kì học, đây là thời điểm quyết định thành quả học tập của các sinh viên.Và đồ án cuối kì chính là bảng báo cáo cho thành quả học tập .

Để hoàn thành tốt đề tài này, trước tiên, chúng em xin chân thành cảm ơn các thầy cô và ban giám hiệu trường ĐH Công Nghệ Thông Tin nói chung và các thầy cô trong khoa nói riêng đã tạo mọi điều kiện giúp đỡ chúng em trong quá trình học tập và nghiên cứu.

Kế đến, chúng em xin chân thành gửi lời cảm ơn đến thầy Nguyễn Trọng Chỉnh, là giáo viên hướng dẫn và cũng là người đã nhiệt tình chỉ bảo cho chúng em trong suốt quá trình hoàn thành đề tài.

Cuối cùng xin gửi lời cảm ơn đặc biệt đến gia đình luôn là một chỗ dựa vững chắc cũng như luôn tạo mọi điều kiện tốt nhất cho chúng em được học tập.

# **1212569_21823227 copyLời Nói Đầu**

Với nhu cầu trao đổi thông tin của con người ngày càng cao, thông tin tràn ngập trên mọi phương tiện truyền thông, đặc biệt là sự phát triển rộng rãi của mạng toàn cầu Internet, hằng ngày con người phải xử lý một lượng thông tin khổng lồ. Những thắc mắc của người dùng dưới dạng truy vấn sẽ được tìm kiếm và trả về một cách ngắn gọn, súc tích, chính xác nhất những gì mà họ mong muốn. Đó chính là mục tiêu của hệ thống hỏi-đáp tự động. Rất nhiều hệ thống hỏi đáp thông tin qua mạng ra đời nhằm đáp ứng nhu cầu này.

Qua báo cáo này, Nhóm sẽ trình bày những gì đã tìm hiểu được về một Hệ Thống Hỏi Đáp Tự Động, Từ đó xây dựng một Hệ Thống Hỏi Đáp Tự Động cụ thể.

Báo Cáo được trình bày thành năm chương, nội dung được trình bày sơ bộ như  
dưới dây:

**Chương 1.** Giới Thiệu Tổng Quan về Hệ Thống Hỏi Đáp Tự Động. Các thành phần chính của Hệ Thống Hỏi Đáp Tự Động và một số vấn đề khác.

**Chương 2.** Giới thiệu về các công cụ sử dụng trong hệ thống. Trình bày về Mô hình của hệ thống hỏi đáp của Nhóm. Nguyên tắc hoạt động của từng thành phần trong mô hình. **Chương 3.** Trình bày về việc Cài đặt Hệ Thống trong Java cùng các thư viện được sử dụng.

**Chương 4.** Thử Nghiệm và Đánh Giá Hệ Thống qua bộ dữ liệu Test.

**Chương 5.** Tổng Kết.

# **Chương I: TÌM HIỂU VỀ HỆ THỐNG HỎI ĐÁP.**

1. **Tổng Quan về Hệ Thống Hỏi Đáp Tự Động:**

Hệ thống hỏi-đáp tự động (Question Answering-QA) là một hệ thống được xây dựng để thực hiện việc tìm kiếm câu trả lời cho một câu hỏi của người dùng. Hệ thống hỏi-đáp tự động liên quan đến 3 lĩnh vực lớn là xử lý ngôn ngữ tự nhiên (Natural Language Processing), tìm kiếm thông tin (Information Retrieval) và rút trích thông tin (Information Extraction).

Hệ thống hỏi đáp tự động có thể coi như một lựa chọn thứ hai bên cạnh hệ thống trích chọn thông tin khi người dùng muốn tìm kiếm thông tin họ cần. Hệ thống trích chọn thông tin nhận đầu vào là các từ khóa và trả về tập các tài liệu liên quan (có chứa các từ khóa đó). Kết quả mà hệ thống trích chọn thông tin (máy tìm kiếm) trả lại cho người dùng là rất lớn, có thể lên đến hàng nghìn trang web mà phần nhiều không chứa thông tin người dùng mong muốn. Trong khi đó, hệ thống hỏi đáp nhận đầu vào là câu hỏi dưới dạng ngôn ngữ tự nhiên của người dùng, trả lại các đoạn văn bản ngắn (các snippet) chứa câu trả lời trực tiếp cho câu hỏi [38]. Nghiên cứu về hệ thống hỏi đáp tự động hiện đang thu hút sự quan tâm của rất nhiều các nhà nghiên cứu từ các trường đại học, các viện nghiên cứu và cả các doanh nghiệp lớn trong ngành công nghệ thông tin, có ý nghĩa khoa học lẫn ý nghĩa thực tế. Rất nhiều các hội nghị thường niên về khai phá dữ liệu, trích chọn thông tin dành một chủ đề riêng cho các nghiên cứu về hệ thống hỏi đáp như TREC1 , CLEF2 … Bài toán xây dựng hệ thống hỏi đáp là một bài toán khó thuộc lĩnh vực xử lý ngôn ngữ tự nhiên. Chúng ta biết rằng ngôn ngữ tự nhiên vốn nhập nhằng, đa nghĩa, việc xác định được ngữ nghĩa của câu hỏi cũng như phát hiện ra câu trả lời là một thách thức không nhỏ. Không những vậy, giữa câu hỏi và câu trả lời còn tồn tại các quan hệ “ngầm” hay phụ thuộc vào ngữ cảnh.

Ví dụ câu hỏi “Ai là tác giả Nhật ký trong tù ?” Câu trả lời: “ Hồ Chí Minh viết Nhật ký trong tù trong thời gian Người bị chính quyền Tưởng Giới Thạch bắt giam và giải đi khắp các nhà giam ở Quảng Tây, Trung Quốc.”

“Nhật ký trong tù, dịch tiếng Hán- Ngục trung nhật ký là một tập thơ của Hồ Chí Minh,…”. Hay câu hỏi “Mozart sinh năm nào ?” và câu trả lời “….Mozart (1751 – 1791)…”. Để tìm được câu trả lời trên cho câu hỏi, hệ thống cần có cơ chế để biết được rằng “tác giả của một tập thơ là người viết tập thơ đó” hoặc cần học được các mẫu thường gặp của câu trả lời (các mẫu về ngày tháng năm sinh, về thời gian, địa chỉ….) tương ứng với từng loại câu hỏi. Các hệ thống Q&A trên thế giới hiện nay sử dụng rất nhiều các công cụ xử lý ngôn ngữ như: Bộ gán nhãn từ loại (POS Tagger), bộ nhận dạng tên thực thể (Named Entity Recognizer), bộ phân tích ngữ pháp (Parser)… và các tài nguyên ngôn ngữ như Wordnet [28], ontology [30] để phân tích câu hỏi và trích xuất câu trả lời.

1. **Lịch Sử Phát Triển:**

Nghiên cứu về QA đã bắt đầu từ những năm 60 của thế kỷ trước. Tuy nhiên, những khó khăn trong việc xử lý ngôn ngữ tự nhiên đã giới hạn việc nghiên cứu QA vào việc nghiên cứu các hệ chuyên gia trả lời câu hỏi trong các lĩnh vực nhỏ hẹp. Vào những năm gần đây, nghiên cứu về hệ QA tổng quát đã được đẩy mạnh cùng với sự xuất hiện của tiểu ban QA (QA Track) ở hội nghị TREC ( Text Retrieval Conference), sự thảo luận về QA ở diễn đàn CLEF (Cross Language Evaluation Forum).

Thiết kế một hệ thống hỏi-đáp không phải là khái niệm mới. Một số hệ thống đầu tiên đã được ra đời từ những năm 1960. Một ví dụ của một hệ thống như vậy là hệ thống BASEBALL được phát triển năm 1961 do nhóm tác giả Green, Chomsky, và Laughery [9]. Hệ thống này được thiết kế để cung cấp các thông tin về các số liệu thống kê của liên đoàn bóng chày Mỹ. Một hệ thống khác tương tự như BASEBALL đã được phát triển bởi Woods (1973) [32] và được đặt tên LUNAR. LUNAR có thể trả lời các câu hỏi liên quan đến các mẫu đá trở về từ tàu thăm dò mặt trăng Apollo. Hệ thống này dịch các câu hỏi thành một hay nhiều truy vấn cơ sở dữ liệu. Hệ thống TEAM được phát triển bởi Grosz (1983) [10] đã có những đặc điểm cơ bản như một chuỗi các biểu diễn ngữ nghĩa và một lược đồ phiên dịch làm cho nó tốt hơn 2 hệ thống trước. Điểm tương đồng giữa cả ba hệ thống là tất cả đều sử dụng cơ sở dữ liệu để lưu trữ cơ sở tri thức. Việc thiết kế của cơ sở dữ liệu này và các dữ liệu có cấu trúc được xây dựng một cách thủ công bởi tất cả các chuyên gia trong các lĩnh vực tương ứng.

Xử lý ngôn ngữ tự nhiên trong quá trình thực hiện đã nhận được rất nhiều sự quan tâm trong những năm 1970. Một trong những hệ thống như thế là Margie (Schank, Goldman, Riesbeck, & Rieger, 1975) [25]. Hệ thống này đã có thể xử lý các tài liệu mà sau này có thể trả lời các câu hỏi cơ bản. Điều này được thực hiện bằng cách phân tích và tổ chức tài liệu của thông tin trong một mô hình tương tự như bộ nhớ của con người. Điều này đã được cải thiện hơn bởi Lenhert, Dyer, Johnson, Yang, và Harley (1983) [17] trong hệ thống Boris. Boris và Margie là các hệ thống gần giống nhất với hệ thống hỏi-đáp lĩnh vực rộng (open-domain) hoạt động ngày nay.

Hệ thống hỏi-đáp nhận được rất nhiều mối quan tâm của cộng đồng các nhà nghiên cứu. Điều này bắt nguồn từ việc một lượng tài liệu lớn có gắn chỉ mục sẵn có trên Internet, cùng với các thành tựu trong lĩnh vực rút trích thông tin và sự gia tăng nhu cầu thương mại cho các sản phẩm QA trên toàn cầu. Một lợi thế nữa, dễ dàng cho việc phát triển các hệ thống QA phức tạp hơn, là các nguồn tài nguyên như là WordNet (Fellbaum, 1998) [8] và OpenCyc (Curtis, Matthews, & Baxter, 2005) [4]. WordNet cung cấp sự truy cập tới các thông tin ngữ nghĩa và các mối liên hệ ngữ nghĩa giữa các cụm từ. OpenCyC cung cấp sự truy cập tới thông tin ontology và tri thức thông dụng. Cả 2 hệ thống này đều được cung cấp miễn phí. Hội thảo TREC (Text Retrieval and Evaluation Conference) [29] về lĩnh vực QA năm 1999 đã đóng góp một phần to lớn cho sự phát triển của lĩnh vực QA với 20 thành viên tham gia. Sự kiện này đã thúc đẩy các nhà nghiên cứu chia sẻ kinh nghiệm và so sánh kết quả của họ sử dụng một độ đo chung.

