|  |  |
| --- | --- |
| **khcnlogovnright** | **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC HOA SEN**  **KHOA KHOA HỌC VÀ CÔNG NGHỆ** |

**XÂY DỰNG KIẾN TRÚC**

**CỔNG THÔNG TIN TÌM VIỆC**

Giảng viên hướng dẫn : TS. Trần Vũ Bình

Nhóm sinh viên thực hiện : Lê Trung Hiếu (09015L)

Phùng Chí Nguyên (09023L)

Lê Dương Công Phúc (09025L)

Lớp : QL092L

**Tháng 12 năm 2011**

< Phiếu giao đề tài >

TRÍCH YẾU

Hiện tại, cùng với sự phát triển bùng nổ Internet trên thế giới cũng như tại Việt Nam, Internet đã cung cấp cho chúng ta khối lượng thông tin và dịch vụ khổng lồ. Bên cạnh đó, khối lượng thông tin ngày càng lớn phát sinh nhu cầu so khớp thông tin ngày càng nhiều nhưng vẫn phải đảm bảo hiệu năng và độ tin cậy của hệ thống. Các website so khớp thông tin trên Internet hiện nay có thể nói rất nhiều nhưng hầu như chưa có một kiến trúc chung, cùng với thực trạng các kiến trúc phần mềm giải quyết bài toán nghiệp vụ hiện nay cũng ít được phổ biến. Với mục tiêu xây dựng một kiến trúc phần mềm để giải quyết nhu cầu nói trên, nhóm chúng tôi đã chọn và tham gia thực hiện khoá luận lần này.

Nhóm chúng tôi đã xây dựng thành công nền tảng kiến trúc nghiệp vụ tổ chức và so khớp thông tin một cách hiệu quả, đồng thời áp dụng vào lĩnh vực cụ thể đang rất được quan tâm hiện nay đó là tìm việc trực tuyến. Đây là cơ hội tốt để nhóm có thể vận dụng những kiến thức đã được học áp dụng vào thực tế, Và quan trọng hơn, với những kiến thức đã được học ở trường cũng như tích lũy trong quá trình thực hiện đề tài, kèm theo đó là sự nỗ lực của các thành viên trong nhóm, chúng tôi đã hoàn thành đề tài đúng thời hạn và đúng yêu cầu ban đầu đề ra. Chúng tôi hy vọng kiến trúc này sẽ được tiếp tục phát triển hơn nữa và được ứng dụng, triển khai rộng rãi trên các hệ thống cần tổ chức và so khớp thông tin.

LỜI CẢM ƠN

Trước tiên, nhóm chúng tôi xin chân thành cảm ơn TS. Trần Vũ Bình đã tận tình hướng dẫn, góp ý và định hướng nhóm chúng tôi trong quá trình thực hiện khoá luận tốt nghiệp này. Khoá luận đã được hoàn thành cũng chính là nhờ sự nhắc nhở, đôn đốc và sự giúp đỡ nhiệt tình của thầy.

Xin chân thành cảm ơn các thầy cô Khoa Khoa Học & Công Nghệ đã nhiệt tình giúp đỡ nhóm trong suốt thời gian học tập tại trường. Chính thầy cô đã xây dựng cho chúng tôi kiến thức nền tảng và những kiến thức chuyên môn để nhóm hoàn thành khoá luận này và công việc sau này.

Nhóm chúng tôi cũng xin cảm ơn gia đình và bạn bè đã luôn bên cạnh, cổ vũ và động viên chúng tôi những lúc khó khăn để có thể hoàn thành tốt khoá luận này.

Trân trọng.

Nhóm thực hiện khoá luận.

MỤC LỤC

[TRÍCH YẾU i](#_Toc311737991)

[LỜI CẢM ƠN ii](#_Toc311737992)

[MỤC LỤC iii](#_Toc311737993)

[DANH MỤC HÌNH ẢNH vi](#_Toc311737994)

[DANH MỤC BẢNG vii](#_Toc311737995)

[DANH MỤC BIỂU ĐỒ viii](#_Toc311737996)

[THUẬT NGỮ ix](#_Toc311737997)

[NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN x](#_Toc311737998)

[1 Nhập đề 1](#_Toc311737999)

[2 Giới thiệu bài toán kiến trúc 2](#_Toc311738000)

[2.1 Bài toán 1: Tổ chức thông tin và sự linh hoạt của hệ thống so khớp 3](#_Toc311738001)

[2.2 Bài toán 2: Mức độ tương quan giữa những thẻ thông tin bất kỳ 4](#_Toc311738002)

[2.3 Bài toán 3: Phương pháp kết hợp việc so khớp và mức độ tương quan giữa các tag 5](#_Toc311738003)

[2.4 Kết quả mong muốn 5](#_Toc311738004)

[3 Lý thuyết về nền tảng kiến trúc 6](#_Toc311738005)

[3.1 Lý thuyết và phương pháp 7](#_Toc311738006)

[3.1.1 Tag 7](#_Toc311738007)

[3.1.2 Taxonomy 7](#_Toc311738008)

[3.1.3 Độ tương quan giữa các tag 11](#_Toc311738009)

[3.1.4 Kết hợp so khớp tag và độ tương quan giữa các tag 12](#_Toc311738010)

[3.1.5 Cây quyết định 12](#_Toc311738011)

[3.2 Kiến trúc phần mềm 31](#_Toc311738012)

[4 Giải quyết bài toán kiến trúc 31](#_Toc311738013)

[4.1 Áp dụng lý thuyết và phương pháp vào bài toán 31](#_Toc311738014)

[4.1.1 Áp dụng tag và taxonomy 31](#_Toc311738015)

[4.1.2 Áp dụng độ tương quan giữa các tag vào việc so khớp 33](#_Toc311738016)

[4.1.3 Áp dụng cây quyết định 33](#_Toc311738017)

[4.2 Áp dụng kiến trúc phần mềm vào bài toán 41](#_Toc311738018)

[4.3 Đánh giá kiến trúc 41](#_Toc311738019)

[5 Áp dụng giải pháp vào bài toán Job Zoom 42](#_Toc311738020)

[5.1 Thực trạng các website tuyển dụng hiện nay 42](#_Toc311738021)

[5.2 Những vấn đề Job Zoom cần giải quyết 42](#_Toc311738022)

[5.2.1 Vấn đề 1: Hỗ trợ người dùng viết CV theo ngành nghề. 42](#_Toc311738023)

[5.2.2 Vấn đề 2: Hỗ trợ nhà tuyển dụng trong việc đăng tải yêu cầu công việc một cách chi tiết và có trọng số theo ngành nghề. 43](#_Toc311738024)

[5.2.3 Vấn đề 3: So khớp hồ sơ ứng viên với yêu cầu tuyển dụng 43](#_Toc311738025)

[5.3 Kết quả mong muốn 43](#_Toc311738026)

[6 Giải quyết những vấn đề trong Job Zoom 44](#_Toc311738027)

[6.1 Khái quát phương pháp giải quyết vấn đề 44](#_Toc311738028)

[6.2 Ứng dụng lưu trữ dưới dạng tag kết hợp taxonomy 44](#_Toc311738029)

[6.3 Ứng dụng hệ thống so khớp kết hợp mức độ tương quan 44](#_Toc311738030)

[6.4 Ứng dụng cây quyết định 44](#_Toc311738031)

[6.4.1 Phương pháp tiền xử lý dữ liệu khi ứng dụng cây quyết định 45](#_Toc311738032)

[7 Kiến trúc cổng thông tin tìm việc JobZoom 46](#_Toc311738033)

[7.1 Điều kiện ra đời 46](#_Toc311738034)

[7.2 Mô hình kiến trúc khắc phục những điểm yếu của các website tìm việc hiện có 47](#_Toc311738035)

[7.3 Kiến trúc framework 47](#_Toc311738036)

[7.3.1 Kiến trúc tổng quan framework 47](#_Toc311738037)

[7.3.2 Hệ thống so khớp (Matching tool) kết hợp độ tương quan giữa các tag 47](#_Toc311738038)

[7.3.3 Data mining 47](#_Toc311738039)

[7.4 Mô hình kiến trúc Job Zoom có nhiều ưu điểm vượt trội 48](#_Toc311738040)

[7.4.1 Kiến trúc linh hoạt và khả năng mở rộng của framework 48](#_Toc311738041)

[7.4.2 Khả năng ứng dụng vào những ngành nghề khác nhau 48](#_Toc311738042)

[7.4.3 Khả năng ứng dụng vào những lĩnh vực khác 48](#_Toc311738043)

[7.4.4 Triển khai nhanh chóng 48](#_Toc311738044)

[7.4.5 Đảm bảo hiệu năng hệ thống 48](#_Toc311738045)

[7.5 So sánh Job Zoom với các website tìm việc hiện tại 49](#_Toc311738046)

[8 Đánh giá và hướng phát triển 49](#_Toc311738047)

[8.1 Những điểm làm được 49](#_Toc311738048)

[8.2 Những điểm hạn chế 50](#_Toc311738049)

[8.3 Hướng phát triển 50](#_Toc311738050)

[8.3.1 Xác thực độ tin cậy của CV và yêu cầu tuyển dụng 50](#_Toc311738051)

[8.3.2 Phát triển Semantic web 51](#_Toc311738052)

[8.3.3 Đánh trọng số cho các thuộc tính dựa vào độ sâu của taxonomy 51](#_Toc311738053)

[8.3.4 Phân tán dữ liệu 51](#_Toc311738054)

[8.3.5 Công cụ hỗ trợ lựa chọn ứng viên 51](#_Toc311738055)

[8.3.6 Thu thập thông tin việc làm tự động 51](#_Toc311738056)

[8.3.7 Áp dụng quy trình tuyển dụng vào hệ thống 51](#_Toc311738057)

[TÀI LIỆU THAM KHẢO 52](#_Toc311738058)

DANH MỤC HÌNH ẢNH

[Hình 1. Ví dụ về một taxonomy có cấu trúc đơn giản 8](#_Toc311802039)

[Hình 2. Ví dụ về flat taxonomy 10](#_Toc311802040)

[Hình 3. Ví dụ về hierarchical taxonomy 10](#_Toc311802041)

[Hình 4. Ví dụ về faceted taxonomy 11](#_Toc311802042)

[Hình 5. Ví dụ về network taxonomy 11](#_Toc311802043)

[Hình 6. Ví dụ về cây quyết định 14](#_Toc311802044)

[Hình 7. Phân chia bảng D thành những tập con Si 19](#_Toc311802045)

[Hình 8. Phân chia Si dựa vào phân lớp 20](#_Toc311802046)

[Hình 9. Độ lợi thông tin thuộc tính Travel cost/Km 21](#_Toc311802047)

[Hình 10. Độ lợi thông tin các thuộc tính còn lại trong Bảng 1 23](#_Toc311802048)

[Hình 11. Node gốc của cây quyết định sau lần lặp đầu tiên 24](#_Toc311802049)

[Hình 12. Bảng D được phân chia sau lần lặp đầu tiên 25](#_Toc311802050)

[Hình 13. Cây quyết định sau lần lặp đầu tiên 25](#_Toc311802051)

[Hình 14. Dữ liệu cho lần phân lớp thứ 2 26](#_Toc311802052)

[Hình 15. Tính Impurity degree cho lần phân lớp thứ 2 26](#_Toc311802053)

[Hình 16. Tính Impurity degree các thuộc tính cho lần phân lớp thứ 2 27](#_Toc311802054)

[Hình 17. Bảng dữ liệu sau khi chia theo phân lớp Gender 27](#_Toc311802055)

[Hình 18. Cây quyết định sau lần phân lớp thứ 2 28](#_Toc311802056)

[Hình 19. Cây quyết định đầy đủ sau 3 lần phân lớp 29](#_Toc311802057)

[Hình 20. 4+1 Architecture View Model 31](#_Toc311802058)

[Hình 21. Mô hình tổ chức thông tin theo tag kết hợp taxonomy 35](#_Toc311802059)

[Hình 22. Ví dụ tổ chức thông tin theo tag kết hợp taxonomy 35](#_Toc311802060)

[Hình 23. Cấu trúc cây quyết định khi sử dụng MS Analysis Services API 37](#_Toc311802061)

[Hình 24. Dữ liệu nhập xuất khi sử dụng MS Analysis Services API 37](#_Toc311802062)

[Hình 25. Dữ liệu đầu vào và đầu ra của Decision Tree Engine 40](#_Toc311802063)

[Hình 26. Tiến trình xây dựng cây quyết định của kiến trúc 40](#_Toc311802064)

[Hình 27. Cấu trúc bảng DecisionTreeNode và DecisionTreeNodeDistribution 41](#_Toc311802065)

DANH MỤC BẢNG

[Bảng 1. Ví dụ bảng dữ liệu lựa chọn phương tiện di chuyển 13](#_Toc311802026)

[Bảng 2. Ví dụ về bảng dữ liệu cần dự đoán phương tiện di chuyển 14](#_Toc311802027)

[Bảng 3. Bảng dữ liệu kết quả dự đoán phương tiện di chuyển 15](#_Toc311802028)

[Bảng 4. Lợi ích khi chia bảng D theo thuộc tính “Travel cost/km” 22](#_Toc311802029)

[Bảng 5. Kết quả độ lợi thông tin sau khi phân chia bảng D theo từng thuộc tính 24](#_Toc311802030)

[Bảng 6. Bảng dữ liệu cho lần phân lớp thứ 3 28](#_Toc311802031)

[Bảng 7. Bảng mô tả dữ liệu đầu vào khi sử dụng MS Analysis Services 37](#_Toc311802032)

[Bảng 8. Bảng mô tả dữ liệu đầu ra khi sử dụng MS Analysis Services 39](#_Toc311802033)

[Bảng 9. Dữ liệu ví dụ trước khi xử lý bằng kỹ thuật Pivot Transformation 42](#_Toc311802034)

[Bảng 10. Dữ liệu bảng 9 sau khi xử lý bằng kỹ thuật Pivot Transformation 42](#_Toc311802035)

[Bảng 11. Bảng dữ liệu các thông tin tuyển dụng trước khi Pivot trên cột Tag 50](#_Toc311802036)

[Bảng 12. Bảng kết quả sau khi Pivot Transformation 50](#_Toc311802037)

[Bảng 13. So sánh JobZoom framework với các website hiện tại 57](#_Toc311802038)

DANH MỤC BIỂU ĐỒ

[Biểu đồ 1. Mối liên hệ giữa giá trị Entropy cực đại và số lượng phân lớp 17](#_Toc311738096)

[Biểu đồ 2. Mối liên hệ giữa giá trị Gini index cực đại và số lượng phân lớp 18](#_Toc311738097)

THUẬT NGỮ

|  |  |  |
| --- | --- | --- |
| STT | Thuật ngữ | Giải thích |
|  | Actor |  |
|  | Attribute |  |
|  | Blog |  |
|  | Class |  |
|  | CV |  |
|  | Email |  |
|  | Forum |  |
|  | Framework |  |
|  | Kiểu dữ liệu Binary |  |
|  | Kiểu dữ liệu Continuos |  |
|  | Kiểu dữ liệu Nominal |  |
|  | Kiểu dữ liệu Ordinal |  |
|  | Pivot table |  |
|  | Resume |  |
|  | Semantic web |  |
|  | Series of rules |  |
|  | Social Network |  |
|  | Tag |  |
|  | Taxonomy |  |
|  | Training set |  |
|  | Database |  |
|  | Prefix | Tiền tố… |
|  | Server |  |

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

*TP. Hồ Chí Minh, ngày tháng 12 năm 2011*

**GIẢNG VIÊN HƯỚNG DẪN**

**Trần Vũ Bình**

# Nhập đề

Các kiến trúc phần mềm ứng dụng hiện nay đều đi nhiều về giao diện, giao tiếp - communication - giữa người dùng, những kiến trúc dành riêng giải quyết bài toán nghiệp vụ ít được phổ biến. Bên cạnh đó, hệ thống kiến trúc hiện tại đều rời rạc và không thống nhất, những ứng dụng tìm việc, tìm bạn, tìm mặt hàng, … có chức năng tương đối giống nhau - người dùng đăng thông tin, tìm kiếm và so khớp - nhưng những ứng dụng này chưa có kiến trúc chung. Từ nhu cầu thông tin nhiều, đa dạng, cần so khớp nhiều và nhanh, nhóm chúng tôi muốn xây dựng một kiến trúc phần mềm giúp người dùng đăng tải và so khớp thông tin một cách dễ dàng, nhanh chóng và hiệu quả hơn.