Năm 2000, Carbonell và các đồng nghiệp trong bài báo The Vision Statement to Guide Research in Question Answering and Text Summarization [13] đã đưa ra các tư tưởng chung cho việc nghiên cứu Q&A. Theo đó một hệ thống hỏi đáp được người dùng đánh giá là hữu ích nếu đáp ứng được các tiêu chuẩn:

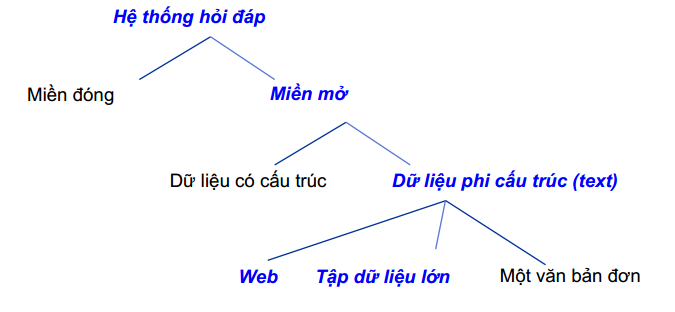
* Tính hợp lý về thời gian (Timeliness): Câu trả lời phải được đưa ra trong thời gian ngắn, ngay cả khi có hàng ngàn người dùng cùng truy nhập hệ thống một lúc. Các nguồn dữ liệu mới cần phải được tích hợp vào hệ thống ngay khi chúng sẵn sàng để có thế cung cấp cho người dùng câu trả lời cho những câu hỏi về các sự kiện có tính thời sự.
* Tính chính xác: Tính chính xác của hệ thống hỏi đáp tự động là cực kì quan trọng bởi việc đưa ra câu trả lời sai còn tai hại hơn nhiều là không đưa ra câu trả lời. Nghiên cứu về Q&A cần tập trung vào việc đánh giá tính đúng đắn của câu trả lời đưa ra, bao gồm cả phương thức để phát hiện các trường hợp mà dữ liệu hiện thời không chứa câu trả lời cho câu hỏi. Các thông tin mâu thuẫn trong dữ liệu cũng cần được tìm ra và các thông tin này cần được xử lý theo một cách phù hợp, nhất quán. Để đạt được sự chính xác, hệ thống Q&A cần được tích hợp các nguồn tri thức (world knowledge ) và cơ chế “bắt chước” việc suy luận thông thường (việc bắt chước có thể hiểu như là một quá trình học).
* Tính khả dụng: Hệ thống Q&A cần đáp ứng được các yêu cầu cụ thể của một người dùng. Các ontology trên từng miền cụ thể và ontology trên miền mở cần được tích hợp trong hệ thống. Hệ thống Q&A cần có khả năng khai phá câu trả lời 5 từ bất kì dạng dữ liệu gì (văn bản, web, cơ sở dữ liệu, …) và đưa ra câu trả lời dưới định dạng mà người dùng mong muốn, cho phép người dùng miêu tả ngữ cảnh của câu hỏi và cung cấp các thông tin giải thích, trích dẫn nguồn cho câu trả lời.
* Tính hoàn chỉnh: Câu trả lời hoàn chỉnh cho câu hỏi của người dùng là điều mà các hệ thống Q&A hướng tới. Trong nhiều trường hợp (câu hỏi về danh sách, nguyên nhân, cách thức…), các phần của câu trả lời nằm rải rác trong một văn bản, thậm chí trong nhiều văn bản. Vì vậy cần phải hợp nhất các phần này dựa trên các thông tin liên kết để tạo ra câu trả lời hoàn chỉnh.
* Tính thích hợp của câu trả lời: Trong ngôn ngữ tự nhiên, câu hỏi đưa ra luôn gắn với ngữ cảnh nào đó và câu trả lời cũng nằm trong một ngữ cảnh nhất định. Câu trả lời mà hệ thống Q&A đưa ra phải phù hợp ngữ cảnh với câu hỏi. Một hệ thống Q&A có khả năng giao tiếp (interactive Q&A) là cần thiết trong nhiều trường hợp bởi chuỗi các câu hỏi liên quan đến một vấn đề sẽ giúp làm sáng tỏ thông tin mà người dùng đang hỏi. Việc đánh giá một hệ thống Q&A cần hướng người dùng bởi ý kiến người dùng là đánh giá tốt nhất cho tính thích hợp của câu trả lời. Các tiêu chuẩn trên được đặt ra với mong muốn xây dựng được một hệ thống Q&A hoàn chỉnh. Tuy nhiên, không phải hệ thống nào cũng có khả năng thông minh và hoàn thiện như thế. Các nghiên cứu về Q&A hiện nay đang tập trung vào xây dựng hệ thống hỏi đáp có tính chính xác cao và có khả năng sử dụng nguồn dữ liệu web khổng lồ trên Internet.

Rất nhiều phương pháp tiếp cận khác nhau được sử dụng trong các hệ thống hỏi-đáp gần đây như phương pháp thống kê, phương pháp dựa trên luật, và các phương pháp tổng hợp. Hầu hết các hệ thống hỏi-đáp ngày càng trở nên phức tạp, thường sử dụng rất nhiều các mô-đun khác nhau, chẳng hạn như tìm kiếm thông tin, phân tích cú pháp câu, phân tích loại câu hỏi, phân tích ngữ nghĩa, và thậm chí dùng phương pháp suy luận (Moldovan et al., 2002) [21] để đánh giá, và xếp hạng câu trả lời. Với sự gia tăng tính phức tạp của hệ thống hỏi-đáp, sẽ rất khó khăn để xác định hiệu quả của hệ thống đến từ hiệu quả của giai đoạn nào.

1. **Phân Loại Hệ Thống Hỏi Đáp:**

Có nhiều cách phân loại hệ thống hỏi đáp dựa trên các tiêu chí khác nhau như: Phân loại theo miền ứng dụng, theo khả năng trả lời câu hỏi, theo cách tiếp cận giải quyết bài toán…

1. **Phân loại theo miền ứng dụng (domain):**

* Hệ thống hỏi đáp miền mở (open domain Question answering): Hệ thống trả lời bất kỳ câu hỏi nào được đưa vào. Khó khăn cho hệ thống miền mở đó chính là việc xây dựng các tri thức nên cho việc trả lời cũng như phân tích câu hỏi, các phương pháp hiện nay thường sử dụng một số các ontology khái quát hay các mạng tri thức như: wikipedia, bách khoa từ điển... Tuy nhiên, dữ liệu cho việc trích rút câu trả lời là phong phú, dễ thu thập.
* Hệ thống hỏi đáp miền đóng (close domain Question answering): Hệ thống tập trung vào trả lời các câu hỏi liên quan đến một miền cụ thể (giáo dục, y tế, thể thao...). Xây dựng hệ thống hỏi đáp miền đóng được coi là bài toán dễ hơn so với xây dựng hệ thống hỏi đáp miền mở vì có thể sử dụng các tri thức miền (thường là ontology của miền cụ thể).

Các nghiên cứu hiện nay về Q&A đang tập trung vào xây dựng hệ thống hỏi đáp trên miền mở, sử dụng nguồn dữ liệu phi cấu trúc (kho văn bản lớn hay dữ liệu web) để tìm câu trả lời. Các nghiên cứu mới và cải tiến những phương pháp cũ để có thể áp dụng cho nguồn dữ liệu web vốn đa dạng, nhiều “nhiễu” và trùng lặp đang rất được quan tâm.

1. **Phân loại theo khả năng trả lời câu hỏi:**

Hệ thống có khả năng trả lời các câu hỏi liên quan đến sự vật, hiện tượng,... dựa trên việc trích ra câu trả lời có sẵn trong tập tài liệu. Câu trả lời là các chuỗi ký tự trong một tài liệu. Kỹ thuật chính được sử dụng là xử lý chuỗi và từ khóa.

Hệ thống có cơ chế lập luận đơn giản: Trích xuất các câu trả lời có sẵn trong tập tài liệu sau đó sử dụng các suy luận để tìm mối liên kết giữa câu trả lời và câu hỏi. Hệ thống sử dụng các nguồn tri thức như ontology về từng miền cụ thể và ontology chung.

Hệ thống trả lời các câu hỏi yêu cầu khả năng tổng hợp: Các phần của câu trả lời được trích rút từ nhiều tài liệu sau đó được tổng hợp lại thành câu trả lời hoàn chỉnh. Câu hỏi thường là về danh sách, về cách thức, nguyên nhân...

Hệ thống có khả năng giao tiếp với người dùng: Trả lời chuỗi các câu hỏi của người dùng về cùng một vấn đề. Ví dụ các câu hỏi của người dùng như: “Giáo sư A sinh năm nào? Ở đâu? Ông ấy đang công tác ở đâu?”.

Hệ thống có khả năng lập luận tương tự: Có thể trả lời các câu hỏi có tính chất suy đoán, câu trả lời ẩn trong tập tài liệu. Hệ thống cần trích ra các luận chứng và sử dụng lập luận tương tự để tìm ra câu trả lời.

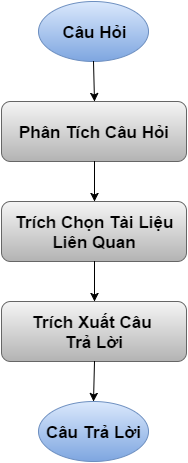
1. **Phân loại theo hướng tiếp cận:**

* Hướng tiếp cận nông (shalow): Nhiều phương pháp sử dụng trong Q&A dùng các kĩ thuật dựa trên từ khóa để định vị các câu, đọan văn có khả năng chứa câu trả lời từ các văn bản được trích chọn về. Sau đó giữ lại các câu, đoạn văn có chứa chuỗi ký tự cùng loại với loại câu trả lời mong muốn (ví dụ các câu hỏi về tên người, địa danh, số lượng…).
* Hướng tiếp cận sâu (deep): Trong những trường hợp khi mà hướng tiếp cận bề mặt không thể tìm ra câu trả lời, những quá trình xử lý về ngữ pháp, ngữ nghĩa và ngữ cảnh là cần thiết để trích xuất hoặc tạo ra câu trả lời. Các kĩ thuật thường dùng như nhận dạng thực thể (named-entity recognition), trích xuất mối quan hệ, loại bỏ nhập nhằng ngữ nghĩa,… Hệ thống thường sử dụng các nguồn tri thức như Wordnet, ontology để làm giàu thêm khả năng lập luận thông qua các định nghĩa và mối liên hệ ngữ nghĩa. Các hệ thống hỏi đáp dựa theo mô hình ngôn ngữ thống kê cũng đang ngày càng phổ biến.

1. **Các Bước Chung của Hệ Thống Hỏi Đáp:**

Một hệ thống hỏi đáp tự động thường gồm 3 bước chung sau:

* Bước 1: Phân tích câu hỏi: Bước phân tích câu hỏi tạo truy vấn cho bước trích chọn tài liệu liên quan và tìm ra những thông tin hữu ích cho bước trích xuất câu trả lời.
* Bước 2: Trích chọn tài liệu liên quan: Bước này sử dụng câu truy vấn được tạo ra ở bước phân tích câu hỏi để tìm các tài liệu liên quan đến câu hỏi.
* Bước3: Trích xuất câu trả lời: Bước này phân tích tập tài liệu trả về từ bước 2 và sử dụng các thông tin hữu ích do bước phân tích câu hỏi cung cấp để đưa ra câu trả lời chính xác nhất.



Các hệ thống hỏi đáp tự động hiện nay có kiến trúc rất đa dạng, tuy nhiên chúng đều bao gồm ba phần cơ bản như trên. Sự khác nhau chính giữa các hệ thống là ở quá trình xử lý trong từng bước, đặc biệt là ở cách tiếp cận trong việc xác định câu trả lời.

Cách tiếp cận theo trích chọn thông tin thuần túy (pure IR) là: chia nhỏ một tài liệu trong tập dữ liệu thành chuỗi các tài liệu con, trích chọn các tài liệu con có độ tương đồng lớn nhất với câu truy vấn (do bước phân tích câu hỏi tạo ra) và trả lại chúng cho người dùng.Thách thức lớn nhất ở đây là làm sao chia nhỏ được tài liệu thành các phần với kích cỡ tương ứng với kích cỡ của câu trả lời mà vẫn đủ lớn để có thể đánh chỉ mục được (nếu chia quá nhỏ thì số lượng tài liệu để đánh chỉ mục sẽ rất lớn, gây gánh nặng cho hệ thống trích chọn thông tin).