Trong khuôn khổ thực hiện đề tài, chúng tôi muốn cung cấp những giải pháp nhằm xây dựng kiến trúc nói trên, đồng thời áp dụng giải pháp kiến trúc này vào lĩnh vực tuyển dụng. Chúng tôi tạm gọi kiến trúc được áp dụng vào lĩnh vực tuyển dụng này là Job Zoom framework. Từ việc áp dụng vào một lĩnh vực cụ thể đó, chúng tôi sẽ tiến hành đánh giá giải pháp của chúng tôi đưa ra liệu có hợp lý và giải quyết hiệu quả các vấn đề đưa ra hay không.

Phương pháp thực hiện của chúng tôi là phân tích nhu cầu người dùng, các hệ thống website có sự so khớp thông tin, đặc biệt các trang tìm việc làm trong và ngoài nước, cùng với việc thu thập sưu liệu các thông tin hiện có trên internet, các mẫu thiết kế và kiến trúc cơ bản, lý thuyết khai thác dữ liệu và phương pháp phân loại động (cụ thể là cây quyết định và taxonomy) để đánh giá điểm mạnh, điểm yếu của các hệ thống hiện có, từ đó đưa ra những đưa ra những giải pháp có tính hiệu quả cao cho đề tài.

Mục tiêu của nhóm chúng tôi khi thực hiện đề tài “**xây dựng kiến trúc hệ thống cổng thông tin tìm việc”** là cung cấp giải pháp kiến trúc nghiệp vụ linh hoạt giúp người dùng đăng tải thông tin và so khớp, cung cấp một framework đảm bảo tính linh hoạt khi áp dụng trong nhiều hệ thống ngành nghề khác nhau, hỗ trợ người tìm việc hoàn thiện CV tìm việc của mình và hỗ trợ nhà tuyển dụng trong việc đánh giá ứng viên. Đồng thời, doanh nghiệp cung cấp website tìm việc có thể dễ dàng xây dựng hoặc hoàn thiện website của mình dựa trên Job Zoom framework.

Với những ưu điểm của kiến trúc nghiệp vụ đăng thông tin và so khớp nói chung và Job Zoom framework nói riêng, chúng tôi hy vọng đề tài sẽ được ứng dụng và triển khai rộng rãi trong các website so khớp, đánh giá cũng như các website tìm việc trực tuyến, góp phần nâng cao hiệu quả và tiện ích cho người dùng đưa ra những quyết định hiệu quả và kịp thời.

# Giới thiệu bài toán kiến trúc

Ngày nay, Internet mang lại rất nhiều tiện ích hữu dụng cho người sử dụng, một trong các tiện ích phổ thông của Internet là hệ thống thư điện tử (email), trò chuyện trực tuyến (chat), máy truy tìm dữ liệu (search engine) và nhiều dịch vụ tiện ích khác. Những hệ thống này cung cấp một khối lượng thông tin và dịch vụ khổng lồ trên Internet. Đã 29 năm trôi qua kể từ ngày 30/8/1982[[1]](#footnote-1), ngày hệ thống liên lạc bằng thư điện tử (email) được cấp bản quyền, có thể nói đây là một phương tiện thông tin rất nhanh, vận tốc truyền thư điện tử chỉ vài giây đến vài phút và chi phí rất nhỏ không đáng kể so với gửi qua đường bưu điện. Việc gửi email cho một nhóm thật sự trở nên khó khăn khi phải nhập danh sách người nhận gây mất thời gian, việc email bị nhận nhầm là thư rác sẽ cao. Để giải quyết vấn đề này, hộp thư nhóm (group) đã ra đời nhằm giải quyết được khuyết điểm của email, việc gửi email cho một nhóm trở nên dễ dàng hơn. Chưa dừng lại ở đó, diễn đàn (forum) ra đời nhằm bổ sung những thiếu sót của group – việc trao đổi thông tin giữa một nhóm người được thực hiện một cách dễ dàng và nhanh chóng hơn, những bài viết cũng được phân chia thành từng chủ đề cụ thể và rõ ràng hơn. Tuy nhiên, việc phân loại bài viết trên forum cũng không phải là vấn đề đơn giản, một bài viết trong forum chỉ thuộc về duy nhất một chủ đề, do đó, khi một bài viết thuộc về nhiều chủ đề khác nhau, khi chúng ta đăng bài viết trong chủ đề này, thì khi vào chủ đề kia lại không thấy và ngược lại. Để giải quyết vấn đề này, thẻ (tag) đã ra đời. Việc ra đời của tag cùng với blog và mạng xã hội đã đánh dấu bước ngoặc phát triển mới của internet. Một bài viết sẽ được tag những tag liên quan, sự liên kết các tag này góp phần giải quyết cho việc phân loại bài viết trở nên dễ dàng hơn. Tag gợi ý cho chúng tôi thực hiện một hệ thống, mà trên đó *các thực thể khác nhau được định nghĩa thông qua tag, những tag được liên kết với nhau tạo thành một cây yêu cầu hay cây thông tin giúp dễ dàng phân loại và đánh giá đối tượng trở nên dễ dàng hơn*. Chẳng hạn khi bạn muốn mua một cái áo, bạn sẽ miêu tả những chi tiết mà bạn cần chẳng hạn như: sơ mi, tay dài, có khuyu, màu xanh,… bên cạnh đó, các công ty dệt may cũng sẽ mô tả sản phẩm của mình dưới dạng tag, việc so sánh giữa cây yêu cầu của người tiêu dùng và cây thông tin về sản phẩm của công ty dệt may sẽ gợi ý cho người tiêu dùng mua sản phẩm mình yêu thích và nhà sản xuất bán được sản phẩm.

Các kiến trúc phần mềm ứng dụng hiện nay đều đi nhiều về giao diện, giao tiếp (communication) giữa người dùng, những kiến trúc dành riêng giải quyết bài toán nghiệp vụ ít được phổ biến. Bên cạnh đó, hệ thống kiến trúc rời rạc và không thống nhất, những ứng dụng tìm việc, tìm bạn, tìm mặt hàng, … có chức năng tương đối giống nhau - người dùng đăng thông tin, tìm kiếm và so khớp - nhưng những ứng dụng này chưa có kiến trúc chung. Từ ***nhu cầu thông tin nhiều, đa dạng, cần so khớp nhiều và nhanh***, chúng tôi nghĩ về một kiến trúc phần mềm giúp người dùng đăng tải và so khớp thông tin một cách dễ dàng, nhanh chóng và hiệu quả. Kiến trúc này phải giải quyết được những bài toán sau:

## Bài toán 1: Tổ chức thông tin và sự linh hoạt của hệ thống so khớp

Bạn cần tìm kiếm một sản phẩm để mua - nhà sản xuất cần tìm khách hàng để bán sản phẩm, làm sao để bạn có thể tìm kiếm những sản phẩm thoả mãn nhu cầu và doanh nghiệp cũng đạt được lợi nhuận cao nhất từ việc bán hàng? Bạn cần tìm kiếm một người bạn đã lâu không còn liên lạc – người bạn của bạn đã tham gia một mạng xã hội, làm sao để bạn và người bạn ấy có thể tìm ra nhau? Bạn cần tìm một công việc phù hợp với kỹ năng và trình độ của bạn thân – doanh nghiệp cần tìm những ứng viên đáp ứng đầy đủ các yêu cầu tuyển dụng?… Đây có thể là những câu hỏi rất thường gặp trong thực tế. Internet là một kho thông tin vô tận, được cung cấp từ hàng triệu website trên khắp thế giới, do có quá nhiều thông tin nên việc tìm kiếm được đúng thông tin cần thiết cũng không đơn giản. Bên cạnh đó, thông tin về một đối tượng thường được thể hiện thông qua những thuộc tính cứng được định nghĩa sẵn về số lượng và kiểu dữ liệu, giá trị của những thuộc tính này được lưu trữ thông qua các records trong cơ sở dữ liệu, cách tổ chức dữ liệu này không linh hoạt và khó khăn trong việc so khớp dữ liệu, phương pháp tìm kiếm thường dùng trong cách lưu trữ này là phương pháp vét cạn nên mất nhiều thời gian và hiệu suất không cao. Chính vì lý do trên, việc tổ chức và phân loại đòi hỏi phải linh động, không quá cứng nhắc để có thể so khớp giữa yêu cầu và “sản phẩm đáp ứng nhu cầu” một cách dễ dàng và sự linh động này giúp cho hệ thống so khớp đánh giá chính xác hơn mức độ so khớp giữa yêu cầu và “sản phẩm đáp ứng yêu cầu”. ***Bài toán cần phải đạt được là tổ chức thông tin một cách linh hoạt, có thể dễ dàng tìm kiếm, so sánh thông tin nội tại mà vẫn đảm bảo được khả năng so khớp giữa các cây thông tin với nhau***

## Bài toán 2: Mức độ tương quan giữa những thẻ thông tin bất kỳ

Năm 2006, Google đã công bố một bước tiến quan trọng trong quá trình xử lý kết quả tìm kiếm bằng việc thêm vào những từ đồng nghĩa với một số từ thường gặp. Google hướng tới mục tiêu là đưa ra những kết quả tra cứu tốt nhất. Vì thế, việc hệ thống thuật toán của Google hiểu được những từ ngữ được sử dụng trên thanh tra cứu là rất quan trọng. Một phần trong việc hiểu từ ngữ là hiểu được các từ đồng nghĩa. Từ đồng nghĩa là những từ nói về cùng một vật, việc nào đó, chẳng hạn như "picture" và "photos". Những người tra cứu cụm từ "sunset pictures" (ảnh hoàng hôn) thường cũng sẽ click vào những web có chứa cụm từ "sunset photos". Một vấn đề nảy sinh là những từ đó có thể có nhiều nghĩa khác nhau. Ví dụ như, từ "case" có thể mang nghĩa "trường hợp", "tình huống" hoặc "thí dụ". Nó cũng có nghĩa là "hộp" hoặc "container - thùng chứa". Cụm từ "guitar box" đồng nghĩa với cụm từ "guitar case" – hộp đàn. Nhưng "O.J Simpson box" lại không phải đồng nghĩa với "O.J Simpson case". Các tính toán của Google chỉ ra rằng từ đồng nghĩa ảnh hưởng đến 70% số người tra cứu trên hơn 100 ngôn ngữ hỗ trợ trên Google. Bên cạnh đó, còn có một số cụm tự có mức độ tương quan về nghĩa, chẳng hạn giữa “côn trùng – cào cào”, chúng ta thấy cào cào và côn trùng có một mức độ tương quan với nhau – cào cào thuộc lớp côn trùng. Nếu chỉ so sánh các thẻ với nhau mà bỏ qua mức độ tương quan, kết quả so khớp không có độ chính xác cao. Chẳng hạn, việc so khớp giữa thuộc tính “áo thun” và thuộc tính “áo thun cổ cao”, nếu chỉ so sánh thuộc tính này phải khớp từng chữ với thuộc tính kia, thì rõ ràng hai thuộc tính này hoàn toàn không khớp với nhau, “áo thun” và “áo thun cổ cao” có liên quan với nhau, cả hai thuộc tính này đều là áo thun nên sẽ xem xét là hai thuộc tính có mức độ tương quan, giúp cho việc so khớp trở nên linh hoạt và hiệu quả hơn. Cụ thể, khi tác nhân truyền vào hai thuộc tính bất kỳ, hệ thống sẽ trả về độ tương quan giữa hai thuộc tính này. ***Người dùng được tự do trong việc lựa chọn các thuộc tính để miêu tả cho nhu cầu hay “sản phẩm đáp ứng nhu cầu”, chính vì vậy chúng ta có thể thấy được việc đánh giá mức độ tương quan giữa các tag là cần thiết để hỗ trợ cho hệ thống so khớp. Bài toán kiến trúc phần mềm phải hỗ trợ cho việc kết nối và lấy giá trị mức độ tương quan của hai tag bất kỳ để cung cấp cho hệ thống so khớp.***

## Bài toán 3: Phương pháp kết hợp việc so khớp và mức độ tương quan giữa các tag

Một khi nhu cầu (NC) và “sản phẩm đáp ứng nhu cầu” (SPDUNC) được tổ chức lưu trữ dưới dạng cây các tag, thì bài toán tiếp theo là so khớp giữa cây nhu cầu và cây “sản phẩm đáp ứng yêu cầu”. NC và SPDUNC được thể hiện bằng dạng cây thông tin, người dùng được tự do định nghĩa thuộc tính cho riêng mình nên việc so khớp giữa NC và SPDUNC đòi hỏi phải kết hợp mức độ tương qua giữa các thuộc tính (hay các tag) để đánh giá mức độ so khớp một cách khách quan và linh hoạt hơn. Việc so khớp kết hợp mức độ tương quan sẽ đánh giá được SPDUNC đáp ứng được bao nhiêu phần trăm NC và SPDUNC cần cải thiện điều gì để có thể đáp ứng cao nhất NC. Mức độ so khớp giữa NC và SPDUNC sẽ được thể hiện bằng một con số phần trăm, giúp người dùng có thể dễ dàng tiếp cận thông tin và đưa ra quyết định.

## Kết quả mong muốn

* Kiến trúc dễ dàng mở rộng, đảm bảo hiệu năng và dễ sử dụng.
* Tổ chức thông tin về nhu cầu và “sản phẩm đáp ứng nhu cầu” một cách linh hoạt, có mức độ tương quan với nhau để dễ dàng so khớp.
* Định nghĩa được mức độ tương quan giữa các tag. Trong phần này, chúng tôi không đề cập đến việc làm sao để xác định chính xác độ tương quan giữa hai tag bất kỳ, chúng tôi chỉ cung cấp kiến trúc về độ tương quan, từ đó có thể mở rộng và triển khai semantic web, việc triển khai mở rộng này sẽ giúp xác định độ tương quan giữa hai tag bất kỳ một cách chính xác và khách quan hơn.
* So khớp nhu cầu và “sản phẩm đáp ứng nhu cầu” kết hợp mức độ tương quan giữa các thuộc tính, thể hiện mức độ so khớp bằng số điểm cụ thể. Dựa vào việc so khớp, kiến trúc sẽ gợi ý cải thiện “sản phẩm đáp ứng nhu cầu” đối với một nhu cầu cụ thể.
* Kết hợp khai thác dữ liệu bằng cây quyết định (decision tree), áp dụng vào việc gợi ý cải thiện “sản phẩm đáp ứng nhu cầu” cho toàn bộ những nhu cầu trong lĩnh vực cụ thể và có liên quan với “sản phẩm đáp ứng nhu cầu”. Việc chạy cây quyết định khi số lượng dữ liệu lớn mà vẫn phải đảm bảo hiệu năng hệ thống.

# Nền tảng xây dựng kiến trúc

## Xu hướng lưu trữ và tìm kiếm thông tin

Dạng phi cấu trúc (dữ liệu thô)

Dạng cấu trúc (có cấu trúc dữ liệu)

Dạng phi cấu trúc (dưới dạng tag)

Dạng cấu trúc (có cấu trúc dạng cây)

## Nhu cầu so khớp thông tin ngày càng nhiều

## Khi nào thông tin có thể so khớp được với nhau

## Tính khả thi của một kiến trúc chung cho việc lưu trữ và so khớp thông tin

## Các vấn đề cần giải quyết khi thực hiện kiến trúc lưu trữ và so khớp thông tin

## Lý thuyết và phương pháp xây dựng kiến trúc lưu trữ và so khớp thông tin

Sau thời gian nghiên cứu và tìm hiểu một số phương pháp và lý thuyết có thể giải quyết bài toán, chúng tôi sử dụng các phương pháp và lý thuyết sau đây để giải quyết bài toán kiến trúc của nhóm:

* Về việc tổ chức thông tin về nhu cầu và “sản phẩm đáp ứng nhu cầu” một cách linh hoạt, chúng tôi đã sử dụng ***Tag*** trong việc lưu trữ các thuộc tính của nhu cầu và sản phẩm đáp ứng nhu cầu, tuy nhiên, cần có sự phân cấp giữa các thuộc tính với nhau, nên chúng tôi đã kết hợp ***Tag*** và ***Taxonomy*** để lưu trữ thuộc tính dưới dạng cấu trúc cây.
* Bên cạnh đó, để giải quyết vấn đề cây có cấu trúc và có mức độ tương quan, giống nhau giữa các thuộc tính, chúng tôi đã áp dụng “***Độ tương quan giữa các tag***” một cách đơn giản là khi truyền hai từ khoá bất kỳ vào hệ thống, hệ thống sẽ trả về mức độ tương quan của hai thuộc tính đã truyền vào.
* Về việc so khớp giữa nhu cầu và “sản phẩm đáp ứng nhu cầu” chúng tôi sẽ đề cập trong phần “***Kết hợp so khớp tag và độ tương quan giữa các tag***”
* Lý thuyết về “***Cây quyết định*”** được chúng tôi áp dụng trong việc gợi ý cải thiện “sản phẩm đáp ứng nhu cầu” cho toàn bộ những nhu cầu trong lĩnh vực cụ thể và có liên quan với “sản phẩm đáp ứng nhu cầu”.

### Tag

Việc phân lớp phân cấp dữ liệu nhằm tổ chức thông tin trong hệ thống có thể được thực hiện bằng nhiều phương pháp khác nhau, một số phương pháp cụ thể như phân chia nội dung theo loại thông tin, theo cấu trúc cây (như các bài viết trong forum được đặt vào một mục cụ thể của cây danh mục). Một phương pháp khá phổ biến để tổ chức, phân loại và chia sẻ một lượng dữ liệu lớn trong cộng đồng mạng xã hội hiện nay là kỹ thuật **Tagging**, nó là *một cách đơn giản và trực quan để tổ chức nguồn thông tin, cho phép người dùng gắn các từ khóa hoặc tag (thẻ) vào thông tin hoặc đối tượng dữ liệu*; còn được biết đến với thuật ngữ social tagging, collaborative tagging, social classification, và social indexing.

“**Tag** là một từ khóa không có thứ tự hay một thuật ngữ nhằm chỉ một mẫu thông tin (như một bookmark trên internet, hình ảnh kỹ thuật số, hoặc một tệp tin máy tính). Loại dữ liệu biến đổi này giúp miêu tả một mục tin và cho phép người sử dụng tìm lại mục đó bằng cách duyệt hay dò tìm.”[[2]](#footnote-2)

Có 2 loại tagging phổ biến đó là Author-based tagging và User-based tagging. Author-based tagging được thực hiện bởi người tạo nội dung, được xem là phương pháp phân lớp theo hướng top-down. User-based tagging cho phép người dùng tự quyết định việc phân lớp nội dung mà họ quan tâm, kỹ thuật này là phương pháp phân lớp theo hướng bottom-up. Một số hệ thống có ứng dụng tagging tiêu biểu là Delicious (http://del.icio.us), Flickr (http://www.flickr.com) và các trang mạng xã hội như Facebook, Google plus…

### Taxonomy

#### Tổng quan về taxonomy

Vào những năm 90 của thế kỷ XX, khái niệm taxonomy được sử dụng trong nhiều lĩnh vực khác nhau như khoa học xã hội, tâm lý học và công nghệ thông tin… để thiết lập sử trùng hợp giữa thuật ngữ của người sử dụng và thuật ngữ của hệ thống. Các chuyên gia đầu tiên phát triển cấu trúc hệ thống web đã dùng thuật ngữ taxonomy để nói đến việc tổ chức nội dung các trang web. Kể từ đó, khái niệm taxonomy được sử dụng rộng rãi với mục đích này.

Do được sử dụng trong nhiều lĩnh vực khác nhau, nên cũng có nhiều định nghĩ về taxonomy. Từ năm 2000 đến 2005 có hơn 36 định nghĩa khác nhau, dựa theo tài liệu của Hypertext[[3]](#footnote-3) chúng tôi định nghĩa taxonomy như sau:

***Taxonomy là sự phân loại toàn bộ thông tin trong một hế thống có phân cấp, sự phân loại này theo một mối quan hệ có trước của các thực thể trong thế giới thực mà nó biểu diễn.***

Một taxonomy thường được mô tả với gốc ở trên cùng, mỗi nút của taxonomy – bao gồm cả gốc – là một thực thể thông tin đại diện cho một thực thể trong thế giới thực. Giữa các nút trong taxonomy có một mối quan hệ đặc biệt gọi là ***is subclassification of*** nếu hướng liên kết từ nút con lên nút cha hoặc là ***is superclassification of*** nếu hướng liên kết từ nút cha xuống nút con. Đôi khi những quan hệ này được xác định một cách chặt chẽ hơn là ***is subclass of*** hoặc ***is superclass of***, nếu thực thể thông tin là một lớp đối tượng.

Ví dụ sau đây mô tả một taxonomy đơn giản gồm lớp Person, lớp con của nó là Employee, Manager; Lớp cha của Person là Agent. Khi đi lên từ gốc của taxonomy, các thực thể khái quát hơn. Khi đi xuống những lá ở cuối, thực thể xác định rõ ràng hơn. Ví dụ, Agent chung chung hơn Person, Employee cụ thể hơn Person.



Hình 1. Ví dụ về một taxonomy có cấu trúc đơn giản

Taxonomy thể hiện ưu điểm trong việc phân lớp thực thể thông tin theo ngữ nghĩa, chúng thiết lập một quan hệ ngữ nghĩa đơn giản để phân biệt giữa các đối tượng trong một miền thông tin.

Taxonomy đóng vai trò rất quan trọng trong việc tổ chức thông tin và tổ chức tri thức. Nó được sử dụng chủ yếu để giúp cho việc tìm kiếm và duyệt thông tin thuận lợi và nhanh chóng hơn, đặc biệt khi chúng ta chỉ có những thông tin chung chung về vấn đề cần tìm kiếm. Khi tìm kiếm trên Internet, nếu sử dụng từ khoá để tìm kiếm thông tin, kết quả trả về có thể từ vài nghìn đến vài chục nghìn tài liệu về các chủ đề khác nhau. Sử dụng taxonomy để tìm kiếm và duyệt thông tin sẽ tiết kiệm được rất nhiều thời gian cho người dùng để tìm được thông tin cần thiết. Đồng thời, taxonomy cho phép các máy tìm kiếm và các ứng dụng có thể dễ dàng tìm được các thực thể thông tin nhanh và chính xác hơn nhiều.

Taxonomy đã được áp dụng trong nhiều bài toán khác nhau: OU Shi-yan, KHOO Christopher S.G, GOH Dion H[[4]](#footnote-4). xây dựng taxonomy hỗ trợ việc tóm tắt tự động văn bản; H.T.Kung và C.H.Wu xây dựng taxonomy cho mạng nội dung[[5]](#footnote-5), Wollersheim và Rahayu[[6]](#footnote-6) xây dựng một taxonomy hỗ trợ việc duyệt cơ sở dữ liệu về y tế.

#### Phân loại taxonomy

Có 04 loại taxonomy:

* Loại 1: Flat (phân lớp phẳng)
* Loại 2: Hierarchical (cấu trúc dạng cây)
* Loại 3: Faceted
* Loại 4: Network

##### Flat taxonomy

Flat taxonomy gom nhóm thông tin vào một tập hợp danh mục sẵn có, không có quan hệ thuộc về giữa các mục, chúng ngang hàng với nhau. Flat taxonomy thường được ứng dụng trong việc phân nhóm sản phẩm theo alphabe.



Hình 2. Ví dụ về flat taxonomy

##### Hierarchical taxonomy

Hierarchical taxonomy được thể hiện dưới dạng cấu trúc cây, mỗi node chỉ có một node cha và không giới hạn số node con. Ứng dụng tiêu biểu của Hierarchical taxonomy là cây phân cấp các chủ đề trong forum.



Hình 3. Ví dụ về hierarchical taxonomy

##### Faceted taxonomy

Faceted taxonomy thể hiện sự phân lớp theo cấu trúc dữ liệu ngôi sao. Mỗi node trong cấu trúc này là một điểm trung tâm, có thể liên kết đến những node khác trong cấu trúc ngôi sao khác. Cũng giống như flat taxonomy, không có quan hệ thuộc về, hay quan hệ cha con giữa các mục trong faceted taxonomy. Tất cả các mục trong faceted taxonomy liên kết đến một đối tượng duy nhất – mỗi node mô tả thuộc tính hoặc giá trị hay một khía cạnh của vấn đề. Ví dụ như sách điện tử, mỗi node mô tả một khía cạnh về cuốn sách – tác giả, tên sách, ngày xuất bản…



Hình 4. Ví dụ về faceted taxonomy

##### Network taxonomy

Network taxonomy là một cấu trúc mạng, mỗi node có thể có nhiều node cha. Mỗi node trong network taxonomy có thể liên kết đến một node bất kỳ. Với network taxonomy thông tin được tổ chức theo cả hai dạng cấu trúc cây và dạng liên kết, các quan hệ giữa các node có nhiều ý nghĩa khác nhau. Network taxonomy có thể ứng dụng trong hệ thống quản lý kiến thức như công cụ gợi ý, semantic network (mạng ngữ nghĩa).



Hình 5. Ví dụ về network taxonomy

### Độ tương quan giữa các tag

Nếu so sánh với các phương pháp phân loại dữ liệu trước đây (categories/ folder), tag là cách đơn giản, linh hoạt và là một công cụ phân loại mạnh mẽ. Tuy nhiên, *tag có một số khuyết điểm đáng kể đó là chúng không thể cung cấp thông tin về ngữ nghĩa của chúng*, ngữ nghĩa của các tag có thể được bao gồm: Tính nhiều nghĩa (cùng một từ có thể tham khảo các khái niệm khác nhau), tính đồng nghĩa (khái niệm tương tự có thể được chỉ ra bằng cách sử dụng từ ngữ khác nhau), các hình thức từ vựng khác nhau (hình thức danh từ khác nhau, chia động từ khác nhau, từ viết tắt, ngôn ngữ khác nhau), lỗi chính tả là một số vấn đề mà phát sinh khi sử dụng các tag. Ví dụ, tag ‘orange’ có thể liên quan đến trái cây hay màu sắc, và tag ‘movie’ và ‘film’ thường được mô tả cùng một khái niệm. Thiếu sự phân biệt ngữ nghĩa này dẫn đến các kết nối không phù hợp giữa các mục, làm cho chúng khó tìm kiếm và duyệt.

*Như vậy, giữa hai tag khác nhau sẽ tồn tại một mối quan hệ nhất định. Đó là một con số thể hiện mức độ tương quan giữa tag A và tag B, thường có giá trị từ -1 đến 1, hoặc giữa 0 và 1, trong đó giá trị 1 biểu hiện tính tương đồng rất cao, và giá trị 0 có nghĩa ít hoặc không có sự liên hệ nào.*

### Kết hợp so khớp tag và độ tương quan giữa các tag

Một khi thông tin về nhu cầu (NC) và “sản phẩm đáp ứng nhu cầu” (SPDUNC) đã được tổ chức linh hoạt, thì công việc tiếp theo là xác định mối liên hệ giữa nhu cầu và “sản phẩm đáp ứng nhu cầu”, việc **so khớp** giữa NC và SPDUNC nhằm giải quyết được vấn đề này.

Kết hợp **so khớp** và **độ tương quan giữa các tag** nhằm đảm bảo kết quả so khớp được chính xác và khách quan hơn. Nếu chỉ so sánh văn bản hay các giá trị thuộc tính theo kiểu so sánh object (so sánh A phải khớp hoàn toàn với B) thì việc so khớp sẽ có nhiều thiếu thiếu sót trong việc so sánh giữa NC và SPDUNC. Thiếu sót này thể hiện ở việc so sánh các tag có mức độ tương quan như đã được đề cập ở phần “***3.1.3******Độ tương quan giữa các tag***”.