Cách tiếp cận theo xử lý ngôn ngữ tự nhiên (pure NLP) là: so khớp giữa biểu diễn ngữ pháp và (hoặc) biểu diễn ngữ nghĩa của câu hỏi với dạng biểu diễn ngữ pháp, ngữ nghĩa của các câu trong các tài liệu liên quan trả về. Khó khăn của cách tiếp cận này là hệ thống phải thực hiện việc phân tích ngữ pháp, ngữ nghĩa và so khớp đủ nhanh để đưa ra câu trả lời trong thời gian chấp nhận được, bởi số lượng các tài liệu cần xử lý là rất lớn trong khi các bước phân tích trên lại phức tạp và tốn nhiều thời gian.

Sự khác nhau trong cách trích xuất câu trả lời dẫn đến việc phân tích câu hỏi cũng trở nên đa dạng. Trong hướng tiếp cận theo trích xuất thông tin thuần túy, phân tích câu hỏi chỉ cần làm tốt việc tạo truy vấn, trong khi với hướng tiếp cận theo xử lý ngôn ngữ tự nhiên, câu hỏi cần được phân tích ngữ pháp, ngữ nghĩa một cách chính xác. Các hệ thống hiện nay thường là sự kết hợp giữa hai hướng tiếp cận, sử dụng hệ thống trích chọn thông tin để thu hẹp không gian tìm kiếm câu trả lời, đồng thời phân tích câu hỏi để tìm ra các thông tin về ngữ pháp, ngữ nghĩa nhằm tìm ra câu trả lời chính xác nhất. Kết quả của bước phân tích câu hỏi là đầu vào cho cả hai bước trích chọn tài liệu liên quan và trích xuất câu trả lời. Bước phân tích câu hỏi có ý nghĩa rất quan trọng, bởi nó ảnh hưởng đến hoạt động của các bước sau và do đó quyết định đến hiệu quả của toàn hệ thống. Chương 2 trình bày chi tiết các nội dung liên quan đến phân tích câu hỏi.

# **Chương II: MÔ HÌNH HỆ THỐNG**

1. **Tổng Quan về Đề Tài:**
2. **Tên Đề Tài:**

**HỆ THỐNG HỎI ĐÁP TỰ ĐỘNG LỊCH SỬ VIỆT NAM**

1. **Mục Tiêu Hệ Thống:**

Xây dựng một chương trình giúp trả lời các các hỏi liên quan đến Lịch Sử Việt Nam.

Nội dung các câu hỏi thuộc các khía cạnh:

* Nhân vật Lịch Sử
* Sự kiện Lịch Sử
* Địa danh Lịch Sử

Các câu hỏi thuộc loại câu hỏi Factoid (câu hỏi tầm thường), không có tính đánh đố hoặc yêu cầu phải suy diễn để tìm ra câu trả lời.

Vd:

* Ai là người đánh bạn quân Nam Hán?
* Ngô Quyền sinh vào thời gian nào?
* Lê Quý Đôn quê ở đâu?

1. **Sơ Lược Về Hệ Thống:**

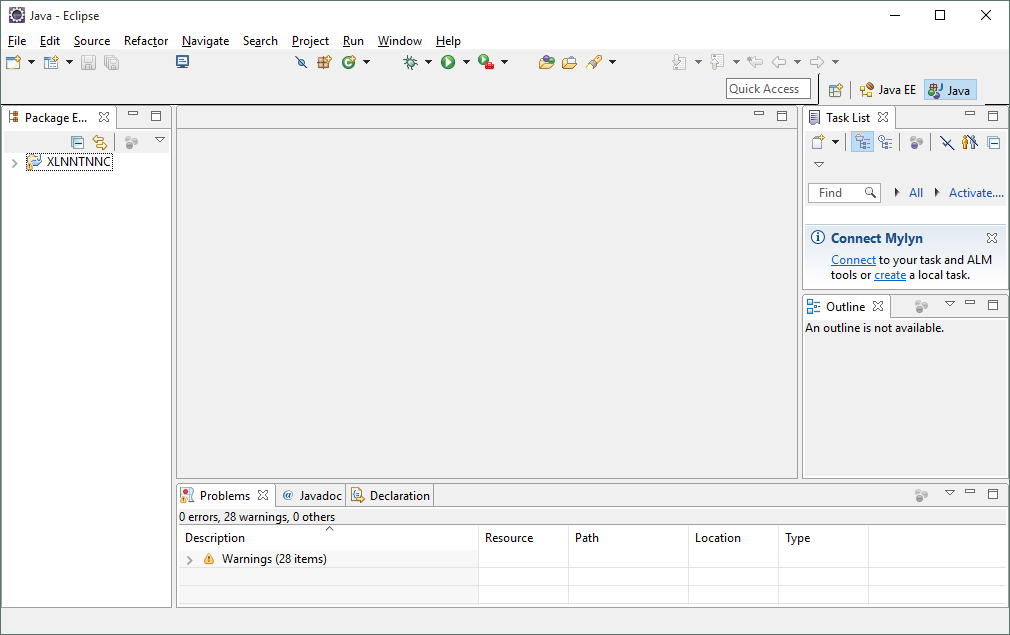
* Loại Hệ Thống Hỏi Đáp: Hệ thống hỏi đáp miền đóng (close domain Question answering): Hệ thống tập trung vào trả lời các câu hỏi liên quan đến một miền cụ thể (giáo dục, y tế, thể thao...). Xây dựng hệ thống hỏi đáp miền đóng được coi là bài toán dễ hơn so với xây dựng hệ thống hỏi đáp miền mở vì có thể sử dụng các tri thức miền (thường là ontology của miền cụ thể).
* Hướng Tiếp Cận: Hướng tiếp cận nông (shalow): Nhiều phương pháp sử dụng trong Q&A dùng các kĩ thuật dựa trên từ khóa để định vị các câu, đọan văn có khả năng chứa câu trả lời từ các văn bản được trích chọn về. Sau đó giữ lại các câu, đoạn văn có chứa chuỗi ký tự cùng loại với loại câu trả lời mong muốn (ví dụ các câu hỏi về tên người, địa danh, số lượng…).
* Cách tiếp cận theo trích chọn thông tin thuần túy (pure IR) là: chia nhỏ một tài liệu trong tập dữ liệu thành chuỗi các tài liệu con, trích chọn các tài liệu con có độ tương đồng lớn nhất với câu truy vấn (do bước phân tích câu hỏi tạo ra) và trả lại chúng cho người dùng.Thách thức lớn nhất ở đây là làm sao chia nhỏ được tài liệu thành các phần với kích cỡ tương ứng với kích cỡ của câu trả lời mà vẫn đủ lớn để có thể đánh chỉ mục được (nếu chia quá nhỏ thì số lượng tài liệu để đánh chỉ mục sẽ rất lớn, gây gánh nặng cho hệ thống trích chọn thông tin).

1. **Nguồn Dữ Liệu:**

Tất cả các Dữ Liệu của chương trình được nhóm thu thập về từ website: ***lichsuvietnam.vn***

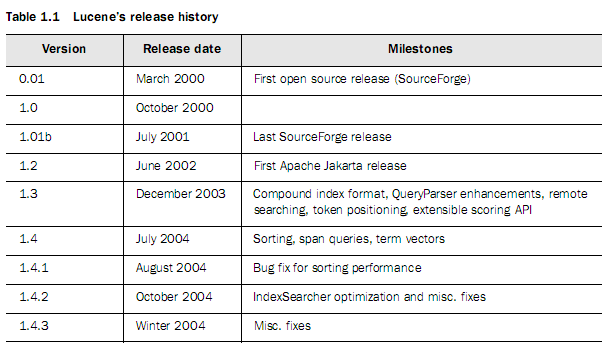
* Đây là Website cung cấp khá đầy đủ tài liệu về Lịch Sử Việt Nam. Nguồn dữ liệu trên Website đã được xác thực đáng tin cậy.
* Hiện tại, nhóm đã thu thập và tổng hợp về tất cả dữ liệu về các Nhân Vật Lịch Sử và Sự Kiện Lịch Sử trên Website lichsuvietnam.vn.
* Cụ thể, bộ dữ liệu bao gồm:
* Hơn 460 Nhân Vật Lịch Sử
* Hơn 80 Sự Kiện Lịch Sử Trung Đại
* Hơn 120 Sự Kiện Lịch Sử Cận Đại
* Hơn 240 Sự Kiện Lịch Sử Hiện Đại
* Mỗi dữ liệu về mỗi Nhân Vật Lịch Sử hay Sự Kiện Lịch Sử được lưu trữ riêng biệt trong các file text .txt
* Vd:
  + Dữ liệu về Nhân Vật Ngô Quyền được lưu trữ trong file NgoQuyen.txt
  + Dữ liệu về Sự Kiện Lịch Sử ngày 3-2-1930 được lưu trữ trong file 3-2-1930.txt

1. **Giới Thiệu Tổng Quan Công Cụ Sử Dụng:**
2. **Eclipse:**

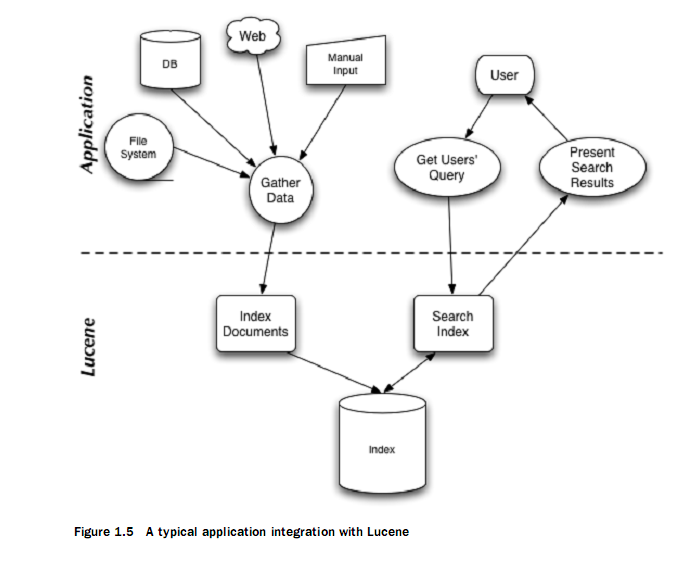
* Eclipse là một nền tảng phát triển mã nguồn mở dựa trên Java có thể mở rộng được. Eclipse bao gồm một tập hợp các dịch vụ để xây dựng các ứng dụng từ các thành phần plug-in. Eclipse được tạo thành bởi một tập hợp các plug-in chuẩn JDT (Java Development Tools).
* Dự án Eclipse được IBM bắt đầu vào tháng 11 năm 2001. Các dự án của Eclipse tập trung vào việc xây dựng nền tảng mã nguồn mở bao gồm các framework có thể mở rộng, các tool và thư viện để xây dựng, triển khai và quản lý các phần mềm.
* Eclipse là một tổ chức phi lợi nhuận, là nơi phát triển cộng đồng mã nguồn mở và hệ thống các sản phẩm, dịch vụ. Những người phát triển plug-in cho Eclipse hoặc những người sử dụng Eclipse như là nền tảng cho các ứng dụng phát triển phần mềm cần tuân thủ giấy phép EPL.
* Download, cài đặt Eclipse: [*http://www.eclipse.org/downloads/*](http://www.eclipse.org/downloads/)