Kết hợp **so khớp** và **độ tương quan giữa các tag** còn giúp cho việc gợi ý cải thiện SPDUNC (sẽ được đề cập trong phần ***4.1 Áp dụng lý thuyết và phương pháp vào bài toán***)

### Cây quyết định

Tại sao sử dụng cây

#### Giới thiệu về cây quyết định

* Cây quyết định là một cây phân cấp có cấu trúc.
* Dùng để phân lớp đối tượng dựa vào dãy các luật (series of rules), các luật này được sinh ra từ tập dữ liệu (training set).
* Các thuộc tính phân lớp thường có kiểu dữ liệu là binary, nominal, ordinal, continuos.
* Ví dụ “Phương tiện di chuyển”: Cho tập dữ liệu (training set) như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Thuộc tính (Attribute) | | | | Thuộc tính phân lớp (Class) |
| Gender | Car ownership | Travel cost ($/km) | Income level | Transportation mode |
| Male | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Female | 1 | Cheap | Medium | Train |
| Female | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Male | 0 | Standard | Medium | Train |
| Female | 1 | Standard | Medium | Train |
| Female | 1 | Expensive | High | Car |
| Male | 2 | Expensive | Medium | Car |
| Female | 2 | Expensive | High | Car |

Bảng 1. Ví dụ bảng dữ liệu lựa chọn phương tiện di chuyển

* Tập dữ liệu trên mô tả 10 đối tượng, mỗi đối tượng được miêu tả bằng 4 thuộc tính là “Gender” (kiểu dữ liệu binary), “Car ownership” (quantitative interger), “Travel cost/km” (ordinal), “Income level” (ordinal) và thuộc tính phân loại – category attribute – “Transportation mode” (ordinal).
* Từ tập dữ liệu trên, chúng ta có thể tạo ra cây quyết định như sau:



Hình 6. Ví dụ về cây quyết định

* Trong cây quyết định trên, thuộc tính “Income level” không xuất hiện trong cây; vì dựa vào Bảng 1 (trang 8), thuộc tính “Travel cost/Km” sẽ sinh ra cây quyết định dùng để phân lớp tốt hơn “Income level”

#### Sử dụng cây quyết định để dự đoán lớp các dữ liệu chưa biết

* Mục đích chính của cây quyết định là dùng để xác định lớp hay nói khác đi là dự đoán lớp của các dữ liệu chưa biết dựa vào cây quyết định được sinh ra từ tập dữ liệu đào tạo (training data)
* Ví dụ: dựa vào ví dụ ở phần 3.1.1 Giới thiệu về cây quyết định (trang 8). Cho tập dữ liệu cần dự đoán sau đây:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Gender | Car ownership | Travel cost ($/km) | Income level | Transportation mode |
| Ngoc | Female | 1 | Cheap | High | ? |
| Hieu | Male | 0 | Standard | High | ? |
| Phuc | Male | 1 | Cheap | Medium | ? |
| Nguyen | Male | 2 | Expensive | High | ? |

Bảng 2. Ví dụ về bảng dữ liệu cần dự đoán phương tiện di chuyển

* Dựa vào cây quyết định (Hình 1, trang 9), cây quyết định sẽ được duyệt từ nút gốc “Travel cost/km”, dãy các luật sau sẽ được sinh ra:
* Nếu “Travel cost/Km” là **Expensive** thì người đó sẽ chọn phương tiện di chuyển là **car**.
* Nếu “Travel cost/Km” là **Standard** thì người đó sẽ chọn phương tiện di chuyển là **train**
* Nếu “Travel cost/Km” là **Cheap**, chúng ta sẽ xem xét thuộc tính “Gender”:
  + Nếu “Gender” là **Male**, người đó sẽ chọn phương tiện là **bus**
  + Nếu “Gender là **Female**, thì xem người đó sở hữu bao nhiêu xe hơi (thuộc tính “Car ownership”). Nếu số xe sở hữu là **0**, thì người đó sẽ chọn phương tiện di chuyển là **bus**; ngược lại, nếu số xe sở hữu lớn hơn hay bằng **1**, thì người đó sẽ chọn phương tiện di chuyển là **train**.
* Bảng 2 sẽ được dự đoán như sau:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Gender | Car ownership | Travel cost ($/km) | Income level | Transportation mode |
| Ngoc | Female | 1 | Cheap | High | Train |
| Hieu | Male | 0 | Standard | High | Train |
| Phuc | Male | 1 | Cheap | Medium | Bus |
| Nguyen | Male | 2 | Expensive | High | Car |

Bảng 3. Bảng dữ liệu kết quả dự đoán phương tiện di chuyển

* **Một số chú ý khi sử dụng cây quyết định:**
* Phụ thuộc rất nhiều vào training data, tập dữ liệu training data càng lớn thì cây quyết định sẽ đáng tin cậy hơn.
* Không thể nói cây quyết định được sinh ra từ cây quyết định trên là tập luật tốt nhất
* Có nhiều thuật toán phân lớp: ID3, J48, C4.5, C5, CART (Classification and Regression Tree), … Việc lựa chọn thuật toán phụ thuốc vào rất nhiều yếu tố, trong đó yếu tố cấu trúc dữ liệu ảnh hưởng rất nhiều đến kết quả của thuật toán. Chẳn hạn, thuật toán ID3 và CART hiệu quả cho việc phân lớp đối với các dữ liệu số (quantitative value), trong khi đó, thuật toán J48, C4.5 có hiệu quả hơn đối với dữ liệu Qualititive value (ordinal, binary, nominal)

#### Một số độ đo thông dụng

* Cho bảng dữ liệu bao gồm các thuộc tính và thuộc tính phân lớp, chúng ta có thể đo được tính đồng nhất hay không đồng nhất thông qua thuộc tính phân lớp. Bảng dữ liệu có tính đồng nhất nếu nó có duy nhất một phân lớp. Ngược lại, nếu nó có nhiều phân lớp khác nhau, thì bảng dữ liệu có tính không đồng nhất hay tính pha trộn. Chúng ta có thể đo được mức độ pha trộn (Impurity Degree); Entropy, độ đo Gini và classification error là những cách tính mức độ pha trộn thông dụng nhất.

*Pj: xác suất xảy ra phân lớp j*

* Ví dụ: Chúng ta cùng nhìn lại Bảng 1. Ví dụ bảng dữ liệu lựa chọn phương tiện di chuyển. “Transportation mode” có 3 nhóm Bus, Car và Train. Bảng 1 có 10 dòng dữ liệu, trong đó “Transportation mode” có 4 buses, 3 cars, 3 trains (4B, 3C, 3T).
* Xác suất để xảy ra cho từng phân lớp là:

##### Entropy

Khi bảng dữ liệu có duy nhất 1 phân lớp thì Entropy sẽ bằng 0 vì xác suất là 1 và . Entropy sẽ đạt giá trị cực đại khi tất cả thuộc tính phân lớp có xác suất xảy ra bằng nhau. Biểu đồ dưới đây thể hiện giá trị cực đại của Entropy sẽ thay đổi phụ thuộc vào số lượng thuộc tính phân lớp n, trong trường hợp xác xuất tất cả thuộc tính phân lớp thì . Giá trị của Entropy sẽ lớn hơn 1 khi số lượng thuộc tính phân lớp nhiều hơn 2



Biểu đồ 1. Mối liên hệ giữa giá trị Entropy cực đại và số lượng phân lớp

##### Gini index

* Một cách khác để tính “Impurity degree”.
* Chúng ta cùng quay lại ví dụ ở trên
* Khi bảng dữ liệu có duy nhất 1 phân lớp thì Gini index sẽ bằng 0 vì xác suất bằng 1 và . Cũng giống như Entropy, Gini index sẽ đạt giá trí cực đại khi tất cả thuộc tính phân lớp có xác suất xảy ra bằng nhau. Biểu đồ dưới đây thể hiển giá trị cực đại của Gini index sẽ khác nhau phụ thuộc vào số lượng phân lớp n, khi xác xuất tất cả thuộc tính phân lớp



Biểu đồ 2. Mối liên hệ giữa giá trị Gini index cực đại và số lượng phân lớp

##### Classification error

* Trong ví dụ trên
* Cũng giống như Entropy và Gini index, Classification error sẽ bằng 0 khi bảng dữ liệu có duy nhất 1 phân lớp vì xác suất bằng 1 và .

#### Nguyên tắc hoạt động của một số thuật toán xây dựng cây quyết định thường dùng

* Các thuật toán xây dựng cây quyết định thường được sử dụng nhất là ID3, C4.5 và CART (classification and regression trees).
* Nhìn chung, các thuật toán xây dựng cây quyết định đều được xây dựng trên quy tắc đệ quy. Ví dụ, Hunt là thuật toán dựng cây quyết định, đệ quy theo nút của cây, bắt đầu từ nút gốc. Mặc dù kết quả đạt được từ thuật toán Hunt không được tối ưu, nhưng đây là một trong những một trong những thuật toán dùng để xây dựng cây quyết định sớm nhất.

##### Lần lặp đầu tiên

* Giả sử, chúng ta có một bảng dữ liệu chứa các thuộc tính và thuộc tính phân lớp, tạm gọi bảng này là D. Từ bảng D, chúng ta sẽ lấy ra từng cột thuộc tính trong bảng để đối chiếu với các giá trị của thuộc tính phân lớp. Nếu chúng ta có p cột dữ liệu, và lấy từng phần tử p là tập con của D, tạm gọi từng tập hợp này là Si. Bảng D là tập hợp những Si và thuộc tính phân lớp.



Hình 7. Phân chia bảng D thành những tập con Si

* Sau khi phân chia bảng dữ liệu D như trên, chúng ta sẽ tính mức độ pha trộn (tham khảo cách tính tại phần 3.1.3 Một số độ đo thông dụng)
* Ví dụ, dựa vào Bảng 1. Ví dụ bảng dữ liệu lựa chọn phương tiện di chuyển. chúng ta có thể tính “Impurity degree” dựa vào thuộc tính phân lớp “Transportation mode”. “Transportation mode” có 4 busses, 3 cars và 3 trains (4B, 3C, 3T)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Thuộc tính (Attribute) | | | | Thuộc tính phân lớp (Class) |
| Gender | Car ownership | Travel cost ($/km) | Income level | Transportation mode |
| Male | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Female | 1 | Cheap | Medium | Train |
| Female | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Male | 0 | Standard | Medium | Train |
| Female | 1 | Standard | Medium | Train |
| Female | 1 | Expensive | High | Car |
| Male | 2 | Expensive | Medium | Car |
| Female | 2 | Expensive | High | Car |

4B, 3C, 3T

Entropy 1.571

Gini index 0.660

Classification index 0.600

* Sau khi phân chia bảng D thành từng tập con Si, chúng ta tiếp tục phân chia Si để tính “Impurity degree” cho từng giá trị thuộc tính trong từng tập Si



Hình 8. Phân chia Si dựa vào phân lớp

* Tiếp tục ví dụ trên, thuộc tính “Travel cost/km” có 3 giá trị, đó là “Cheap”, “Standard” và “Expensive”.



Hình 9. Độ lợi thông tin thuộc tính Travel cost/Km

* **Độ lợi thông tin (Information Gain):**
  + Có sự khác nhau giữa các cách tính mức độ pha trộn “Impurity degree” giữa bảng D và các tập con Si, chúng ta sẽ tiến hành so sánh “Impurity degree” trước và sau khi phân chia bảng thành những tập con (ví dụ như phân chia D thành từng tập con Si). Information gain là một phương pháp để đo lường sự khác nhau này. Chúng ta sẽ so sánh lợi ích khi chia bảng dữ liệu theo các giá trị của thuộc tính, từ đó chọn ra thuộc tính tối ưu để phân chia
* Xét ví dụ trên, bảng D có 3 phân lớp 4B, 3C và 3T và có Entropy là 1.571. Bây giờ chúng ta sẽ sử dụng thuộc tính “Travel cost/km” để chia bảng thành 3 phần:
* “Travel cost/km” có giá trị là Cheap, thuộc tính phân lớp có 4B và 1T

* “Travel cost/km” có giá trị là Standard, thuộc tính phân lớp có 2T, vì lúc này chỉ có duy nhất 1 thuộc tính phân lớp)
* “Travel cost/km” có giá trị là Expensive, thuộc tính phân lớp có 3C, vì lúc này chỉ có duy nhất 1 thuộc tính phân lớp)
* Tương tự chúng ta sẽ tính được độ đo Gini index và Classification error:

|  |  |
| --- | --- |
| **Lợi ích khi chia theo thuộc tính “Travel cost/km” dựa vào** | |
| Entropy | 1.210 |
| Gini index | 0.500 |
| Classification error | 0.500 |

Bảng 4. Lợi ích khi chia bảng D theo thuộc tính “Travel cost/km”

* Chúng ta thực hiện lần lượt cho các thuộc tính còn lại của bảng D: “Gender”, “Car ownership” và “Income level”



Hình 10. Độ lợi thông tin các thuộc tính còn lại trong Bảng 1

* Bảng dưới đây sẽ cho chúng ta thấy độ lợi thông tin cho tất cả 4 thuộc tính trong bảng D. Chúng ta không cần tính “Impurity degree” dựa trên cả 3 độ đo Entropy, Gini index và Classification error, chỉ cần chọn lựa 1 trong 3 độ đo trên.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kết quả của lần lặp đầu tiên** | | | | |
| Độ lợi | Gender | Car ownership | Travel cost/Km | Income level |
| Entropy | 0.125 | 0.534 | **1.210** | 0.695 |
| Gini index | 0.060 | 0.207 | **0.500** | 0.293 |
| Classification error | 0.100 | 0.200 | **0.500** | 0.300 |

Bảng 5. Kết quả độ lợi thông tin sau khi phân chia bảng D theo từng thuộc tính

* Sau khi tính được độ lợi thông tin cho từng thuộc tính, chúng ta sẽ lựa chọn thuộc tính có độ lợi thông tin cao nhất để làm node gốc cho cây quyết định. Bảng trên cho thấy, độ lợi thông tin khi phân lớp theo thuộc tính “Travel cost/km” là cao nhất nên chúng ta sẽ chọn “Travel cost/km” là node gốc của cây.



Hình 11. Node gốc của cây quyết định sau lần lặp đầu tiên

* Bảng D sẽ được phân chia theo thuộc tính “Travel cost/km” như sau:



Hình 12. Bảng D được phân chia sau lần lặp đầu tiên



Hình 13. Cây quyết định sau lần lặp đầu tiên

##### Những lần phân lớp tiếp theo

Trong phần này sẽ sử dụng lại ví dụ ở phần 3.1.4.1

###### Lần phân lớp thứ 2

* Sau lần lặp đầu tiên, chúng ta cần cập nhật lại bảng dữ liệu. Khi chúng ta phân chia bảng D theo node gốc “Travel cost/km”, những dòng dữ liệu có giá trị thuộc tính “Travel cost/km” là Expensive và Standard đã là phân lớp thuần khiết, chúng ta không cần sử dụng lại những dòng dữ liệu này để phân lớp. Trong lần lặp này, “Travel cost/km” chỉ còn lại giá trị Cheap, chúng ta sẽ loại bỏ thuộc tính này



Hình 14. Dữ liệu cho lần phân lớp thứ 2

* Tính “Impurity degree” cho bảng chính



Hình 15. Tính Impurity degree cho lần phân lớp thứ 2

* Tính “Impurity degree” cho các thuộc tính của bảng chính



Hình 16. Tính Impurity degree các thuộc tính cho lần phân lớp thứ 2

* Độ lợi thông tin khi phân lớp theo thuộc tính “Gender” lớn nhất. Chúng ta sẽ phân chia bảng dữ liệu như sau:



Hình 17. Bảng dữ liệu sau khi chia theo phân lớp Gender

* Cây quyết định sau lần phân lớp thứ 2:



Hình 18. Cây quyết định sau lần phân lớp thứ 2

###### Lần phân lớp thứ 3

Tương tự sau lần phân lớp thứ 2, bảng dữ liệu cho lần phân lớp thứ 3 như sau:



Bảng 6. Bảng dữ liệu cho lần phân lớp thứ 3

Nếu chia bảng trên theo “Car ownership” hay “Income level” thì bảng dữ liệu được chia đều có duy nhất 1 phân lớp. Chúng ta có thể chia bảng dữ liệu theo 1 trong 2 thuộc tính này.



Hình 19. Cây quyết định đầy đủ sau 3 lần phân lớp

#### Đánh giá cây quyết định trong lĩnh vực khai thác dữ liệu

##### Điểm mạnh

* **Cây quyết định sinh ra các quy tắc hiểu được**

Cây quyết định có thể sinh ra các quy tắc được chuyển đổi sang tiếng Anh hay câu lệnh SQL. Thậm chí đối với những tập dữ liệu lớn làm cho hình dáng cây quyết định lớn và phức tạp, việc duyệt cây cũng rất dễ dàng. Bất cứ một sự phân lớp hay dự đoán nào đều tương đối minh bạch. Đây là ưu điểm nổi bật của cây quyết định.

* **Cây quyết định có thể thực thi trong những lĩnh vực hướng quy tắc**

Cây quyết định là sự lựa chọn hoàn hảo cho những lĩnh vực có quy tắc, từ lĩnh vực di truyền đến các những quá trình công nghiệp chứa các quy tắc ẩn, không rõ ràng (underlying rules), dữ liệu lỗi chưa được tiền xử lý khá phức tạp và tối nghĩa. Cây quyết định là một lựa chọn tối ưu khi cần tìm ra những quy tắc ẩn, không rõ ràng trong kho dữ liệu (data warehouse)

* **Dễ dàng tính toán trong khi phân lớp**

Những thuật toán xây dựng cây quyết định thường tạo ra cây với số phân nhánh thấp và kiểm tra đơn giản tại từng node. Những thuật toán này thường kiểm tra bằng cách so sánh số, xem xét phần tử của một tập hợp hay các phép nối đơn giản, những thao tác này sẽ được chuyển thành các toán hàm logic và số nguyên, đây là những toán hạng thực thi nhanh và chi phí tối ưu. Trong môi trường thương mại, các mô hình dự đoán này được sử dụng để phân lớp hàng triệu, thậm chí hàng tỷ bản ghi, bởi vậy có thể nói đây là một ưu điểm quan trọng của cây quyết định.

* **Cây quyết định xử lý với cả thuộc tính liên tục và thuộc tính rời rạc**

Các thuộc tính liên tục hay rời rạc đều có thể xử lý bằng cây quyết định. Tuy nhiên, thuộc tính liên tục cần nhiều tài nguyên tính toán hơn, được phân chia bằng việc chọn ra một ngưỡng trong tập các giá trị đã được sắp xếp của thuộc tính đó.

* **Thể hiện rõ ràng những thuộc tính quan trọng nhất cho việc dự đoán phân lớp**

Các thuật toán xây dựng cây quyết định chỉ ra những thuộc tính dùng để phân chia tốt nhất tập dữ liệu đào tạo bắt đầu từ node gốc của cây. Chính vì vậy, chúng ta có thể thấy được thuộc tính nào là quan trọng, có mức độ phụ thuộc lớn cho việc dự đoán hay phân lớp

##### Điểm yếu

Mặc dù có những điểm mạnh nổi bật trên, cây quyết định vẫn không tránh khỏi những điểm yếu. Cây quyết định không thích hợp lắm với những bài toán mục tiêu, như là dự đoán giá trị của thuộc tính liên tục: thu nhập, huyết áp, lãi suất ngân hàng… Bên cạnh đó, cây quyết định cũng khó giải quyết với những dữ liệu thời gian liên tục.

* **Cây quyết định dễ xảy ra lỗi khi có nhiều phân lớp**

Một số thuật toán chỉ tao tác với những lớp giá trị nhị phân dạng “có/không” hay “đồng ý/từ chối”. Số khác có thể chỉ định các bản ghi vào một số lớp bất kỳ nhưng dễ xảy ra lỗi khi ứng với một phân lớp có số lượng dữ liệu đào tạo nhỏ.

* **Chi phí tính toán để đào tạo cao**

Điểm này nghe có vẻ mâu thuẩn với điểm mạnh phía trên. Quá trình xây dựng cây quyết định khác đắt về mặt tính toán. Điều này cũng dễ hiểu vì *cây quyết định có nhiều node trong trước khi đi đến node lá cuối cùng*, ứng với từng node, chúng ta cần tính một độ đo (hay tiêu chuẩn phân chia) trên từng thuộc tính. Bên cạnh đó, đối với thuộc tính liên tục, chúng ta còn phải thêm thao tác sắp xếp lại dữ liệu theo thứ tự giá trị của thuộc tính đó. Quá trình này chọn ra những thuộc tính để phân lớp tốt nhất. Một số thuật toán còn sử dụng tổ hợp các thuộc tính kết hợp với nhau có trọng số để phát triển cây. Quá trình cắt tỉa cây chi phí cũng khá cao vì trong quá trình cắt tỉa, nhiều cây con sẽ được tạo ra và so sánh.

## Xây dựng kiến trúc dễ dàng mở rộng, đảm bảo hiệu năng và dễ sử dụng

Để xây dựng kiến trúc đáp ứng được nhu cầu nói trên, nhóm chúng tôi áp dụng một số lý thuyết về kiến trúc phần mềm sau đây:

### Một số lý thuyết cơ bản OOP Design

### 4+1 Architecture View Model



Hình 20. 4+1 Architecture View Model

4+1 View Model là mô hình lập trình được thiết kế bởi Philippe Kruchten. Mô hình cho phép các stakeholders khác nhau (những cá nhân có liên quan đến dự án phần mềm) có thể tìm thấy cái họ cần trong kiến trúc phần mềm thông qua các biểu đồ theo các View khác nhau.

* **Scenarios (Use Case view):** thể hiện các vấn đề và các giải pháp liên quan đến chức năng tổng quát của hệ thống.

*Đối tượng*: tất cả các stakeholders của hệ thống, bao gồm cả người dùng cuối.

*Mô tả*: diễn tả các chức năng quan trọng, trung tâm của hệ thống. Mô tả những actors và các user cases của hệ thống.

* **Logical view:** thể hiện các vấn đề liên quan đến cấu trúc thiết kế của hệ thống, mô tả thiết kế tổ chức của hệ thống thành các hệ thống con, thành phần, lớp, giao diện, và làm thế nào mà các yếu tố này kết hợp với nhau để tạo ra các chức năng đã miêu tả trong phần Use-Case View. Ngoài ra, Logical View cũng rất quan tâm đến những chức năng mà hệ thống cung cấp cho người dùng, thể hiện qua lược đồ Class, lược đồ tuần tự.

*Đối tượng:* Designers.

*Mô tả:* miêu tả phương thức mà các chức năng của hệ thống sẽ được cung cấp. Ngược lại với use case view, Logical view nhìn vào phía bên trong hệ thống; miêu tả các lớp,các đối tượng, quan hệ.

* **Process view:** thể hiện các vấn đề liên quan đến việc xử lý giao tiếp và đồng bộ trong hệ thống. Process View quan tâm đến khía cạnh động của hệ thống, giải thích về các quy trình hoạt động của hệ thống và cách mà các thành phần giao tiếp với nhau, tập trung vào các hành vi của hệ thống vào thời điểm hoạt động. Nó minh họa quá trình phân hủy hệ thống chính thành biểu đồ của các lớp và các hệ thống con bởi một luồng thực thi tác vụ, một tiến trình hoạt động của chức năng hệ thống. Process View được thể hiện chi tiết ở lược đồ hoạt động

*Đối tượng*: Integrators.

*Mô tả*: chia hệ thống thành các tiến trình (process) có thể được thực thi song song, hướng nhìn này cũng phải quan tâm đến vấn đề giao tiếp và đồng bộ hóa các tiến trình đó.

* **Deployment view**: thể hiện các vấn đề liên quan đến việc triển khai hệ thống. Deployment View mô tả cách mà các tiến trình thực thi tác vụ được cấp phát cho phần cứng bên dưới hệ thống, môi trường thực thi, và nó còn thể hiện con đường truyền thông giao tiếp giữa các phần cứng với nhau.

*Đối tượng*: Deployment managers.

*Mô tả*: khai chỉ cho chúng ta sơ đồ triển khai về mặt vật lý của hệ thống, ví dụ như các máy tính cũng như các máy móc và sự liên kết giữa chúng với nhau. Hướng nhìn này cũng bao gồm sự ánh xạ các thành phần của hệ thống vào cấu trúc vật lý.

* **Implementation view**: thể hiện các vấn đề liên quan đến việc tổ chức các thành phần trong hệ thống. Implementation view mô tả các thành phần của hệ thống đã được ta nhận ra bởi quan điểm Logical View, thể hiện sự phụ thuộc giữa các thành phần này bên trong hệ thống.

*Đối tượng*: Programmers.

*Mô tả*: chỉ ra khía cạnh tổ chức của các thành phần code, mô tả các lớp và các hệ thống con của ứng dụng.

### *Related Artifacts:* Implementation model, components Framework Design Guidelines

Designing for Extensibility

Pattern for Reusable Library

### Design Patterns

Design Pattern là những thiết kế đã được sử dụng và được đánh giá tốt giúp giải quyết những vấn đề thiết kế thường gặp, Design pattern chú trọng việc giúp cho bản thiết kế có tính uyển chuyển, dễ nâng cấp, thay đổi.

#### Vai trò của design pattern:

* Cung cấp phương pháp giải quyết những vấn đề thường gặp trong thực tế đã được đánh giá, kiểm nghiệm.
* Là biện pháp tái sử dụng tri thức các chuyên gia phần mềm.
* Hình thành kho tri thức, ngữ vựng trong giao tiếp giữa những người làm phần mềm
* Giúp người tìm hiểu về design pattern nắm vững hơn đặc điểm của ngôn ngữ lập trình, nhất là lập trình hướng đối tượng.

#### Khả năng ứng dụng của design pattern

* Tìm kiếm đối tượng: Chúng ta thường gặp khó khăn trong việc phân chia hệ thống thành một tập hợp các đối tượng hoạt động hiệu quả, việc sử dụng các mẫu thiết kế giúp đưa ra những đối tượng thường gặp trong những trường hợp thiết kế tương tự đã gặp trước đây.
* Xác định số lượng và kích thước đối tượng: trong trường hợp hệ thống cần ràng buộc số lượng xác định đối tượng đang hoạt động hoặc người thiết kế đang băn khăn về việc nên tập trung một số chức năng nào đó vào trong một đối tượng hay tách ra thành nhiều đối tượng.
* Xác định interface và hiện thực (implement) của đối tượng. “Program to an interface, not an implementation”: lập trình bắt đầu từ interface sau đó hiện thực chúng.
* Giúp thiết kế theo hướng tái sử dụng và linh động bằng cách sử dụng mối quan hệ giữa các đối tượng (tính bao đóng, tính thừa kế…) một cách phù hợp và thiết kế theo hướng tiên đoán trước các thay đổi trong tương lai

#### Phân loại Object Oriented Design Patterns

* Mỗi nhóm design pattern có các pattern về lớp (class patterns) và pattern về đối tượng (object patterns)
* **Class patterns** dựa trên mối quan hệ thừa kế giữa các lớp, mối quan hệ này là tĩnh (xác định tại thời điểm dịch), do đó class patterns thích hợp cho hệ thống không cần thay đổi động trong thời gian chạy
* **Object patterns** dựa trên mối quan hệ giữa các đối tượng, do đó có thể thay đổi ở thời điểm chạy

#### Design patterns có ba nhóm chính:

* **Structural**: Cung cấp cơ chế xử lý những lớp không thể thay đổi (lớp thư viện của third party…), ràng buộc muộn (lower coupling) và cung cấp các cơ chế khác để kế thừa
* **Creational**: Khắc phục các vấn đề khởi tạo đối tượng, hạn chế sự phụ thuộc vào nền tảng (platform).
* **Behavioral**: Che dấu hiện thực của đối tượng, che dấu giải thuật, hỗ trợ việc thay đổi cấu hình đối tượng một cách linh động.

#### Nhóm Structural patterns

* Nhóm này tập trung giải quyết các vấn đề về kết hợp các lớp, các đối tượng thành một kiến trúc lớn hơn
* Các mẫu cấu trúc lớp (structural class patterns) sử dụng kế thừa nhằm kết hợp các class hay các interface với nhau. Tương tự quá trình đa thừa kế, một lớp thừa kế từ nhiều lớp cha hay interface sẽ mang đặc điểm của tất cả các lớp cha hay interface gộp lại.
* Các mẫu cấu trúc đối tượng (structural object patterns) tập trung vào việc kết hợp các đối tượng để thực hiện một chức năng cụ thể
* Structural patterns bao gồm: Adapter, composite, proxy, decorator, facade và flyweight

#### Nhóm Creational patterns

* Creational design patterns giúp xây dựng hệ thống linh động về mặt khởi tạo, quản lý và sử dụng các đối tượng. Chúng cho phép hệ thống chủ động trong việc xác định đối tượng nào được tạo ra, ai tạo ra đối tượng, cách thức và thời điểm khởi tạo.
* Đặc điểm nổi bật trong creationnal patterns là chương trình cần sử dụng đối tượng không trực tiếp sinh ra đối tượng mà nhờ các phần tử trung gia để tăng độ linh động
* Class creational patterns sử dụng đặc điểm thừa kế để thay đổi class sẽ được sử dụng để sinh ra đối tượng.
* Object creational patterns truyền quá trình khởi tạo đối tượng cho một đối tượng khác.
* Creational patterns bao gồm Abstract Factory, Factory Method, Prototype, Singleton và Builder.

#### Nhóm Behavioral patterns

* Nhóm behavioral tập trung vào giải thuật và sự phân bố công việc giữa các object.
* Behavioral class patterns sử dụng kế thừa để chuyển giao thao tác giữa các class.
* Object patterns sử dụng tính đa hình và bao đóng để truyền thao tác từ đối tượng này sang đối tượng khác.
* Behavioral patterns bao gồm Chain of Responsibility, Template method, Strategy, Command, State, Observer

## Lưu trữ thông tin dưới dạng cây là tiền đề cho việc lưu trữ thông tin của kiến trúc

# Giải quyết bài toán kiến trúc

## Áp dụng lý thuyết và phương pháp vào bài toán

Lý thuyết và phương pháp ở trên đã được chúng tôi áp dụng vào việc giải quyết bài toán kiến trúc được đề cập ở phần “***2. Giới thiệu bài toán kiến trúc***”.

Tổ chức thông tin và sự linh hoạt của hệ thống so khớp được giải quyết bằng việc lưu trữ thông tin về nhu cầu và “sản phẩm đáp ứng nhu cầu” dưới dạng **tag** kết hợp **taxonomy**. Đồng thời, **taxonomy** và **cây quyết định** sẽ gợi ý cho actor xây dựng cây thông tin nhu cầu và “sản phẩm đáp ứng nhu cầu” một cách nhanh chóng và hiệu quả. Bài toán 2 được giải quyết bằng việc lưu trữ “độ tương quan giữa các tag” thông qua bảng **SimilarTerm**. So khớp kết hợp mức độ tương quan giữa các tag nhằm giải quyết bài toán kiến trúc còn lại.

### Áp dụng tag và taxonomy

Với mục tiêu xây dựng một kiến trúc linh hoạt trong việc tổ chức, lưu trữ thông tin tạo khả năng tìm kiếm và so khớp thông tin đa dạng trong các lĩnh vực khác nhau, nhóm chúng tôi đã nghiên cứu *Tag* và *Taxonomy* để giải quyết Bài toán 1: Tổ chức thông tin và sự linh hoạt của hệ thống so khớp.