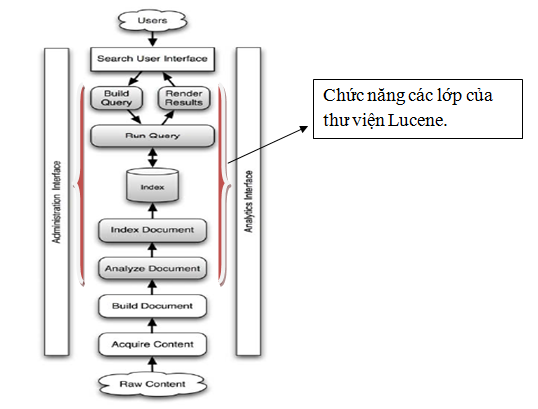
1. **Lucene:**

* Tác giả : được khởi xướng bởi Doug Cutting (<http://en.wikipedia.org/wiki/Doug_Cutting> ) và trở thành một trong những sản phẩm chất lượng cao của Apache Foundation tháng 9-2001.
* Source code : <http://www.manning.com/hatcher2>.
* Một vài ứng dụng search trên apache : <http://wiki.apache.org/jakarta-lucene/PoweredBy> .
* Lịch sử phát triển Lucene:



* Lucene là một thư viện cho việc tìm kiếm toàn văn, được phát triển bởi Dough Cutting (hiện đang làm việc cho Apache). Thư viện này cung cấp các hàm cơ bản hỗ trợ cho việc đánh chỉ mục và tìm kiếm. Để có thể sử dụng Lucene, bạn cần phải có sẵn dữ liệu. Dữ liệu có thể là tập hợp các tập tin dạng PDF, Word hay là các trang web HTML… Hoặc là dữ liệu lưu trong các hệ quản trị CSDL như MS SQL Server hay MySQL. Dùng Lucene, bạn có thể tiến hành đánh chỉ mục trên dữ liệu hiện có để sau này có thể thực hiện thao tác tìm kiếm toàn văn trên dữ liệu đó.
* Lucene là một thư viện tìm kiếm thông tin có khả năng xử lý và khả năng mở rộng ở mức cao, cho phép chúng ta có thể tích hợp vào các ứng dụng. Lucene là một dự án mã nguồn mở và nguyên thuỷ được phát triển bằng ngôn ngữ Java, ngày nay Lucene được phát triển bằng nhiều ngôn ngữ khác nhau như Delphi, Perl, C#, C++, Python, Ruby và PHP…
* Thành phần chức năng chính của Lucene bao gồm hai phần: Thành phần tạo chỉ mục và thành phần tìm kiếm. Đây là hai thành phần quan trọng cho một hệ thống tìm kiếm.
* Từ thư viện Lucene này, có nhiều kịch bản sử dụng sau :
* Dùng Lucene tích hợp vào ứng dụng hiện có. Muốn phát triển một semantic video search engine, trong đó đã có dữ liệu văn bản là các transcript và muốn có một công cụ hỗ trợ cho việc tìm kiếm dựa trên văn bản. Bằng cách này, người dùng có thể gõ vào từ khóa President Bush để tìm các video transcript có nói về President Bush. Sử dụng các hàm trong thư viện Lucene liên quan đến việc đánh chỉ mục và tìm kiếm, ta có thể thực hiện thao tác này khá dễ dàng.
* Xây dựng một search engine riêng. Lúc này cần phải có một web crawler chuyên đi thu thập các trang web trên Internet, để đem về đánh chỉ mục và cho phép tìm kiếm. Ứng dụng kiểu này có thể thấy tương tự ở trang www.baomoi.com. Trong ứng dụng này, web crawler sẽ được dùng để đi thu thập các tin từ các website (chủ yếu là tin tức, ví dụ vnexpress, tuoitre.com.vn, nld.com.vn, etc), sau đó tiến hành phân loại, lập chỉ mục để hỗ trợ tìm kiếm. Trong trường hợp cần web crawler, Nutch là một phần mềm mã nguồn mở có thể dùng , cũng do chính tác giả của Lucene là Dough Cutting phát triển. Để có thể tiến hành thu thập và lưu trữ hàng triệu trang web một cách có hiệu quả ở nhiều máy khác nhau, Hadoop sử dụng công nghệ của GoogleFS có thể được tích hợp cùng. Bên cạnh đó Solr, một phần mềm mã nguồn mở dùng cho xây dựng các search server, cung cấp giao diện bằng web với người sử dụng cũng được xây dựng dựa trên thư viện Lucene.
* Lucene ban đầu được viết hoàn toàn bằng Java. Sau đó được port qua các ngôn ngữ khác ví dụ như C, C++ ( CLucene), .NET (Lucene.NET ), Perl (Plucene), Ruby ( Ferret) và đặc biệt là PHP (Zend Framework ).

Sơ đồ thể hiện mối tương tác của một ứng dụng sử dụng Lucene.

* Những người mới tiếp xúc với Lucene thường nhầm Lucene như một chương trình tìm kiếm hoặc một web site search engine, điều này là hoàn toàn không đúng, Lucene là một thư viện nguồn mở hỗ trợ việc đánh chỉ mục và tìm kiếm toàn văn.
* Như ta thấy trong hình trên đó là các thành phần trong ứng dụng tìm kiếm cơ bản, ta cũng sẽ thấy rõ được chức năng các lớp trong thư viện Lucene hỗ trợ (phần màu trong ngoặc nhọn đỏ):
* Phân tích dữ liệu (dạng văn bản) để đánh chỉ mục: Analyze Document
* Đánh chỉ mục: Index Document và Index
* Thực hiện việc xây dựng các câu truy vấn và tìm kiếm trong chỉ mục: Build Query, Render results, RunQuery.
* Lucene cho phép thêm khả năng tìm kiếm cơ bản vào ứng dụng, Lucene có thể đánh chỉ mục (index) và tìm kiếm bất cứ dữ liệu gì (các loại dữ liệu này cần chuyển về dữ liệu dạng văn bản (text)).
* Thư viện này cung cấp các lớp với các hàm cơ bản hỗ trợ cho việc đánh chỉ mục và tìm kiếm. Lucene chỉ là một phần lõi của một search engine, nó không bao gồm web spider hay bộ phân tích định dạng dữ liệu (phân tích các định dạng dữ liệu khác nhau thành dữ liệu dạng văn bản (text)).
* Lucene không quan tâm đến nguồn dữ liệu, định dạng dữ liệu, thậm chí ngôn ngữ của nó, chỉ cần bạn có thể chuyển đổi nó sang text. Nghĩa là bạn có thể đánh chỉ mục và tìm kiếm các dữ liệu từ servers từ xa hoặc là local, các tập tin văn bản, MS word, html, pdf … (Wikipedia).

1. **VnTagger:**

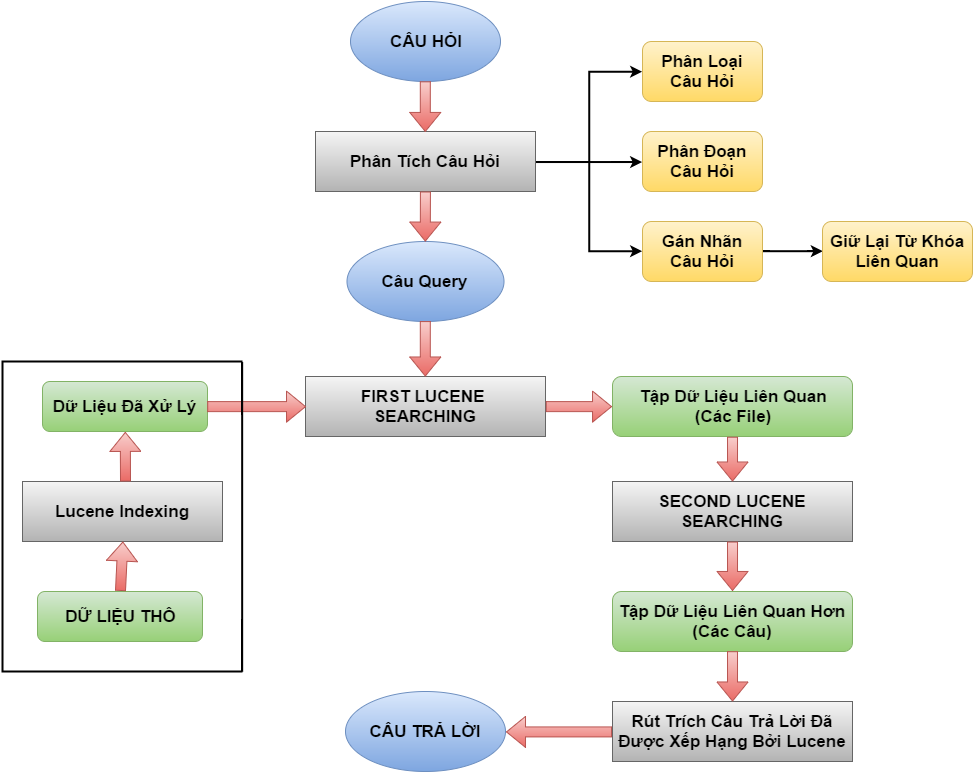
* VnTagger là chương trình gán nhãn từ loại tự động tiếng Việt mã nguồn mở, có độ chính xác cao.
* VnTagger thực hiện tách câu, tách từ và gán nhãn từ loại (danh từ, động từ, tính từ...) tự động cho văn bản tiếng Việt.

1. **VnTokenizer:**

* VnTokenizer là công cụ tách từ tiếng Việt được nhóm tác giả Nguyễn Thị Minh Huyền, Vũ Xuân Lương và Lê Hồng Phương phát triển dựa trên phương pháp so khớp tối đa (Maximum Matching) với tập dữ liệu sử dụng là bảng âm tiết tiếng Việt và từ điển từ vựng tiếng Việt.
* Công cụ được xây dựng bằng ngôn ngữ Java, mã nguồn mở. Có thể đễ dàng sửa đổi nâng cấp và tích hợp vào các hệ thống phân tích văn bản tiếng Việt khác.
* Quy trình thực hiện tách từ theo phương pháp khớp tối đa:
* Quy trình tách từ
  + Đầu vào của công cụ tách từ vnTokenizer là một câu hoặc một văn bản được lưu dưới dạng tệp.
  + Đầu ra là một chuỗi các đơn vị từ được tách.
* Các đơn vị từ bao gồm các từ trong từ điển cũng như các chuỗi số, chuỗi kí từ nước ngoài, các hình vị ràng buộc (gồm các phụ tố), các dấu câu và các chuỗi kí tự hỗn tạp khác trong văn bản (ISO, 2008). Các đơn vị từ không chỉ bao gồm các từ có trong từ điển, mà cả các từ mới hoặc các từ được sinh tự do theo một quy tắc nào đó (như phương thức thêm phụ tố hay phương thức láy) hoặc các chuỗi kí hiệu không được liệt kê trong từ điển.
* Công cụ sử dụng tập dữ liệu đi kèm là tập từ điển từ vựng tiếng Việt, danh sách các đơn vị từ mới bổ sung, được biểu diễn bằng ôtômat tối tiểu hữu hạn trạng thái, tệp chứa các biểu thức chính quy cho phép lọc các đơn vị từ đặc biệt (xâu dạng số, ngày tháng,…), và các tệp chứa các thống kê unigram và bigram trên kho văn bản tách từ mẫu.
* Với các đơn vị từ đã có trong từ điển, khi thực hiện tách từ cũng được xử lý hiện tượng nhập nhằng bằng cách kết hợp với các thống kê unigram và bigram. Chẳng hạn trong tiếng Việt thường gặp các trường hợp nhập nhằng như:
* Xâu AB vừa có thể hiểu là 1 đơn vị từ, vừa có thể là chuỗi 2 đơn vị từ A-B.
* Xâu ABC có thể tách thành 2 đơn vị AB-C hoặc A-BC.
* Đánh giá kết quả:

Kết quả đánh giá của công cụ được cho là ổn định đối với nhiều loại văn bản/ văn phong khác nhau. Độ chính xác trung bình đạt được là khoảng 94%.