Hình 21. Mô hình tổ chức thông tin theo tag kết hợp taxonomy

Đầu tiên, ta có một bảng **Object** lưu các thông tin của đối tượng, ví dụ một sản phẩm với các thông tin cơ bản như tên sản phẩm và mô tả. Và kiến trúc hỗ trợ actor gắn các từ khóa (TagName) vào đối tượng, và các tập dữ liệu này được lưu vào bảng **TagAttribute** liên kết với đối tượng thông tin qua thuộc tính “ObjectId”. Đồng thời chính TagAttribute cũng phân nhóm dữ liệu theo cấu trúc cây với thuộc tính ParentId – TagId, tức là mỗi Tag có thể thuộc về một Tag gốc đã được định nghĩa sẵn. Ví dụ sản phẩm A được gán tag “Red” và tag cha là “Color”, hay tag “Điện gia dụng” thuộc tag “Loại sản phẩm”. Vậy như ta thấy kiến trúc đảm bảo tính linh hoạt , cho phép actor thực hiện tagging dễ dàng, mà vẫn có thể quản lý được tập thuộc tính của đối tượng.



Hình 22. Ví dụ tổ chức thông tin theo tag kết hợp taxonomy

Kiến trúc còn hỗ trợ actor quy định trọng số (Weight) cho mỗi tag nhằm thể hiện mức độ quan trọng giữa các tiêu chí yêu cầu khi thực hiện so khớp với đối tượng “sản phẩm đáp ứng nhu cầu”. Thuộc tính “Required” cho phép actor quy định tag này có phải là yêu cầu bắt buộc hay không. Và thông qua thuộc tính “TableReference” hệ thống có thể phân biệt các tag thuộc những loại đối tượng riêng biệt như yêu cầu tuyển dụng và hồ sơ ứng viên. Tuy nhiên có một số trường hợp chỉ gán một từ khóa cho đối tượng thì vẫn chưa đủ, ví dụ giá bán của sản phẩm nếu chỉ gán từ khóa “Price” thì không có ý nghĩa, vì vậy thuộc tính “TagValue” sẽ định nghĩa giá trị cho “TagName”, trong trường hợp này sẽ là một con số cụ thể cho giá sản phầm.

Với kiến trúc tổ chức thông tin như trên, đảm bảo tính mở rộng của kiến trúc, khả năng tích hợp vào một hệ thống khác dễ dàng và việc so khớp thông tin giữa các đối tượng khác nhau được thực thi hiệu quả hơn so với mô hình kiến trúc nghiệp vụ thông thường.

### Áp dụng độ tương quan giữa các tag vào việc so khớp

Trong quá trình thực hiện đề tài, khi nghiên cứu phương pháp so khớp thông tin dựa trên các tập thuộc tính Tag của đối tượng, chúng tôi nhận thấy vấn đề độ tương quan ngữ nghĩa giữa các từ khóa, như đã được đề cập ở phần trên, nhưng do phạm vi đề tài và thời gian hạn chế nên chúng tôi quyết định không nghiên cứu phương pháp tính độ tương quan ngữ nghĩa của hai từ khóa. Thay vào đó chúng tôi tích hợp kết quả tính toán độ tương quan vào kiến trúc, sau đó xây dựng phương pháp so khớp thông tin có ngữ nghĩa. Nhờ vậy kiến trúc đảm bảo khả năng mở rộng và phát triển Semantic web nhằm xây dựng hệ thống so khớp hoàn thiện hơn.



Hình 23. Cấu trúc bảng mức độ tương quan SimilarTerm

Về phương pháp so khớp, nhóm thực hiện bằng cách duyệt và so sánh từng thuộc tính trong hai tập thuộc tính của hai đối tượng khác nhau. Nếu hai từ khóa so sánh không khớp hoàn toàn với nhau, sẽ tìm từ khóa tương quan trong bảng SimilarTerm, sau đó sử dụng từ khóa thực hiện so sánh lần nữa. Kết quả so sánh của hai đối tượng là số phần trăm khớp với nhau.

Công thức tính mức độ so khớp

Tính dễ mở rộng

### Áp dụng cây quyết định

#### Mircosoft Analysis Services

Microsoft SQL Server Denali cung cấp một môi trường làm việc với các mô hình khai thác dữ liệu được gọi là “Business Intelligence Development Studio”. Môi trường này bao gồm các thuận toán khai thác dữ liệu và các công cụ xây dựng giải pháp toàn diện cho các dự án khác nhau dễ hơn. Microsoft SQL Server Analysis Services là một thành phần của Microsoft SQL Server Denali chạy trên nền tảng của Microsoft Visual Studio cung cấp nhiều giải pháp, ứng với từng giải pháp là Microsoft tập hợp nhiều thuật toán có thể sử dụng trong khai thác dữ liệu. Sau khi tiến hành sử dụng và đánh giá một số công cụ hỗ trợ tạo cây quyết định hiện có như Weka, Mircosoft Analysis Services API,… chúng tôi quyết định lựa chọn Mircosoft Analysis Services API để ứng dụng giải quyết bài toán của nhóm. Việc sử dụng Mircosoft Analysis Services API giúp chúng tôi có thể dễ dàng xây dựng cây quyết định một cách độc lập, không phụ thuộc vào “Business Intelligence Development Studio”.



Hình 23. Cấu trúc cây quyết định khi sử dụng MS Analysis Services API[[7]](#footnote-7)



Hình 24. Dữ liệu nhập xuất khi sử dụng MS Analysis Services API

|  |  |
| --- | --- |
| Dữ liệu đầu vào | Mô tả |
| Data source | Khai báo database chứa các bảng dữ liệu |
| Data source view | Ứng với Data source, lựa chọn các table hay view dùng để khai thác với các phương pháp khai thác dữ liệu được cung cấp sẵn trên MS Analysis Services |
| Mining structure | Cấu trúc của dữ liệu khai thác (liên kết với Data Source View) |
| Mining model | Khai báo một model từ Mining Structure để tiến hành khai thác dữ liệu |
| Algorithm name and Parametter | Tên thuật toán và các tham số cần cung cấp cho thuật toán. Khai báo kèm theo Mining model |

Bảng 7. Bảng mô tả dữ liệu đầu vào khi sử dụng MS Analysis Services

|  |  |
| --- | --- |
| Model content | Mô tả |
| MODEL\_CATALOG | Tên database chứa model |
| MODEL\_NAME | Tên model |
| ATTRIBUTE\_NAME | Tên thuộc tính dự đoán (phân lớp) |
| NODE\_NAME  (NODE\_UNIQUE\_NAME) | Mã node |
| NODE\_TYPE | Kiểu node.  *Kiểu 1 - Model*: node gốc của model  *Kiểu 2 - Tree*: node gốc của cây, có NODE\_CAPTION là “All”  *Kiểu 3 - Interior*: node có node con  *Kiểu 4 - Distribution*: node lá  *Kiểu 5 - Regression tree*: node gốc của cây đệ quy, có NODE\_CAPTION là “All” |
| NODE\_CAPTION | Tên đầu đề của node  Ví dụ: “.NET = True”: các node con của node này bao gồm chính nó đều có các dòng dữ liệu chứa thuộc tính .NET |
| CHILDREN\_CARDINALITY | Số lượng node con cấp 1  Node lá luôn có giá trị bằng 0 (KHÔNG) |
| PARENT\_UNIQUE\_NAME | Node cha |
| NODE\_DESCRIPTION | Tập hợp các NODE\_CAPTION để dẫn đến node hiện tại, phụ thuộc vào độ sâu của cây.  Ví dụ: Java = False and C# = True, để đến node này, các dòng record phải không chứa Java và có chứa C#, node có đầu đề là “Java = False” sẽ là node cha của node “C# = True” |
| NODE\_RULE | Mô tả NODE\_DESCRIPTION bằng XML |
| MARGINAL\_RULE | Mô tả NODE\_CAPTION bằng XML |
| NODE\_PROBABILITY | Xác suất xảy ra node |
| MARGINAL\_PROBABILITY | Xác suất xảy ra node khi xảy ra node cha |
| NODE\_DISTRIBUTION | Sự phân bố giá trị phân lớp của node  NODE\_DISTRIBUTION là bảng lồng, được lưu trữ trong Mining model. Bảng NODE\_DISTRIBUTION bao gồm các thuộc tính sau:  ATTRIBUTE\_NAME: tên thuộc tính phân lớp  ATTRIBUTE\_VALUE: các giá trị có thể xảy ra của thuộc tính phân lớp  SUPPORT: số lượng trường hợp có thuộc tính phân lớp mang giá trị bằng ATTRIBUTE\_VALUE ứng với NODE\_SUPPORT.  PROBABILITY: bằng tỷ số giữa SUPPORT và NODE\_SUPPORT  VARIANCE: Phương sai. Đối với các giá trị rời rạc, phương sai sẽ bằng 0 (KHÔNG)  VALUETYPE: kiểu giá trị (Missing hoặc Discrete) |
| NODE\_SUPPORT | Số lượng dòng dữ liệu tương ứng với node (số trường hợp xảy ra) |
| MSOLAP\_MODEL\_COLUMN | Chỉ ra các cột có thể dùng để dự đoán |
| MSOLAP\_NODE\_SCORE | Hiển thị đến số điểm tương ứng với node |
| MSOLAP\_NODE\_SHORT\_CAPTION | Phục vụ cho mục đích hiển thị |

Bảng 8. Bảng mô tả dữ liệu đầu ra khi sử dụng MS Analysis Services

#### Các bước triển khai xây dựng cây quyết định



Hình 25. Dữ liệu đầu vào và đầu ra của Decision Tree Engine



Hình 26. Tiến trình xây dựng cây quyết định của kiến trúc



Hình 27. Cấu trúc bảng DecisionTreeNode và DecisionTreeNodeDistribution

##### Tiền xử lý dữ liệu

Dữ liệu đầu vào, cụ thể là taxonomy sẽ được Decision Tree Engine tiền xử lý để cho kết quả tối ưu nhất.

* *Hệ thống sẽ tạo ra các view tương ứng đối với từng loại đối tượng trong taxonomy. L*ọc các dòng dữ liệu lưu trữ “nhu cầu” hay “sản phẩm đáp ứng nhu cầu”, các “nhu cầu” hay “sản phẩm đáp ứng nhu cầu” này được phân nhóm theo tên và có một tiền tố (prefix) tương ứng (ví dụ “nhu cầu” có tiền tố là “NC” trước mỗi tên view).
* *Dữ liệu lưu trữ dưới dạng taxonomy phải thông qua kỹ thuật Pivot Transformation của SQL Server* nhằm giúp cho việc khai thác dữ liệu bằng cây quyết định thực hiện dễ dàng hơn. V*í dụ, chúng ta có dữ liệu như sau:*

|  |  |  |
| --- | --- | --- |
| Actor | Tên sản phẩm đáp ứng nhu cầu | Tên tag |
| A | Áo thun nam | Tay dài |
| A | Áo thun nam | Sọc caro |
| B | Áo thun nam | Tay dài |
| C | Áo thun nam | Cổ cao |
| C | Áo thun nữ | Tay dài |
| D | Áo thun nữ | Có túi |
| D | Áo thun nữ | Không cổ |
| … | … | … |

Bảng 9. Dữ liệu ví dụ trước khi xử lý bằng kỹ thuật Pivot Transformation

Sau khi xử lý bằng kỹ thuật *Pivot Transformation*, dòng sẽ được hoán đổi thành cột, giúp xây dựng cây quyết định dễ dàng hơn.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Actor | SPDUNC | Tay dài | Sọc caro | Cổ cao | Có túi | Không cổ | … |
| A | Áo thun nam | **True** | **True** | False | False | False | … |
| B | Áo thun nam | **True** | False | False | False | False | … |
| C | Áo thun nam | False | False | **True** | False | False | … |
| C | Áo thun nữ | **True** | False | False | False | False | … |
| D | Áo thun nữ | False | False | False | **True** | **True** | … |

Bảng 10. Dữ liệu bảng 9 sau khi xử lý bằng kỹ thuật Pivot Transformation

* *Lọc các ký tự đặc biệt mà MS Analysis Services không thể xử lý bao gồm “.,;'`:/\\*|?"&%$!+=()[]{}<>” để đảm bảo giữ nguyên tên thuộc tính cho “nhu cầu” và “sản phẩm đáp ứng nhu cầu”.* Việc mã hoá các ký tự đặc biệt khắc phục được khuyến điểm của Microsoft Analysis Services cũng như của “Business Intelligence Development Studio (BIDS)” (trong BIDS các ký tự đặc biệt không được giữ lại, tên các thuộc tính có ký tự đặc biệt sẽ bị BIDS lược bỏ)

**Đầu vào**: taxonomy

**Đầu ra**: các view lưu trữ cho từng “nhu cầu” và từng “sản phẩm đáp ứng nhu cầu”, có tiền tố để phân biệt giữa view lưu trữ “nhu cầu” và view lưu trữ “sản phẩm đáp ứng nhu cầu”. Các view được xử lý bằng kỹ thuật Pivot Transformation và được mã hoá các ký tự đặc biệt

##### Tạo Data source

Tạo data source dựa vào thông số kết nối đến Database Engine lưu trữ taxonomy.

**Đầu vào**: Thông số kết nối đến Database Engine lưu trữ taxonomy

**Đầu ra**: Data source object

##### Tạo Data source view

Dựa vào Data Source, tự động dựa vào các tiền tố được tạo ra ở bước tiền xử lý dữ liệu, từ đó tạo ra các Data source view. Các view có tiền tố phù hợp sẽ được tạo view hỗ trợ tạo Mining Structure.

**Đầu vào**: Data source object, prefix được định nghĩa ở bước tiền xử lý dữ liệu

**Đầu ra**: Data source view object

##### Định nghĩa các tham số thuật toán

Lựa chọn thuật toán xây dựng cây quyết định và các tham số tương ứng.

*Các tham số thuật toán bao gồm:*

* HoldoutMaxPercent: số lượng dữ liệu dùng để kiểm tra (đơn vị tính: %)
* SCORE\_METHOD: Tên thuật toán dùng để xây dựng cây quyết định (Entropy, Bayesian with K2 Prior, hay Bayesian Dirichlet Equivalent (BDE) Prior)
* COMPLEXITY\_PENALTY:
* SPLIT\_METHOD:
* MAXIMUM\_INPUT\_ATTRIBUTES:
* MAXIMUM\_OUTPUT\_ATTRIBUTES:
* MINIMUM\_SUPPORT:

**Đầu vào:** KHÔNG

**Đầu ra:** các tham số tham gia vào quá trình xây dựng cây quyết định

##### Tạo Mining Structures và Mining Models

Các tham số thuật toán được tạo ở giai đoạn trên sẽ được sử dụng để tạo ra Mining Structure và Mining Model. Data source và Data source view cùng với prefix sẽ xác định các model cần tiến hành xây dựng cây quyết định, các view có trong Data source view có prefix tương ứng sẽ được tiến hành xây dựng cây ở bước tiếp theo nhằm tránh sự lãng phí tài nguyên hệ thống.

**Đầu vào**: Data source, Data source view, tham số thuật toán, prefix được định nghĩa bước tiền xử lý dữ liệu

**Đầu ra**: Mining Structures và Mining Models được tạo

##### Tiến hành khai thác dữ liệu

Sau khi các Mining Structures và Mining Models được tạo ra, dữ liệu sẽ được nạp và xử lý, quá trình xử lý có thể kéo dài hàng giờ, hàng tuần hay hàng tháng phụ thuộc vào số lượng dữ liệu. Việc tiến hành khai thác dữ liệu trên một server độc lập với server chính sẽ góp phần đảm bảo hiệu năng của hệ thống khi xây dựng cây quyết định.

**Đầu vào**: Mining Structures và Mining Models được tạo

**Đầu ra**: Mining Models đã được xử lý

##### Xuất dữ liệu đã khai thác vào Database Engine

Sau khi dữ liệu đã được phân tích xong sẽ được lưu trữ về Server chính để dễ dàng truy xuất. Các ký tự đặc biệt được mã hoá ở giai đoạn tiền xử lý dữ liệu sẽ được giải mã ở giai đoạn này.

Một “Linked Server” được tạo ra trên Database Engine nhằm kết nối, lấy dữ liệu từ Analysis Services thông qua ngôn ngữ truy vấn MDX và ngôn ngữ truy vấn SQL. “Linked Server” là kỹ thuật liên kết các Server lại với nhau, giúp truy vấn dữ liệu từ một database đặt tại server khác (hoặc một instance khác của SQL Server).

Trong quá trình xuất dữ liệu, ***các Mining Model sẽ được liên kết với nhau thông qua node gốc của model*** (NODE\_NAME = “0”, NODE\_TYPE = 1), các node gốc của cây (NODE\_CAPTION = “All”, NODE\_TYPE = 2) sẽ được chuyển thành tên của “nhu cầu” hay “sản phẩm đáp ứng nhu cầu” để dễ dàng truy vấn.

**Đầu vào**: Mining Models đã được xử lý

**Đầu ra**: DecisionTreeNode và DecisionTreeNodeDistribution chứa kết quả khai thác (các cây quyết định đã được liên kết với nhau)

##### Dọn dẹp và đóng các kết nối

Sau khi cây quyết định được hoàn thành và lưu trữ trở về database chính, cần dọn dẹp các bảng dữ liệu tạm và đóng các kết nối sử dụng trong quá trình tạo cây.

**Đầu vào**: Kết quả của quá trình xây dựng cây quyết định.

**Đầu ra**: Dọn dẹp dữ liệu và đóng các kết nối.

#### Cấu trúc dữ liệu cho người dùng

## Áp dụng kiến trúc phần mềm vào bài toán

## Đánh giá kiến trúc

# Áp dụng giải pháp vào bài toán Job Zoom

## Thực trạng các website tuyển dụng hiện nay

***Về phía người tìm việc:***

* Đăng tải thông tin CV theo khuôn mẫu nhất định
* Hạn chế trong việc hỗ trợ người tìm việc mô tả, minh chứng cho những kinh nghiệm hay kỹ năng của họ
* Không hỗ trợ ứng viên viết CV, không thể hiện kiến thức, kỹ năng nào có mức độ quan trọng đối với công việc hiện tại mà họ muốn apply, không thể hiện cho ứng viên biết kỹ năng nào ứng viên còn thiếu sót cần học tập, trau dồi thêm. Các website hiện tại, chức năng gợi ý cho ứng viên bổ sung thông tin hồ sơ còn đơn giản (chỉ gợi ý ứng viên nên cập nhật các thông tin quan trọng như kinh nghiệm làm việc, học vấn), không có hướng đến một vị trí làm việc hay ngành nghề mà ứng viên muốn làm việc, không gợi ý những thông tin có thể quyết định cơ hội việc làm cho ứng viên.

***Về phía nhà tuyển dụng***

* Đăng tải thông tin việc làm theo dạng văn bản theo khuôn mẫu nhất định.
* Chưa hỗ trợ nhà tuyển dụng đăng tải thông tin về công việc
* Hỗ trợ tốt việc tìm kiếm công việc, hồ sơ ứng viên nhưng chưa thấy được sự so khớp giữa yêu cầu tuyển dụng và hồ sơ của ứng viên.

## Những vấn đề Job Zoom cần giải quyết

### Vấn đề 1: Hỗ trợ người dùng viết CV theo ngành nghề.

**Bạn là người tìm việc?** Có thể bạn đã từng gặp khó khăn trong việc viết sơ yếu lý lịch (CV hay còn gọi là resume) để tạo ấn tượng cho nhà tuyển dụng, tiếp thị cho bản thân mình. Bên cạnh đó, mẫu CV sẽ khác nhau đối với từng ngành nghề khác nhau. Để lọt vào danh sách ứng viên hàng đầu của nhà tuyển dụng, bạn cần phải có một sơ yếu lý lịch ngắn gọn, súc tích nhưng cần đầy đủ và rõ ràng về quá trình làm việc, trình độ chuyên môn, học vấn và các kỹ năng mềm khác, việc viết CV này thật sự trở nên khó khăn khi bạn vẫn loay hoay chưa biết phải trình bày hay liệt kê những thông tin như thế nào. Những ứng dụng tìm việc hiện tại không thể gợi ý cho bạn tạo một CV chuẩn cho một ngành nghề cụ thể hay nói khác đi là không hỗ trợ người dùng tạo CV tìm việc đầy đủ và rõ ràng đối với ngành nghề mà người tìm việc muốn ứng tuyển.

Khi người tìm việc muốn tìm một công việc nào đó, bên cạnh những thông tin cá nhân, ứng dụng sẽ gợi ý một CV chuẩn về ngành nghề, người dùng sẽ tạo ra sơ yếu lý lịch cho mình dựa vào các thuộc tính chương trình gợi ý. Ngoài ra, khi cần ứng tuyển vào một ví trí của một nhà tuyển dụng nào, ứng dụng sẽ gợi ý người dùng những thông tin cần bổ sung cho vị trí tìm việc.

### Vấn đề 2: Hỗ trợ nhà tuyển dụng trong việc đăng tải yêu cầu công việc một cách chi tiết và có trọng số theo ngành nghề.

**Bạn là nhà tuyển dụng?** Các website tìm việc hiện nay, đều cho nhà tuyển dụng đăng thông tin dưới dạng văn bản thô, không đưa ra gợi ý cho bạn về các ứng viên có khả năng đáp ứng yêu cầu công việc dựa vào các trọng số các yêu cầu của bạn đưa ra. Khi nhà tuyển dụng muốn đăng thông tin tuyển dụng, các website tìm việc hiện tại không thể đánh giá mức độ đáp ứng của những ứng viên dựa vào những thông tin nhà tuyển dụng cung cấp, không thể biết ứng viên đáp ứng được bao nhiêu phần trăm những yêu cầu này, nhà tuyển dụng phải tự đánh giá bằng việc đọc những CV của ứng viên sau đó lựa chọn, việc này gây mất nhiều thời gian và chi phí.

Với Job Zoom framework, ứng dụng hỗ trợ nhà tuyển dụng đăng tải thông tin tuyển dụng linh động hơn và có thể đặt trọng số cho từng yêu cầu của công việc. Ứng dụng đánh giá mức độ phù hợp của ứng viên và liệt kê những ứng viên tiềm năng dựa vào thông tin của nhà tuyển dụng cung cấp.

### Vấn đề 3: So khớp hồ sơ ứng viên với yêu cầu tuyển dụng

Đối với người tìm việc: Hệ thồng so khớp gợi ý người dùng hoàn thiện, bổ sung những thiếu sót của CV, tăng cơ hội ứng tuyển vào một vị trí tại một công ty cụ thể.

Đối với nhà tuyển dụng: Hệ thồng so khớp đánh giá ứng viên xin ứng tuyển vào công việc, hỗ trợ cho nhà tuyển dụng trong việc đánh giá ứng viên

## Kết quả mong muốn

Áp dụng “***Giải quyết bài toán kiến trúc***” vào giải quyết các vấn đề trên.

* Xây dựng JobZoom framework đảm bảo thông tin về CV và yêu cầu công việc được lưu trữ linh hoạt, dễ dàng so khớp.
* Áp dụng so khớp kết hợp độ tương quan giữa các tag, hỗ trợ nhà tuyển dụng đánh giá ứng viên và giúp ứng viên nhận ra khuyết điểm của mình, hoàn thiện CV.
* Kết hợp phương pháp khai thác dữ liệu bằng cây quyết định tạo nền tảng cải thiện khả năng thích ứng của hệ thống với từng loại ngành nghề, gợi ý ứng viên viết CV và nhà tuyển dụng đăng yêu cầu công việc một cách nhanh chóng và hiệu quả
* Xây dựng được kiến trúc phần mềm đảm bảo hiệu năng của hệ thống với số lượng người sử dụng lớn khi triển khai cây quyết định và hệ thống so khớp.
* JobZoom framework dễ dàng mở rộng, triển khai.

# Giải quyết những vấn đề trong Job Zoom

## Khái quát phương pháp giải quyết vấn đề

Những vấn đề của JobZoom framework xoay quanh bài toán kiến trúc của chúng tôi. Để dễ dàng so khớp giữa các hồ sơ của ứng viên và yêu cầu tuyển dụng, đòi hỏi thông tin về hồ sơ ứng viên cũng như yêu cầu tuyển dụng cần phải đạt mức độ linh hoạt, có mối quan hệ tương đồng trong quá trình phân loại, lưu trữ để có thể dễ dàng so khớp. Đồng thời, việc ứng dụng cây quyết định và hệ thống so khớp sẽ giúp nhà tuyển dụng dễ dàng hơn trong việc đăng tải yêu cầu công việc và giúp cho người tìm việc dễ dàng viết một hồ sơ xin việc một cách chi tiết, rõ ràng hơn.

## Ứng dụng lưu trữ dưới dạng tag kết hợp taxonomy

Khi ứng dụng lưu trữ dưới dạng tag kết hợp taxonomy, chúng tôi nhận thấy CV của người tìm việc và yêu cầu tuyển dụng của doanh nghiệp được mô tả rất linh hoạt. Thông tin được lưu trữ dưới dạng cấu trúc cây phân cấp, có mức độ tương đồng giữa cây thông tin về CV và yêu cầu tuyển dụng nên việc so khớp có thể được thực hiện và đảm bảo hiệu năng.

## Ứng dụng hệ thống so khớp kết hợp mức độ tương quan

Áp dụng vào đâu, áp dụng dc ko, có hiệu quả ko

## Ứng dụng cây quyết định

Giải pháp kiến trúc

Thuộc tính phân lớp “Is Approve” (thuộc tính xác định ứng viên có được nhà tuyển dụng nhận hay không sau khi apply) đóng vai trò là thuộc tính phân lớp.

Như vậy, khi khai thác dữ liệu, hệ thống sẽ sinh ra rất nhiều cây quyết định theo từng vị trí làm việc, những cây này sẽ được gom lại giúp việc truy vấn kết quả trở nên dễ dàng hơn

Kết quả khi tạo cây quyết định thể hiện chúng ta thấy mức độ phụ thuộc, tầm quan trọng của các thuộc tính trong việc lựa chọn ứng viên vào một ví trí cụ thể. Những thuộc tính này gợi ý cho người dùng những kỹ năng, kiến thức người tìm việc nên có đối với vị trí làm việc mà họ đang tìm. Như vậy, khi người dùng sử dụng chương trình, chương trình sẽ đặt câu hỏi và liệt kê các thuộc tính (kỹ năng, kinh nghiệm) có mức độ phụ thuộc cao vào xác suất ứng viên được nhận, ứng viên có thể thay đổi CV cho phù hợp với vị trí mình đang cần ứng tuyển. Nếu một trong những thuộc tính mà Job Zoom gợi ý người tìm việc không thể đáp ứng được, thì Job Zoom có thể gợi ý cho ứng viên những thuộc tính khác hoặc ứng viên có thể học tập, rèn luyện để bổ sung kịp thời những thuộc tính đó nhằm tăng xác suất và tính cạnh tranh của họ khi apply vào công việc.

Thông qua cây quyết định và việc gom nhóm cây, vấn đề 1 của JobZoom framework đã được giải quyết. Tuy nhiên, mức độ giải quyết này đáp ứng cho những ứng viên muốn ứng tuyển vào một vị trí công việc bất kỳ, xác suất của ứng viên được nhận cao khi ứng viên apply vào nhiều công ty tuyển dụng tương ứng với vị trí đó. Nhưng chính việc ứng dụng taxonomy được đề cập ở trên đã giải quyết phần còn lại của bài toán 1, khi ứng viên apply vào vị trí công việc tại một công ty cụ thể, thì những thuộc tính nào giúp ứng viên apply vào vị trí đó có xác suất được nhận cao nhất.

* **Pivot transformation:**

Trong quá trình xây dựng cây quyết định thông qua giải pháp chúng tôi đề ra, chúng tôi tiến hành tiền xử lý dữ liệu cho JobZoom framework như sau:

* Thông tin cho từng vị trí làm việc được lưu trữ theo dạng Tagging (Taxonomy) bằng kỹ thuật Pivot Transformation trong SQL Server.

Ví dụ dữ liệu của thông tin tuyển dụng gồm vị trí đăng tuyển (Job Title), tên công ty (Company), và một tập hợp các yêu cầu/ tiêu chí, với mỗi hàng là một tiêu chí cụ thể.

|  |  |  |
| --- | --- | --- |
| **Usage** | **SetKey** | **PivotKey** |
| **Column Name** | **Job Title** | **Tag** |
| **Data**  **Records** | Developer | OOP |
| Developer | Software design |
| Developer | DBMS |
| Developer | .Net Framework |
| Tester | Testing Technique |
| Tester | .Net Framework |
| Tester | Automation Testing |
| Tester | DBMS |

Bảng 11. Bảng dữ liệu các thông tin tuyển dụng trước khi Pivot trên cột Tag

Bảng sau thể hiện kết quả sau khi Pivot Transformation

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Job Title** | **OOP** | **Software design** | **DBMS** | **.Net Framework** | **Testing Technique** | **Automation Testing** |
| Data  Records | Developer | True | True | True | True | False | False |
| Tester | False | False | True | True | True | True |

Bảng 12. Bảng kết quả sau khi Pivot Transformation

# Kiến trúc cổng thông tin tìm việc JobZoom

## Điều kiện ra đời

Ngày nay, internet đã dần chiếm hữu cuộc sống hiện tại, hết thảy mọi việc đều được “số hoá” cũng là lúc tuyển dụng và người tìm việc đều chọn các website tìm việc làm cầu nối cho mình.

Theo số liệu thống kê của tổ chức Internet Usage World Stats[[8]](#footnote-8), tính đến nay có hơn 2,1 tỉ người trên thế giới sử dụng Internet. Bên cạnh đó, nghiên cứu về người tiêu dùng do Mintel International Group Ltd. (tổ chức quốc tế chuyên nghiên cứu về thị trường và người tiêu dùng) tiến hành cho thấy có đến 30% số người dùng Internet đã tham khảo các website tuyển dụng trực tuyến để tìm kiếm thông tin về việc làm. Chúng ta có thể thấy xu hướng tìm việc trực tuyến đang ngày càng phát triển với tốc độc rất nhanh.

Việt Nam là quốc gia đang phát triển với dân số trên 90 triệu người với trên 28,6 triệu người sử dụng internet chiếm khoảng 31,6% dân số (2011)[[9]](#footnote-9). Các trang web về lao động việc làm ra đời đã ngày càng đem lại nhiều thông tin tổng hợp, đa dạng về công việc, mở ra những cơ hội lớn hơn cho người tìm việc, đồng thời giúp doanh nghiệp giải quyết nhu cầu nhân sự nhanh chóng và hiệu quả hơn. Tuy nhiên, thực trạng chung của các website tìm việc hiện nay chưa ứng dụng triệt để khai thác dữ liệu trong việc việc gợi ý người dùng viết CV hay hỗ trợ doanh nghiệp trong việc đánh giá ứng viên thông qua thông tin ứng viên cung cấp. Để góp phần đáp ứng những yêu cầu thực tiễn này, đề tài “**xây dựng kiến trúc hệ thống cổng thông tin tìm việc**” được ra đời.