1. **Mô Hình Hệ Thống:**

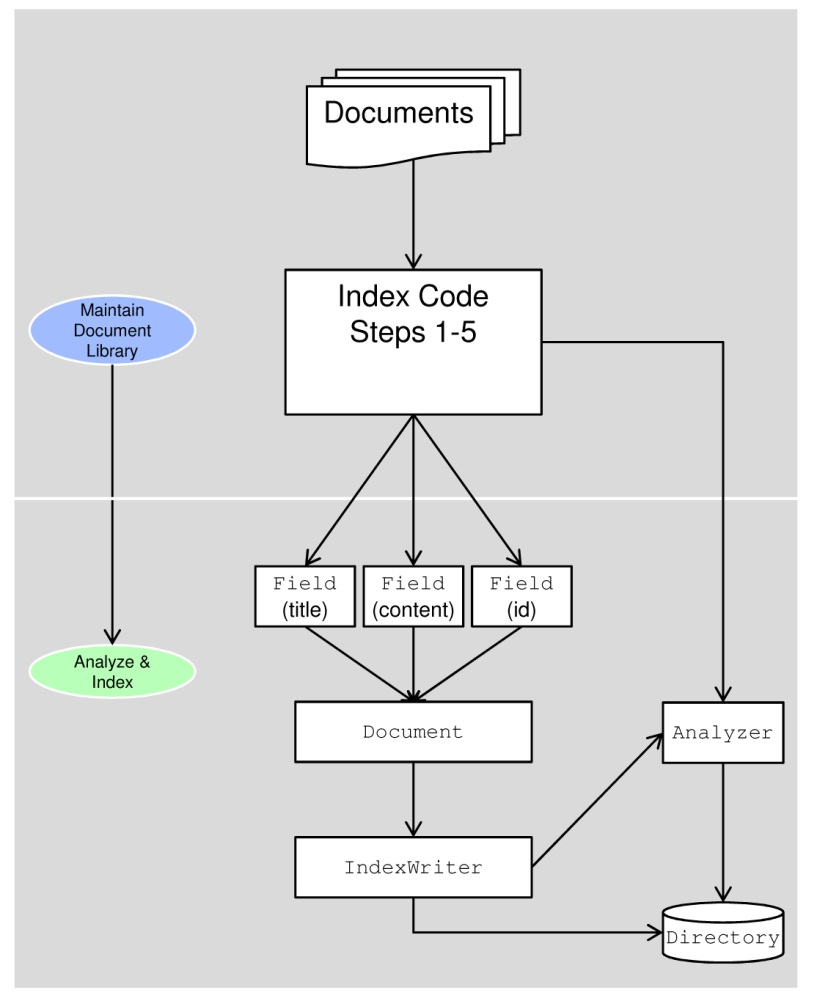


1. **Mô Tả Nguyên Tắc Hoạt Động Của Thành Phần Chính Trong Mô Hình:**
2. **Quá Trình Lập Chỉ Mục (Lucene Indexing):**

* Phương pháp lập chỉ mục:

Một Documemt có nhiều fields, mỗi fields có nhiều value.Và sau đây là hình vẽ biểu diển hoạt động lập Index của Lucene

Sau khi tách các field của document (3 field hoặc có thể nhiều hơn tạo nên 1 document) sau đó sẽ tiến hành phân tích và ghi dữ liệu vào Index



Mô hình  lập Index  của Lucene

* Bước 1: Chọn đường dẫn lưu Index
* Bước 2: Tạo  *Analyzer* cho tài liệu
* Bước 3: Tạo *Document* và thêm *Fields*
* Bước 4: Lấy *Index Writer* và thêm *Document*
* Bước 5: Đóng *Index*

1. **Thành phần  lập chỉ mục Index****:**

* Directory (Đường dẫn lưu chỉ mục)

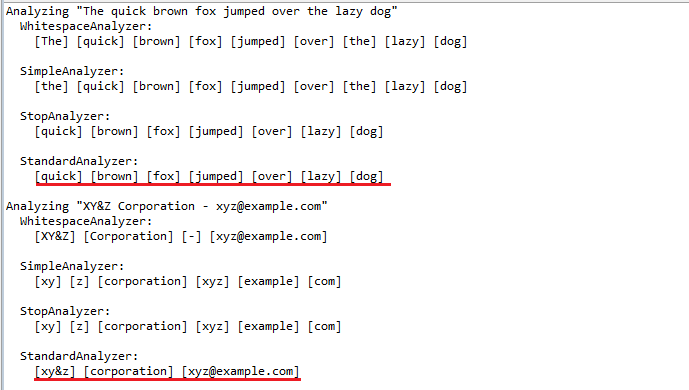
Gồm các kiểu Directory cơ bản sau:

* RAMDirectory: dữ liệu index sẽ được lưu tậm trên bộ nhớ ram (cách nhanh nhất).
* FSDirectory: Dữ liệu index sẽ được lưu trữ trong hệ thống file. Cho phép Lucene lựa chọn kiểu lưu trữ lý tưởng nhất trong 3 kiểu:
* SimpleFSDirectory
* MmapDirectory
* NIOFSDirectory
* Ở đây nhóm sử dụng cả 2 loại RAMDirectory và FSDirectory trong 2 lần index.
* Analyzers ( bộ phân tích tài liệu):

Chức năng của Analyser (phân tích) bao gồm *Tokenizer (phân đoạn),* *Stemmers (rút về dạng nguyên bản của từ)*  và  *stop-words filter (loại bỏ stop word).*

Công việc của Analyzer là phân tách từng field (trường) data thành những token có thể index được hoặc keywords.

Lucene hỗ trợ sẵn cho người dùng các bộ Analyser cơ bản: WhitespaceAnalyzer, SimpleAnalyzer, StopAnalyzer, StandardAnalyzer.

Chức năng của các bộ Analyser có thể được hình dung qua hình sau:

* Một vài đặc điểm quan trọng của các bộ phân tích có trong Lucene:
* WhitespaceAnalyzer không thực hiện lowercase(chuyển tất cả sang ký tự thường ) , vẫn giữ dấu “-“ , loại bỏ tất cả các khoảng trắng, và dựa vào các khoảng trắng để làm đường biên phân chia tokenize.
* SimpleAnalyzer thực hiện lowercase, vẫn giữ các từ nằm trong danh sách stop-word, dùng các ký tự không phải là các chữ cái alphabetic để làm đường biên phân chia tokenize.
* Cả SimpleAnalyzer và StopAnalyzer đều làm hỏng tên của tên các công ty như ví dụ trên (xy&z bị rã ra thành [xy] [z] , loại bỏ ký hiêu &) .
* StopAnalyzer và StandarAnalyzer loại bỏ các từ nằm trong Stop-Word ví dụ như “*the*” ở ví dụ trên.
* StandarAnalyzer vẫn giữ được tên của công ty và thực hiện chức năng lowercase , loại bỏ dấu “-” , giữ được địa chỉ email (*xyz@example.com*).
* Ngoài ra còn có các bộ Analyser dành riêng cho từng ngôn ngữ khác như Pháp, Đức, Nga... (chưa hỗ trợ cho Tiếng Việt).
* Ở đây nhóm sử dụng bộ StandardAnalyzer để phân tích văn bản.
* Document và Field

Document là một đối tượng cơ bản dùng để chứa các Field. Mỗi Field gồm 3 phần: name, type và value.

Field giúp Lucene có thể lưu trữ các thuộc tính bổ sung cho mỗi tài liệu trong chỉ mục. Mục đích của Field gồm có :

* Khả năng giới hạn tìm kiếm trên những đối tượng meta-data xác định (như chỉ có tiêu đề , tác giả..).
* Khả năng lưu trữ dữ liệu để xác định các tài liệu.

# **Chương III: Cài Đặt Chương Trình:**

1. **Mô Tả Xây Dựng Hệ Thống Trong Java:**
2. **Quá Trình Lập Chỉ Mục – Indexing:**

* Nhiệm vụ chính của quá trình này là sắp xếp và lưu trữu thông tin dữ liệu thành từng chỉ mục để giúp quá trình tìm kiếm trở nên nhanh hơn.
* Quá trình index được thực hiện bằng hàm IndexFiles()
* Đầu tiên tạo một Directory dùng để lưu chỉ mục. Ở đây ta sử dụng loại FSDirectory nên dữ liệu sẽ được ghi thẳng xuống đĩa cứng. Chỉ mục sẽ được lưu vào 1 folder mặc định sẵn là “index”. Dữ liệu cần lập chỉ mục sẽ là các file text được lưu ở docsPath.

|  |
| --- |
| String indexPath = "index";  String docsPath = "E:\\UIT\\Hoi dap\\docs";  **final** Path docDir = Paths.*get*(docsPath);  Directory dir = FSDirectory.*open*(Paths.*get*(indexPath)); |

* Tiếp đến tạo ra 1 Analyzer là StandardAnalyzer. Nhiệm vụ của Analyzer này là phân tách từng field (trường) data thành những token có thể index được hoặc keywords.

|  |
| --- |
| Analyzer analyzer = **new** StandardAnalyzer();  IndexWriterConfig iwc = **new** IndexWriterConfig(analyzer);  IndexWriter writer = **new** IndexWriter(dir, iwc); |

* Đối tượng IndexWriterConfig ở đây được tạo ra để nắm giữ tất cả config dùng để tạo ra đối tượng [IndexWriter](https://lucene.apache.org/core/5_2_1/core/org/apache/lucene/index/IndexWriter.html" \o "class in org.apache.lucene.index). [IndexWriter](https://lucene.apache.org/core/5_2_1/core/org/apache/lucene/index/IndexWriter.html) là đối tượng chính thực hiện chức năng tạo/cập nhật các chỉ mục trong suốt quá trình lập chỉ mục.
* Một khi chỉ mục được tạo, ta tiếp hành thêm các Document chứa dữ liệu vào.
* Quá trình thêm các Document này được thực hiện bởi hàm indexDoc

|  |
| --- |
| **static** **void** indexDoc(IndexWriter writer, Path file) **throws** IOException {  **try** (InputStream stream = Files.*newInputStream*(file)) {  BufferedReader in = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(file.toString()), StandardCharsets.***UTF\_8***));  Document doc = **new** Document();  Field pathField = **new** StringField("path", file.toString(), Field.Store.***YES***);  doc.add(pathField);  doc.add(**new** TextField("contents", **new** BufferedReader(**new** InputStreamReader(stream, StandardCharsets.***UTF\_8***))));  //Tạo thêm 1 field chứa line đầu để tăng độ chính xác cho kết quả doc.add(**new** TextField("tilte",in\_.readLine() ,Field.Store.***YES***));  writer.addDocument(doc);  }  } |

* Trong mỗi Document được tạo ra, ta thêm vào 3 field dữ liệu là : pathField, contents, tilte.

pathField: Chứa thông tin về đường dẫn của file tài liệu.

contents : Chứa nội dung file tài liệu.

tilte : Chứa dòng đầu tiên của file tài liệu. Vì dòng đầu tiên của file tài liệu thường chứa nhiều thông tin về sự kiện và thời gian nên khi tìm kiếm bằng trường này kết hợp với trường contents sẽ cho ra kết quả chính xác hơn, giúp câu trả lời không bị trôi khi tìm kiếm trên tập dữ liệu lớn. Ví dụ: Dòng đầu tiên của file *19-08-1945.txt* chứa nội dung: *19-08-1945: Cách mạng tháng tám, tổng khởi nghĩa giành chính quyền.*

Sau khi các Document được tạo IndexWriter sẽ được đóng lại.

|  |
| --- |
| writer.close(); |

1. **Quá Trình Phân Tích Câu Hỏi – Tạo Câu Query:**

* Khi người dùng nhập vào câu hỏi, câu hỏi hỏi sẽ được phân tích trước khi đưa vào hàm tìm kiếm.
* Quá trình phân tích gồm các công đoạn: phân đoạn từ (Tokenizer), gán nhãn (POS tagger), loại bỏ những từ không cần thiết (stop word).
* Nhóm sử dụng công cụ vnTokenizer và vnTagger của tác giả Lê Hồng Phương để phân đoạn và gán nhãn từ. Và sau đó dựa và các từ đã gán nhãn, sử dụng biểu thức chính quy để loại bỏ stop word (giữ lại các tag cần thiết).
* Quá trình phân tích câu hỏi được thực hiện bởi hàm ExtractQuery.

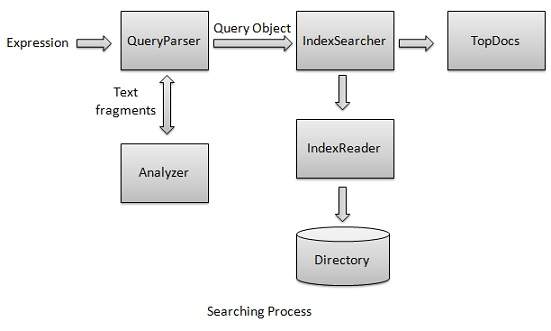
|  |
| --- |
| public String ExtractQuery(String input) throws IOException {  String segmented = tokenizer.segment(input);  System.*out*.println(segmented);  String tagged = tagger.tagging(segmented);  System.*out*.println(tagged);  String query\_extract="";  String regex\_ ="\\S+/(N|Np|V|M|E|A)"; //Tag giữ lại  Pattern pattern = Pattern.*compile*(regex\_);  Matcher matcher = pattern.matcher(tagged);  while (matcher.find())  {  query\_extract += matcher.group() + " ";  System.*out*.println(matcher.group() + " ");  }    System.*out*.println(query\_extract);  query\_extract = query\_extract.replaceAll("/\\S+", "").replaceAll("\_", " ");  System.*out*.println(query\_extract);  textArea.append("Query after extract----->"+ query\_extract+'\n');  return query\_extract;    } |

* Lấy ví dụ với câu hỏi : Ngô Quyền sinh năm bao nhiêu?
* Sau khi phân đoạn sẽ được: Ngô\_Quyền sinh năm bao\_nhiêu ?
* Sau khi gán nhãn sẽ được: Ngô\_Quyền/Np sinh/V năm/N bao\_nhiêu/P ?/?
* Dùng biểu thức chính quy giữ lại các tag /N/Np/V/M/E/A ta được: Ngô\_Quyền/N sinh/V năm/N
* Qua các bước xử lý string cơ bản ta thu được câu: Ngô Quyền sinh năm
* Ở đây ta cần xử lý thêm 1 bước cuối cùng nữa là đóng ngoặc kép cho danh từ riêng. Lí do phải đóng ngoặc kép cho danh từ là do đối tượng QueryParser khi gọi trong quá trình tìm kiếm sẽ tách câu query thành 2 loại: Single Term và Phrases. Khi không có ngoặc kép từ Ngô Quyền sẽ được chia thành Single Term và câu query truy vấn thực tế sẽ là “ngô” or “quyền” . Điều này làm cho kết quả tìm kiếm sẽ không được chính xác như mong đợi nếu như một tài liệu nào đó không liên quan nhưng chứa nhiều các từ ngô hoặc quyền. Vì vậy ta dùng ngoặc kép để chuyển danh từ riêng của ta về dạng Phrases. Quá trình này được xử lý trong hàm PatternRepalace.

|  |
| --- |
| public String PatternRepalace(String text){    String segmented = tokenizer.segment("hehe "+text);  String tagged = tagger.tagging(segmented);    String noun\_list="";  String regex\_replace ="\\S+/Np|/M";  Pattern pattern = Pattern.*compile*(regex\_replace);  Matcher matcher = pattern.matcher(tagged);  while (matcher.find())  {  noun\_list += matcher.group()+"|";  }    noun\_list = noun\_list.replace("\_", " ").replaceAll("/Np|/M","").replace("|", "").replaceAll("\\|$", "");  if(noun\_list==""){  noun\_list=" ";  }  if(!(noun\_list.toLowerCase().contains(" ")))  {  noun\_list=" ";  }  System.*out*.println("Black list ----->"+noun\_list);  black\_list=noun\_list;  String [] noun\_list\_array = noun\_list.split("\\|");    String result = text;      for(String noun : noun\_list\_array)  {  if(noun!="")  {  result = result.replace(noun, "\""+noun+"\"");  //System.out.println("-----> "+noun);  }  }    return result;  } |

1. **Quá Trình Tìm Kiếm:**
2. **Tìm kiếm tập văn bản (tìm lần 1):**

* Quá trình tìm kiếm là một trong những chức năng cốt lõi được cung cấp bởi Lucene. Sơ đồ sau minh hoạ quá trình tìm kiếm và sử dụng các lớp. IndexSearcher là thành phần quan trọng và cốt lõi nhất của quá trình tìm kiếm.

****

* Đối tượng IndexReader được gọi để đọc dữ liệu index được lưu ở folder *index.*

|  |
| --- |
| IndexReader reader = DirectoryReader.*open*(FSDirectory.*open*(Paths.*get*(index)));  IndexSearcher searcher = **new** IndexSearcher(reader);  Analyzer analyzer = **new** StandardAnalyzer();  QueryParser parser = **new** QueryParser(“contents”, analyzer);    query = parser.parse(QueryParseTitle(query\_for\_file\_search, "text") +" OR "+ query\_for\_file\_search); |

* Đối tượng IndexSearcher được tạo từ IndexReader.
* Đối tượng QueryParser được tạo với field tìm kiếm mặc định là contents . QueryParser cung cấp cho ta khả năng để tạo ra những câu query theo ý muốn.
* Ở phần Index ta có tạo ra 2 field là *contents* và *title*. Ta sẽ sử dụng cả 2 field này để trong QueryParser để tạo ra câu query tìm kiếm trong cả 2 field.
* Ví dụ: Sau khi phân tích câu hỏi ta được câu: **Ngô Quyền sinh năm**. Sau đó ta lại biết đổi câu này thành sử dụng hàm QueryParseTitle.

|  |
| --- |
| title:Ngô OR title:Quyền OR title:sinh OR title:năm OR "Ngô Quyền" sinh năm |

* Và câu query cuối cùng mà QueryParser biến đổi thành:

|  |
| --- |
| title:ngô title:quyền title:sinh title:năm contents:"ngô quyền" contents:sinh contents:năm |

* Câu query dùng để tìm kiếm được gửi tới IndexSearcher để thực hiện quá trình tìm kiếm. Sau đó *IndexSearcher* sẽ trả về một đối tượng TopDocs chứa các chi tiết tìm kiếm cùng với các Document ID là các kết quả tìm kiếm.

|  |
| --- |
| TopDocs results = searcher.search(query, 5 \* hitsPerPage);  ScoreDoc[] hits = results.scoreDocs; |

* Kết thúc qúa trình tìm kiếm đầu tiên ta nhận được kết quả là danh sách các file tài liệu liên quan được Lucene xếp hạng.

1. **Tìm Kiếm Đoạn Văn Bản (Tìm Lần 2):**

* Sau khi quá trình tìm kiếm đầu tiên kết thúc. Ta chọn ra 5 file có kết quả cao nhất gộp thành 1 văn bản, sau đó ta tiến hành theo từng câu và tìm kiếm lần 2.

|  |
| --- |
| **public** String [] SearchSentences(String Question) **throws** IOException, ParseException { StandardAnalyzer analyzer = **new** StandardAnalyzer();  // 1. Tạo index  Directory indexS = **new** RAMDirectory();  IndexWriterConfig config = **new** IndexWriterConfig(analyzer);  IndexWriter w = **new** IndexWriter(indexS, config);  String [] sentences = txt\_doc.split("\\.");  **for**(**int** i =0; i < sentences.length; i++)  {  String title="";  **for**(String value:txt\_doc\_check)  {  **if**(value.toLowerCase().contains(sentences[i].toLowerCase())){  String arr[] = value.split("\\.", 2);  title = arr[0];  **break**;  }  }  // Thêm Document và dữ liệu field  *addDoc*(w,Integer.*toString*(i+1),title, sentences[i]);  }  w.close();  // 2.Tạo và tuỳ biến câu query  String querystr = PatternRepalace(queryString);.  Query q = **new** QueryParser( "contents", analyzer).parse(QueryParseTitle(querystr, "title") +" OR "+ querystr);    System.***out***.println(q);  // 3. search  **int** hitsPerPage = 20;  IndexReader reader = DirectoryReader.*open*(indexS);  IndexSearcher searcher = **new** IndexSearcher(reader);  TopScoreDocCollector collector = TopScoreDocCollector.*create*(hitsPerPage);  searcher.search(q, collector);  ScoreDoc[] hits = collector.topDocs().scoreDocs;    // 4. Hiển thị kết quả    System.***out***.println("Found " + hits.length + " hits.");  String SentenceMerge = "";  **for**(**int** i=0;i<hits.length;++i) {  **int** docId = hits[i].doc;  Document d = searcher.doc(docId);  SentenceMerge += d.get("contents") + ".";  System.***out***.println((i + 1) + ". " + d.get("id") + "\t" + d.get("contents"));  }  String [] result = PatternMacth(SentenceMerge, Question);  **int** i = 0;  **for**(String value : result){  i++;  System.***out***.println(i + ". " + value+'\n');  textArea.append(i + ". " + value+'\n');  }  reader.close();  **return** result;  } |

|  |
| --- |
| **private** **static** **void** addDoc(IndexWriter w,String id, String title, String contents) **throws** IOException {  Document doc = **new** Document();  doc.add(**new** TextField("title", title, Field.Store.***YES***));  doc.add(**new** TextField("contents", contents, Field.Store.***YES***));  doc.add(**new** StringField("id", id, Field.Store.***YES***));  w.addDocument(doc);  } |

* Quá trình index và tìm kiếm này cũng giống như quá trình ở mục 1 và 3. Điều khác biệt ở đây là dữ liệu index lần này đc lưu trên RAM và index theo từng câu chứ không phải theo từng file.
* Sau khi quá trình tìm kiếm kết thúc, kết quả trả về là tập các câu liên quan được Lucene xếp hạng cao nhất.