JobZoom framework được phát triển dựa theo phần 4 - Giải quyết bài toán kiến trúc.

## Kiến trúc JobZoom framework

### Kiến trúc tổng quan framework

Secenario



|  |  |
| --- | --- |
| Actor | Mô tả |
| <Actor> | Actor tổng quát của Framework, mô tả người sử dụng Framework này vào nhiều mục đích khác nhau. Ở mỗi mục đích, Actor đó có một thể hiện riêng cụ thể: Người tìm kiếm việc (JobSeeker), Doanh nghiệp tuyển việc (Employer), Lập trình viên phát triển ứng dụng dựa trên Framework này (Developer) |
| JobSeeker | Ứng viên tìm việc sử dụng hệ thống (ứng dụng JobZoom Framework) |
| Employer | Nhà tuyển dụng sử dụng hệ thống (ứng dụng JobZoom Framework) |
| Developer | Lập trình viên phát triển hệ thống tìm việc (ứng dụng JobZoom Framework) |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Tên Use Case | Actor | Description |
| UC01 | Manage Flexible Attribute | Actor |  |
| UC02 | Manage Taxonomy Engine | Actor |  |
| UC02 | Manage Data Mining Engine | Actor |  |
| UC03 | Manage Framework Engine | Actor |  |
| UC04 | Support Search Engine | Emloyer, JobSeeker |  |
| UC05 | Support Resume Zoomming | JobSeeker |  |
| UC06 | Support Employer’s Decision | Employer |  |
| UC07 | Support JobSeeker Resume Finalization | JobSeeker |  |

Logical View



Process View



Physical View

Physical View

### Kiến trúc thông tin linh hoạt

### Taxonomy



### Hệ thống so khớp (Matching tool) kết hợp độ tương quan giữa các tag

### Data mining

+ Decision tree

+ Đảm bảo hiệu năng hệ thống.

Hệ thống tự động tạo view, khai thác dữ liệu và lưu dữ liệu vào kho để truy xuất nhanh chóng và dễ dàng. Quá trình này được thực hiện độc lập trên một máy chủ khác …

## Mô hình kiến trúc Job Zoom có nhiều ưu điểm vượt trội

### Kiến trúc linh hoạt và khả năng mở rộng của framework

#### Interface minh chứng CV

#### Interface minh chứng yêu cầu công việc

#### Interface semantic web

#### Interface data mining

Tuỳ thuộc vào điều kiện sẽ có cách mining khác nhau

### Khả năng ứng dụng vào những ngành nghề khác nhau

### Khả năng ứng dụng vào những lĩnh vực khác

### Triển khai nhanh chóng

Người dùng không cần thay đổi cấu trúc bảng trong database.

Cách triển khai: …

### Đảm bảo hiệu năng hệ thống

ADO.NET Entity Framework tốc độ so với Linq

## So sánh Job Zoom với các website tìm việc hiện tại

Mô hình kiến trúc khắc phục những điểm yếu của các website tìm việc hiện có:

|  |  |  |
| --- | --- | --- |
| STT | Các website tìm việc hiện tại | JobZoom framework |
| 1 | Đăng thông tin ứng viên và thông tin về việc làm dưới dạng văn bản | Đăng thông tin ứng viên và yêu cầu tuyển dụng dưới dạng tag, đạt được sự linh hoạt và dễ dàng so khớp hơn. |
| 2 | CV theo khuôn mẫu, hạn chế trong việc hỗ trợ người tìm việc mô tả minh chứng cho những kinh nghiệm hay kỹ năng của họ |
| 3 | Chức năng gợi ý ứng viên bổ sung thông tin hồ sơ còn đơn giản (chỉ gợi ý ứng viên nên cập nhật các thông tin quan trọng như kinh nghiệm làm việc, học vấn), không có hướng đến một vị trí làm việc hay ngành nghề mà ứng viên muốn làm việc, không gợi ý những thông tin có thể quyết định cơ hội việc làm cho ứng viên. | Hỗ trợ ứng viên viết CV: JobZoom gợi ý những thuộc tính quan trọng giúp ứng viên tạo ấn tượng với CV của mình, đồng thời, những kỹ năng mà JobZoom gợi ý ứng viên chưa có để ứng viên kịp thời bổ sung kiến thức hay kinh nghiệm cho bản thân. |
| 4 | Hỗ trợ tốt việc tìm kiếm công việc, hồ sơ ứng viên nhưng chưa thấy được sự so khớp giữa yêu cầu tuyển dụng và hồ sơ của ứng viên. | Có thể so khớp giữa yêu cầu tuyển dụng và hồ sơ của ứng viên |
| 5 | Không hỗ trợ nhà tuyển dụng đăng thông tin tuyển dụng | JobZoom hỗ trợ nhà tuyển dụng đăng thông tin tuyển dụng bằng phương pháp khai thác dữ liệu |

Bảng 13. So sánh JobZoom framework với các website hiện tại

# Đánh giá và hướng phát triển

## Những điểm làm được

* Kiến trúc giải quyết được nhu cầu lưu trữ thông tin nhiều, đa dạng, dễ dàng trong việc tìm kiếm và so khớp
* Tổ chức thông tin linh động dưới dạng tag kết hợp taxonomy. Thông tin được lưu trữ dưới dạng cây đa cấp và có mức độ tương quan giữa các tag.
* Hệ thống so khớp linh hoạt, kết hợp mức độ tương quan giữa các tag giải quyết được khuyết điểm của việc so sánh object (A phải khớp hoàn toàn với B).
* Kiến trúc dễ dàng triển khai, mở rộng, đồng thời hiệu năng của hệ thống vẫn được đảm bảo khi áp dụng cây quyết định và hệ thống so khớp để gợi ý cho actor xây dựng cây thông tin nhu cầu và “sản phẩm đáp ứng nhu cầu” một cách nhanh chóng và hiệu quả.
* Khắc phục hạn chế của “Microsoft Analysis Services[[10]](#footnote-10)”: không hỗ trợ xử lý ký tự đặc biệt. Hoàn toàn không phụ thuộc vào công cụ “SQL Server Business Intelligence Development Studio[[11]](#footnote-11)”
* Ứng dụng kiến trúc vào bài toán JobZoom nhằm đánh giá khả năng tổ chức thông tin linh hoạt, hiệu quả của hệ thống so khớp, đồng thời đánh giá tính mở rộng, dễ dàng triển khai và tái sử dụng của kiến trúc khi áp dụng trong các lĩnh vực cần lưu trữ và so khớp thông tin.
* Nhóm đã xây dựng và phát triển JobZoom framework ứng dụng những điểm mạnh của bài toán kiến trúc:
* Người tìm việc tìm kiếm công việc phù hợp nhanh chóng, viết CV dễ dàng hơn dựa vào gợi ý từ kết quả khai thác dữ liệu và hệ thống so khớp
* Nhà tuyển dụng dễ dàng đăng tuyển yêu cầu công việc một cách chi tiết và lựa chọn ứng viên phù hợp với vị trí làm việc thông qua hệ thống so khớp và cây quyết định. Hệ thống so khớp giúp đánh giá ứng viên cho kết quả cụ thể (yêu cầu nào đạt hay không đạt và số điểm ứng viên đạt được)
* JobZoom Framework dễ dàng triển khai, mở rộng, kết hợp được với phương pháp khai thác dữ liệu “cây quyết định” tạo nền tảng cải thiện khả năng thích ứng của hệ thống với từng loại ngành nghề.

## Những điểm hạn chế

Tuy có những ưu điểm như trên nhưng do kinh nghiệm, thời gian hạn chế nên kiến trúc hệ thống của nhóm chúng tôi cũng còn một số hạn chế cần khắc phục:

* Cây đa cấp còn ở mức trừu tượng thông qua bảng SimilarTerm để thể hiện mức độ tương quan giữa các tag, việc xác định độ tương quan giữa các tag sẽ được triển khai dễ dàng dựa vào tính dễ dàng mở rộng của kiến trúc.
* Chưa được áp dụng vào thực tế nên bài toán kiến trúc vẫn còn ở mức trừu tượng, cần điều chỉnh để đảm bảo tính khách quan và linh hoạt của hệ thống so khớp. Dữ liệu kiểm tra vẫn chưa nhiều chưa chứng minh được hiệu quả của kiến trúc khi áp dụng vào hệ thống nhiều ngành nghề.
* Chưa triển khai được kiến trúc JobZoom framework trong thực tế nên khó kiểm tra tính khả thi và tốc độ tính toán, so khớp và xây dựng cây quyết định do nhóm phát triển khi triển khai với số lượng người dùng lớn. Tuy nhiên, hiệu năng của hệ thống được cải thiện thông qua việc xây dựng cây quyết định trên một hệ thống độc lập sau đó lưu trữ kết quả về database chính để truy vấn nhanh chóng, duyệt thông tin được lưu trữ dưới dạng cây nhanh hơn lưu trữ dưới dạng văn bản.

## Hướng phát triển

### Xác thực độ tin cậy của CV và yêu cầu tuyển dụng

Các đối tượng tham gia vào những cộng đồng trực tuyến thường là các đối tượng “ảo” và thường không được xác định tính chính xác, tính xác thực của các thông tin đăng tải. Việc xác thực độ tin cậy của CV và yêu cầu tuyển dụng sẽ giúp hệ thống vận hành tốt hơn, tránh rủi ro và tạo uy tín với người sử dụng. Với kiến trúc linh hoạt của hệ thống ta có thể dễ dàng tích hợp công cụ xác định độ tin cậy của người dùng từ đó xác thực thông tin của họ.

### Phát triển Semantic web

Với hệ thống hiện tại của chúng tôi chỉ mô hình lại kết quả phân tích về tương quan ngữ nghĩa (Semantic similarity) giữa các tag và tích hợp vào hệ thống bằng một interface cơ bản. Việc xây dựng và triển khai hệ thống Semantic web sẽ hỗ trợ người dùng tìm kiến thông tin nhanh chóng, chuẩn xác, thông minh và trả ra kết quả so khớp chính xác hơn.

### Đánh trọng số cho các thuộc tính dựa vào độ sâu của taxonomy

Phát triển cây đa cấp dựa vào semantic web, những tag có liên quan với những tag khác, khi matching sẽ được một số điểm tương ứng, số điểm này được dựa vào độ sâu của tag.

### Phân tán dữ liệu

Ứng dụng những điểm mạnh của cơ sở dữ liệu phân tán nhằm tăng hiệu năng cho hệ thống.

### Công cụ hỗ trợ lựa chọn ứng viên

Công cụ hỗ trợ lựa chọn ứng viên: phỏng vấn trực tuyến, thuyết trình qua internet… có thể được phát triển để hỗ trợ nhà tuyển dụng và ứng viên.

### Thu thập thông tin việc làm tự động

Giúp thu thập thông tin việc làm từ các cổng thông tin tìm việc hiện tại.

### Áp dụng quy trình tuyển dụng vào hệ thống

Kiến trúc đưa ra các ứng viên có số điểm cao cho nhà tuyển dụng quyết định, đây chỉ là một khâu trong quy trình tuyển dụng ứng viên. Hệ thống nên có thêm quy trình tuyển dụng, quản lý quy trình từ lúc ứng viên nộp hồ sơ đến khi ứng viên trở thành nhân viên của công ty, giúp nhà tuyển dụng dễ dàng trong việc quản lý quy trình tuyển dụng của mình.

TÀI LIỆU THAM KHẢO

[1] Lior Rokach & Oded Maimon, Data Mining with Decision Trees: Theory and Applications, 2008.

[2]Jamie MacLennan, ZhaoHui Tang & Bogdan Crivat, *Data Mining with Microsoft SQL Server 2008*, Wiley Publishing Inc., 2008

[3] Lan H. Witten, Eibe Frank and Mark A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques - Third Edition*, Elsevier, 2011

[4] Rafael Olivas, *Decision Trees - A Primer forDecision-making Professionals,* 2007

[5] Scott A. Golder & Bernardo A. Huberman, *The Structure of Collaborative Tagging Systems*

[6] Adam Freeman & Steven Sanderson, *Pro ASP.NET MVC 3 Framework – Third Edition*, Apress

[7] Tomislav Piasevoli, *MDX with Microsoft SQL Server 2008 R2 Analysis Services: Cookbook,* Packt Publishing, 2011

[8] Arttennick, *Practical MDX Queries for Microsoft SQL Server Analysis Serivces 2008*, McGraw-Hill Companies

[9] http://bis.net.vn/forums/p/378/661.aspx

[10] Nguyễn Thị Thùy Linh, *Thuật toán phân lớp cây quyết định*, Khóa luận tốt nghiệp đại học, Trường Đại học Công nghệ, 2005.

[11] K. Cwalina, B. Abrams, *Framework Design Guidelines: Conventions, Idiom, and Patterns for Reusable .NET libraries Second Edition*, Addison Wesley, October 2008

[12] Kruchten, Philippe (1995, November). *Architectural Blueprints — The “4+1” View Model of Software Architecture*

[13] H.T.Kung, C.H.Wu *Content Networks: Taxonomy and New Approaches The Internet as a Large-Scale Complex System*, Kihong Park and Walter Willinger (Editors), published by Oxford University Press as part of Sante Fe Institute series, 2002.

[14] D.Wollersheim, W.J.Rahayu *Using Medical Test Collection Relevance Judgement to Identify Ontological Relationships Useful for Query Expansion* 21st International Conference on Data Engineering 2005.

[15] OU Shi-yan, KHOO Christopher S.G, GOH Dion H. Division of Information Studies, *Constructing a taxonomy to support multi-document summarization of dissertation abstracts*. Proceedings Issue of the 1st International Conference on Universal Digital Library (ICUDL 2005).

[16]

[17]

[18]

[19]

[20]

[21]

[22]

[23]

[24]

1. Nguồn: http://www.pcworld.com.vn/articles/tieu-dung/song-cong-nghe/2011/08/1227918/email-hanh-trinh-29-nam-lich-su/ [↑](#footnote-ref-1)
2. Nguồn: http://vi.wikipedia.org/wiki/Tag [↑](#footnote-ref-2)
3. Tham khảo: http://www.hipertext.net/english/pag1011.htm#origenNota2 [↑](#footnote-ref-3)
4. Tài liệu tham khảo [15] [↑](#footnote-ref-4)
5. Tài liệu tham khảo [13] [↑](#footnote-ref-5)
6. Tài liệu tham khảo [14] [↑](#footnote-ref-6)
7. Nguồn: http://msdn.microsoft.com/en-us/library/cc645758.aspx [↑](#footnote-ref-7)
8. http://www.internetworldstats.com/stats.htm [↑](#footnote-ref-8)
9. http://www.internetworldstats.com/asia.htm#vn [↑](#footnote-ref-9)
10. Microsoft Analysis Services: Thành phần của Microsoft SQL Server, dùng để khai thác dữ liệu [↑](#footnote-ref-10)
11. SQL Server Business Intelligence Development Studio: Công cụ hỗ trợ khai thác dữ liệu của Microsoft SQL Server [↑](#footnote-ref-11)