1. **Rút Trích Thông Tin Câu Trả Lời:**

* Từ tập các câu liên quan được xếp hạng do Lucene trả về, ta tiến hành gộp lại thành 1 văn bản. Rồi sau đó dùng các biểu thức chính quy tương ứng với từng loại câu hỏi để rút trích câu trả lời.

|  |
| --- |
| **public** String[] PatternMacth(String SentenceMerge,**int** QuestionType){    String [] result = **new** String [Integer.*parseInt*(reNum.getText())];  String regex\_when\_born ="(?<=\\()[a-zA-Z\\u00BF-\\u1FFF\\u2C00-\\uD7FF\\d\\s]+(?=-|–)|(?<=-)\\d+(?=:)|(?<=sinh năm |sinh ngày )[^s,.ở]+";  String regex\_when\_die = "(?<=-|–)[a-zA-Z\\u00BF-\\u1FFF\\u2C00-\\uD7FF\\d\\s]+(?=\\))|(?<=-)\\d+(?=:)|(?<=mất năm |mất ngày )[^s,.ở]+";  String regex\_when\_event = "[^a-zA-Z]+\\sđến\\s[^a-zA-Z\\u00BF-\\u1FFF\\u2C00-\\uD7FF :.]+|\\d+-\\d+-\\d+|\\d+-\\d+|Năm \\d+\\s?:|Năm \\d+ \\(\\D+\\)\\s?:|\\d+ \\(\\D+\\)\\s?:|\\d+\\s?:";  String regex\_who\_event = "(?!"+black\_list+"|Nhà|Việt Nam|Triệu Đà|Sông )(?<!quân xâm lược |thành |\\S+Chiến thắng |quân |sông |nhà |Nhà |năm |Năm |văn học |triều đại |phong trào |ở |bến |xã |làng |quê |châu |huyện |\\()([\\p{Lu}][\\p{Ll}]+\\s?){2,}(?!\\))";  String regex\_where\_home = "(?<=sinh ở |sinh tại |Quê:|quê ở |Quê ở |quê tại |người làng |người thôn )[a-zA-Z\\u00BF-\\u1FFF\\u2C00-\\uD7FF\\d\\s,]+";  String regex\_where\_event = "(?<=sông |tại |trên |ở )(\\p{Ll}+\\s)?([\\p{Lu}][\\p{Ll}]+\\s?){2,}";  String regex\_which = "(?<=\\d+-\\d+-\\d+)[^.\\d+]+";  System.***out***.println("Black list ----->"+black\_list);    String regex\_final="\\.";    **switch** (QuestionType) {  **case** 1: regex\_final = regex\_when\_born;  System.***out***.println("Người -----> năm sinh");  textArea.append("Class: When -----> năm sinh\n");  **break**;  **case** 2: regex\_final = regex\_when\_die;  System.***out***.println("Người -----> năm mất");  textArea.append("Class: When -----> năm mất\n");  **break**;  **case** 3: regex\_final = regex\_when\_event;  System.***out***.println("Sự kiện -----> sự kiện");  textArea.append("Class: When -----> sự kiện\n");  **break**;  **case** 4: regex\_final = regex\_who\_event;  System.***out***.println("Class: Who");  textArea.append("Class: Who\n");  **break**;  **case** 5: regex\_final = regex\_where\_home;  System.***out***.println("Where -----> quê quán");  textArea.append("Class: Where -----> quê quán\n");  **break**;  **case** 6: regex\_final = regex\_where\_event;  System.***out***.println("Where");  textArea.append("Class: Where\n");  **break**;  **case** 7: regex\_final = regex\_which;  System.***out***.println("Which---> Sự kiện");  textArea.append("Class:Which----> Sự kiện\n");  **break**;  **default**:  **break**;  }    String [] sentences = SentenceMerge.split("\\.");  Pattern pattern = Pattern.*compile*(regex\_final);  **int** i=0;  **for**(String value : sentences){  **int** j=0;  Matcher matcher = pattern.matcher(value);  **while** (matcher.find()&&i<Integer.*parseInt*(reNum.getText()))  {  result[i] = matcher.group().replaceAll("[^\\u00BF-\\u1FFF\\u2C00-\\uD7FF\\w\\s-]", "").replaceAll("\\s+$", "").trim(); //xoá các ký tự đặc biệt  i++;  j++;  }  }    **return** result;  } |

* Ví dụ : Từ câu hỏi **Lý Thường Kiệt sinh năm nào?** Ta thu được tập các câu đươc xếp hạng cao nhất:

|  |
| --- |
| ﻿1. Lý Thường Kiệt Lý Thường Kiệt (Kỉ Mùi 1019 - Ất Dậu 1105) Lý Thường Kiệt (Kỉ Mùi 1019 - Ất Dậu 1105) Danh tướng, đại thần nhà Lý  2. 1077 :Lý Thường Kiệt đánh bại quân Tống xâm lược  3. Chính họ tên là Ngô Tuấn, tự Thường Kiệt, sau được ban theo họ vua, nên lấy tự làm tên và mang họ Lý, thành Lý Thường Kiệt  4. Cuộc kháng chiến chống quân Tống xâm lược của nhân dân ta do Lý Thường Kiệt chỉ huy đã thắng lợi hoàn toàn  5. Khi quân Tống tiến tới bờ bắc sông Cầu mới biết quân Đại Việt làm phòng tuyến sông Như Nguyệt rất kiên cố ngăn chặn chúng lại bên bờ nam, do Lý Thường Kiệt bố trí, dài khoảng 30km  6. Lý Thường Kiệt và triều đình nhà Lý đã chủ động đặt vấn đề nghị hòa, bớt đổ máu và mở đường thoát cho quân Tống, chấm dứt chiến tranh  7. Quân ta lâm vào tình trạng nguy khốn, Lý Thường Kiệt đã dùng kế sai người giả làm thần dân nấp ở bên bờ Nam bến đò Như Nguyệt, đọc thơ động viên tinh thần binh sĩ với những lời lẽ bất hủ được truyền mãi muôn đời: đó là bài thơ thần “Nam quốc sơn hà”  8. Nhưng theo Hoàng Xuân Hãn trong quyển “Lý Thường Kiệt thì: “Một học giả đời Tống là Thẩm Hoạt đã ghi tên bà là Lê Thị Yến Loan”  9. Dương Diên Nghệ Dương Diên Nghệ (…- Đinh Dậu 937) Danh tướng có sách chép là Dương Đình Nghệ, không rõ năm sinh  10. Tuy vậy Hoàng Xuân Hãn cho rằng đây chỉ là ghi theo âm, nên có thể chữ Yến Loan là cho chữ Ỷ Lan ghi chệch ra, cũng giống như chữ Lý Thường Kiệt chính quyển sách đó ghi Lý Thượng Cát |

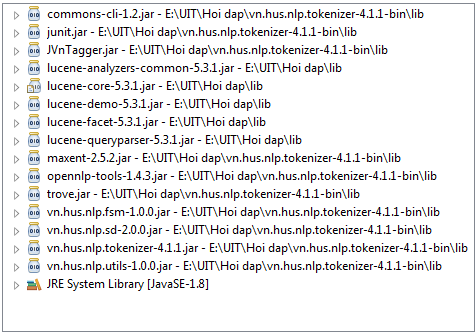
* Đây là câu hỏi thuộc loại 1 (năm sinh) nên sẽ dùng biểu thức chính quy regex\_when\_born

|  |
| --- |
| (?<=\\()[a-zA-Z\\u00BF-\\u1FFF\\u2C00-\\uD7FF\\d\\s]+(?=-|–)|(?<=-)\\d+(?=:)|(?<=sinh năm |sinh ngày )[^s,.ở]+ |

* Kết quả thu được:

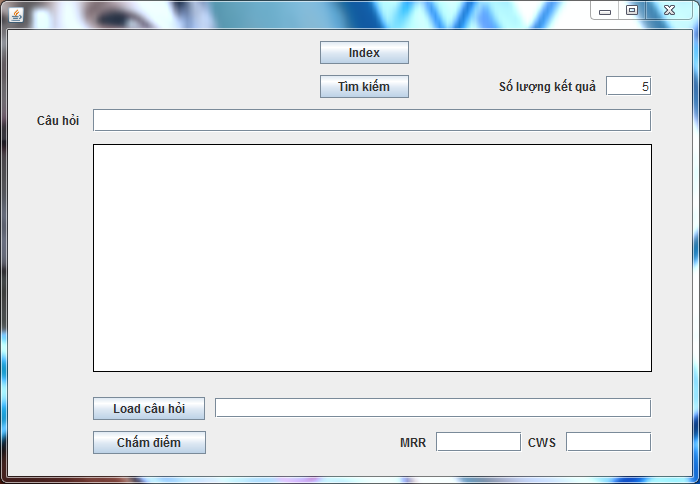
|  |
| --- |
| 1. Kỉ Mùi 1019  2. Kỉ Mùi 1019  3. 8  4. null  5. null |

1. **Giới Thiệu Thư Viện Sử Dụng:**



# **Chương IV: Thử Nghiệm và Đánh Giá Chương Trình:**

1. **Giao Diện Chương Trình:**



1. **Đánh Giá Hệ Thống:**
2. **Dữ Liệu Sử Dụng Để Đánh Giá:**

* Nhóm sử dụng một bộ dữ liệu test gồm 15 file dữ liệu về cả 3 khía cạnh dữ liệu đã nêu ở đầu.

Cụ thể, bộ dữ liệu gồm:



* Bộ câu hỏi mẫu: Nhóm đã lập một bộ câu hỏi mẫu dùng để test, bộ bao gồm 30 câu hỏi – đi kèm câu trả lời chính xác dùng để so sánh với câu trả lời từ hệ thống, cụ thể như sau:
* Ngô Quyền sinh năm nào?
* Kỉ Dậu 899
* Ngô Quyền quê ở đâu?
* Đường Lâm, huyện Phúc Lộc, châu Giao (nay thuộc tỉnh Hoà Bình)
* Ai là người đánh bại quân Nam Hán?
* Ngô Quyền
* Chu Văn An quê ở đâu?
* thôn Văn, Xã Quang Liệt, nay là Thanh Liệt, huyện Thanh Trì, Hà Nội (quê mẹ)
* Chu Văn An mất năm nào?
* 1370
* Miếu Thờ Chu Văn An ở đâu?
* làng Huỳnh Cung
* Nguyễn Công Trứ quê ở đâu?
* làng Uy Viễn, huyện Nghi Xuân, tỉnh Hà Tĩnh
* Nguyễn Công Trứ mất năm nào?
* Mậu Ngọ 1858
* Ai là người khai sáng ra nhà Lý?
* Lý Công Uẩn Lý Thái Tổ
* Lý Công Uẩn sinh năm nào?
* Giáp Tuất 974
* Lý Công Uẩn quê ở đâu?
* châu Cổ Pháp, lộ Bắc Giang
* Lý Công Uẩn sinh năm nào?
* Giáp Tuất 974
* Đinh Tiên Hoàng sinh năm nào?
* Ất Dậu 925
* Đinh Tiên Hoàng quê ở đâu?
* động Hoa Lư, châu Đại Hoàng (nay là huyện Gia Viễn, tỉnh Ninh Bình)
* Đảng Cộng Sản Việt Nam được thành lập năm nào?
* 3-2-1930
* Ngô Quyền đánh bại quân Nam Hán ở đâu?
* Sông Bạch Đằng
* Nhà Trần được thành lập vào năm nào?
* 1226
* Nhà Trần sụp đổ khi nào?
* 1440
* Quốc tế Cộng Sản được thành lập vào năm nào?
* 3-1919
* Quốc tế Cộng sản tan rã vào thời gian nào?
* 5-1943
* Hiệp ước Patơnốt được kí kết khi nào?
* 6-6-1884
* Ai là người lập ra nhà Hồ?
* Hồ Quý Ly
* Ngày Quốc Tế Thiếu Nhi là ngày nào?
* 1-6-1949
* Cao Bá Quát sinh năm nào?
* Kỉ Tị 1809
* Cao Bá Quát quê ở đâu?
* làng Phú Thị, huyện Gia Lâm, tỉnh Bắc Ninh (nay thuộc xã Quyết Chiến, huyện Gia Lâm, ngoại thành Hà Nội)
* Lý Thường Kiệt sinh năm nào?
* Kỉ Mùi 1019
* Lý Thường Kiệt là danh thần nhà nào?
* Nhà Lý
* Nhà Lý thành lập năm nào?
* 1010
* Chủ tịch Hồ Chí Minh từ bến Nhà Rồng (Sài Gòn) ra đi tìm đường cứu nước năm nào?
* 5-6-1911
* Đảng Cộng Sản Việt Nam thành lập năm nào?
* 3-2-1930

1. **Kết quả Đánh Giá:**
2. **Kết Quả Trả Lời của Hệ Thống:**

|  |  |  |
| --- | --- | --- |
| Câu Hỏi | Đáp Án | Câu Trả Lời Của Hệ Thống Đưa Ra |
| 1 | Kỉ Dậu 889 | Kỉ Dậu 899  Kỉ Dậu 899  null  null  null |
| 2 | Đường Lâm, huyện Phúc Lộc, châu Giao (nay thuộc tỉnh Hoà Bình) | Đường Lâm huyện Phúc Lộc châu Giao  null  null  null  null |
| 3 | Ngô Quyền | Ngô Quyền  Vua Nam Hán  Hoằng Tháo  Kiều Công Tiễn  Ngô Quyền |
| 4 | thôn Văn, Xã Quang Liệt, nay là Thanh Liệt, huyện Thanh Trì, Hà Nội (quê mẹ) | Văn Xã Quang Liệt nay là Thanh Liệt huyện Thanh Trì Hà Nội  xã Quang Vinh  null  null  null |
| 5 | 1370 | 1370  1370  1802  1369  null |
| 6 | làng Huỳnh Cung | làng Huỳnh Cung  Quốc Tử Giám  núi Phượng Hoàng  núi Phượng Hoàng  làng Huỳnh Cung |
| 7 | làng Uy Viễn, huyện Nghi Xuân, tỉnh Hà Tĩnh | null  null  null  null  null |
| 8 | Mậu Ngọ 1858 | Mậu Ngọ 1858  Mậu Ngọ 1858  14-11 Âm lịch  Phnom Pênh  null |
| 9 | Lý Công Uẩn Lý Thái Tổ | Lý Thái  Lý Thái  Trần Lý  Trần Thừa  Trần Lý |
| 10 | Giáp Tuất 974 | Giáp Tuất 974  Giáp Tuất 974  Giáp Tuất 974  Giáp Tuất 974  12-2 năm Giáp Tuất 8-3-974 |
| 11 | châu Cổ Pháp, lộ Bắc Giang | châu Cổ Pháp lộ Bắc Giang  châu Cổ Pháp lộ Bắc Giang  null  null  null |
| 12 | Giáp Tuất 974 | Giáp Tuất 974  Giáp Tuất 974  Giáp Tuất 974  Giáp Tuất 974  12-2 năm Giáp Tuất 8-3-974 |
| 13 | Ất Dậu 925 | 1225  4  Mậu Dần 1218  Mậu Dần 1218  16 tháng 6 |
| 14 | động Hoa Lư, châu Đại Hoàng (nay là huyện Gia Viễn, tỉnh Ninh Bình) | null  null  null  null  null |
| 15 | 3-2-1930 | 3-2-1930  24-2-1930  26-06-2000  20-03-1989  03-12-1980 |
| 16 | Sông Bạch Đằng | sông Bạch Đằng  Ái Châu  thành Đại La  Bạch Đằng  trấn Hải Môn |
| 17 | 1226 | 1226-1400  1010-1225  1400-1407  1225-1226  null |
| 18 | 1440 - 1407 | 1400-1407  1400-1407  1226-1400  1802 Nhâm Tuất  null |
| 19 | 3-1919 | 3-1919  3-1919  ừ 3 đến 7-2-1930  3-1919  3-1919 |
| 20 | 5-1943 | 27-10-1929  12-1920  12-1924  3-2-1930  03-02-1930 |
| 21 | 6-6-1884 | 3-1863  6-6-1884  5-6-1862  15-3-1874  5-6-1862 |
| 22 | Hồ Quý Ly | Hồ Quý Ly  Hồ Quý Ly  Nghệ An  Thanh Hóa  Hồ Quý Ly |
| 23 | 1-6-1949 | 01-06-1949  null  null  null  null |
| 24 | Kỉ Tị 1809 | Kỉ Tị 1809  Kỉ Tị 1809  null  null  null |
| 25 | làng Phú Thị, huyện Gia Lâm, tỉnh Bắc Ninh (nay thuộc xã Quyết Chiến, huyện Gia Lâm, ngoại thành Hà Nội) | null  null  null  null  null |
| 26 | Kỉ Mùi 1019 | Kỉ Mùi 1019  Kỉ Mùi 1019  8  null  null |
| 27 | Nhà Lý | null  null  null  null  null |
| 28 | 1010 | 1010-1225  1226-1400  12-2  8-3-974  12-2 |
| 29 | 5-6-1911 | 5-6-1911  5-6-1911  5-6-1911  8-6-1911  14-6-1911 |
| 30 | 3-2-1930 | 26-06-2000  20-03-1989  03-12-1980  07-12-1981  null |

1. **Điểm Trung Bình Thứ Hạng Đối Xứng [MRR]:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Câu Hỏi | Thứ Hạng | RR | Câu Hỏi | Thứ Hạng | RR |
| 1 | 1 | 1 | 16 | 1 | 1 |
| 2 | 1 | 1 | 17 | 1 | 1 |
| 3 | 1 | 1 | 18 | 1 | 1 |
| 4 | 1 | 1 | 19 | 1 | 1 |
| 5 | 1 | 1 | 20 | >5 | 0 |
| 6 | 1 | 1 | 21 | 2 | 0.5 |
| 7 | >5 | 0 | 22 | 1 | 1 |
| 8 | 1 | 1 | 23 | 1 | 1 |
| 9 | >5 | 0 | 24 | 1 | 1 |
| 10 | 1 | 1 | 25 | >5 | 0 |
| 11 | 1 | 1 | 26 | 1 | 1 |
| 12 | 1 | 1 | 27 | >5 | 0 |
| 13 | >5 | 0 | 28 | 1 | 1 |
| 14 | >5 | 0 | 29 | 1 | 1 |
| 15 | 1 | 1 | 30 | >5 | 0 |
| MRR | | | 0.717 | | |

1. **Điểm Trọng Số Tin Cậy [CWS]:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Câu Hỏi | Đúng? | ic/i | Câu Hỏi | Đúng? | ic/i |
| 1 | 1 | 1 | 16 | 1 | 0.75 |
| 2 | 1 | 1 | 17 | 1 | 0.765 |
| 3 | 1 | 1 | 18 | 1 | 0.778 |
| 4 | 1 | 1 | 19 | 1 | 0.789 |
| 5 | 1 | 1 | 20 | 0 | 0.75 |
| 6 | 1 | 1 | 21 | 0 | 0.714 |
| 7 | 0 | 0.857 | 22 | 1 | 0.727 |
| 8 | 1 | 0.875 | 23 | 1 | 0.739 |
| 9 | 0 | 0.778 | 24 | 1 | 0.75 |
| 10 | 1 | 0.8 | 25 | 0 | 0.72 |
| 11 | 1 | 0.818 | 26 | 1 | 0.731 |
| 12 | 1 | 0.833 | 27 | 0 | 0.704 |
| 13 | 0 | 0.769 | 28 | 1 | 0.714 |
| 14 | 0 | 0.714 | 29 | 1 | 0.724 |
| 15 | 1 | 0.733 | 30 | 0 | 0.7 |
| CWS | | | 0.808 | | |

# **Chương V. Tổng Kết:**

1. **Kết Luận – Đánh Giá:**
2. **Kết Quả Đạt Được:**

* Hệ thống hỏi đáp tự động đang nhận được sự quan tâm đặc biệt của các nhà nghiên  
  cứu và doanh nghiệp trong ngành công nghệ thông tin. Nhu cầu về một hệ thống hỏi đáp tiếng Việt là rất lớn.
* Qua báo cáo, Nhóm đã trình bày được một cách Tổng Quan về Hệ Thống Hỏi Đáp Tự Động.
* Từ kiến thức tổng quan đó, với sự hỗ trợ của các công cụ và thư viện bổ ích khác, Nhóm đã xây dựng được một Hệ Thống Hỏi Đáp cụ thể giúp trả lời các câu hỏi về Lịch Sử Tiếng Việt. Hệ thống tuy khá đơn giản nhưng đã thể hiện đầy đủ mô hình của một Hệ Thống Hỏi Đáp Tự Động. Kết quả thử nghiệm trên bộ dữ liệu gồm 30 câu hỏi, hệ thống của nhóm cho kết quả khá khả quan.
* Bên cạnh đó, Hệ Thống của Nhóm cũng còn khá nhiều hạn chế:
* Hệ Thống ban đầu chỉ trả lời được các câu hỏi tầm thường (factoid) và không có tính đánh đố.
* Giao diện hệ thống còn khá đơn giản.
* Quá trình tự đánh giá của hệ thống vẫn chưa thật sự tối ưu.

1. **Hướng Phát Triển:**

* Nghiên cứu mở rộng Hệ Thống ra Hệ Thống Hỏi Đáp miền Mở.
* Tối ưu giao diện chương trình.
* Tối ưu chức năng tự đánh giá của hệ thống.
* Mở rộng thêm nhiều mẫu câu Regex để xử lý câu hỏi và rút trích thông tin câu trả lời được tốt hơn.

1. **Tài Liệu Tham Khảo:**

* Phạm Thu Uyên, Nguyễn Đức Vinh, Nguyễn Đạo Thái. *Hệ thống hỏi đáp tự động*  
  *sử dụng trích rút quan hệ ngữ nghĩa trong kho văn bản tiếng Việt*. Nghiên cứu khoa  
  học sinh viên cấp trường Đại học Công Nghệ, 2009
* Roser Morante , Martin Krallinger , Alfonso Valencia and Walter Daelemans. Machine Reading of Biomedical Texts about Alzheimer's Disease. CLEF 2012 Evaluation Labs and Workshop. September 17, 2012
* Hirschman, L. & Gaizauskas, R. (2001) [Natural Language Question Answering. The View from Here](http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=96167). Natural Language Engineering (2001), 7:4:275-300 Cambridge University Press.
* Galitsky B, Pampapathi R. [Can many agents answer questions better than one](http://firstmonday.org/ojs/index.php/fm/article/view/1204/1124). *First Monday*. 2005;10.
* Nhóm Xử Lý Ngôn Ngữ Tiếng Việt: <http://vnlp.net/>
* Tài Liệu về Lucene:
* [Lucene Wikipedia indexer](http://schmidt.devlib.org/software/lucene-wikipedia.html) — bài hướng dẫn tìm kiếm với mã lệnh Java [Wikipedia data](http://download.wikimedia.org/wikipedia/)
* Article "[Behind the Scenes of the SourceForge.net Search System](http://blog.dev.sf.net/index.php?/archives/10-Behind-the-Scenes-of-the-SourceForge.net-Search-System.html)" viết bởi [Chris Conrad](https://vi.wikipedia.org/w/index.php?title=Chris_Conrad&action=edit&redlink=1)
* <https://lucene.apache.org/core/>
* <http://lucenevietnam.blogspot.com/>
* Tài Liệu về VnTagger và VnTokenizer:
* Bộ công cụ của Lê Hồng Phương - Department of Informatics, [Hanoi University of Science,](http://www.hus.vnu.edu.vn/)[Vietnam National University of Hanoi](http://www.vnu.edu.vn/).