**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN HỆ THỐNG THÔNG TIN**

**TRẦN THỊ LY LY – 1512312**

**TRẦN HIỀN LƯƠNG – 1512308**

**XÂY DỰNG HỆ THỐNG CHATBOT HỖ TRỢ HỎI ĐÁP CHO CÁC ỨNG DỤNG TRONG LĨNH VỰC KĨ THUẬT TẬP TRUNG VÀO CÂU HỎI WHAT – HOW**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT**

**GIÁO VIÊN HƯỚNG DẪN**

**TS. PHẠM NGUYỄN CƯƠNG**

**KHÓA 2015 - 2019**

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

…………………………………………………………………………………………………………………………………………………..

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

TpHCM, ngày ……. tháng ….. năm 2019

Giáo viên hướng dẫn

[Kí tên và ghi rõ họ tên]

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

…………………………………………………………………………………………………………………………………………………..

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………

TpHCM, ngày ……. tháng ….. năm 2019

Giáo viên hướng dẫn

[Kí tên và ghi rõ họ tên]

**LỜI CẢM ƠN**

Chúng em xin gửi lời cảm ơn chân thành nhất đến thầy – T.S Phạm Nguyễn Cương vì đã cho tụi em cơ hội được thực hiện đề tài này khi mà tụi em chưa có chút kiến thức nào lúc mới bắt đầu. Cảm ơn thầy đã tận tình chỉ bảo tụi em bằng một cách nhẹ nhàng và hiệu quả nhất, không vồ vập chỉ để hoàn thành đề tài mà những gì thầy cho tụi em là cái tầm nhìn lớn của một người đã nhiều năm kinh nghiệm.

Chúng em xin chân thành cảm ơn thầy Lê Đình Thắng – nhờ thầy mà chúng em có thể thực hiện đề tài này. Cảm ơn thầy đã cung cấp cho chúng em nhiều tư liệu mà thầy cùng với nhóm của thầy đã nghiên cứu và tổng hợp lại qua nhiều năm.

Chúng em xin chân thành cảm ơn thầy Lê Nguyễn Hoài Nam đã giúp đỡ chúng em trong suốt quá trình thực hiện đề tài bằng những kiến thức vô cùng quí giá và cái nhìn khách quan nhất. Cảm ơn thầy đã có mặt trong những buổi báo cáo tiến độ của tụi em và đưa ra những góp ý giúp tụi em hoàn thiện sản phẩm tốt hơn.

Chúng em cũng xin gửi lời cảm ơn sâu sắc đến các quí thầy cô trường đại học Khoa Học Tự Nhiên, đặc biệt là khoa Công Nghệ Thông Tin – những người đã dạy tụi em trong 4 năm học ở trường. Thầy cô đã cung cấp những kiến thức cũng như kinh nghiệm của mình, giúp tụi em hoàn thiện bản thân về cả kiến thức chuyên môn lẫn xã hội.

Chúng em cũng xin cảm ơn khoa Công Nghệ Thông Tin, trường Đại học Khoa Học Tự Nhiên – ĐHQG. TPHCM đã cho chúng em một môi trường thuận lợi để học tập và thực hiện đề tài, cũng như cơ hội gặp gỡ những người thầy, người bạn giỏi nhất.

Do hạn chế về khả năng, thời gian và điều kiện nên tụi em không thể hoàn thành sản phẩm một cách tốt nhất. Kính mong thầy cô thông cảm và nhiệt tình góp ý, những nhận xét chân thành của thầy cô và các bạn chính là cơ hội để tụi em nhìn lại bản thân và cố gắng hơn sau này.

Cuối cùng, chúng em xin trân trọng cảm ơn và chúc thầy cô sức khỏe!

**Nhóm thực hiện**

Trần Thị Ly Ly và Trần Hiền Lương

**ĐỀ CƯƠNG CHI TIẾT**

|  |
| --- |
| **Tên đề tài**: Xây dựng hệ thống chatbot hỗ trợ hỏi đáp cho các phần mềm kĩ thuật tập trung vào câu hỏi What – How. |
| **Giáo viên hướng dẫn**: TS. Phạm Nguyễn Cương. |
| **Thời gian thực hiện**: 1/12/2018 – 30/6/2018. |
| **Sinh viên thực hiện**: 1512312 – Trần Thị Ly Ly, 1512308 – Trần Hiền Lương |
| **Loại đề tài:** Tìm hiểu công nghệ có ứng dụng minh họa. |
| **Nội dung đề tài:**  **Về phần nghiên cứu lí thuyết:**   * Tìm hiểu về khái niệm web ngữ nghĩa (semantic web) và cách xây dựng. * Tìm hiểu về hướng nghiên cứu hệ thống hỏi đáp (question - answering system) để ứng dụng cho chatbot. * Nghiên cứu các công cụ xây dựng chatbot. * Tìm hiểu các thuật toán xử lí ngôn ngữ tự nhiên (cụ thể là intent classification – phân loại ý định người dùng và name entity recognition – nhận diện thực thể). * Tìm hiểu về Restful API service và các framework xây dựng giao diện web cho ứng dụng.   **Về phần công nghệ:**  Để xây dựng ứng dụng minh họa, sinh viên đã sử dụng các công cụ như sau:   * RASA – Phần mềm mã nguồn mở được viết bằng python dùng để xử lí ngôn ngữ tự nhiên và phân tích ý định của người dùng. Nhóm sử dụng RASA cùng với Python làm phần backend cho chatbot. * Protégé – Công cụ để tạo ontology bằng các ngôn ngữ phổ biến hiện nay như OWL, RDF được nhóm sử dụng để xây dựng dữ liệu ngữ nghĩa cho chatbot. * VueJS – Framework javascript dùng để xây dựng giao diện người dùng cho ứng dụng trên nền web.   **Phương pháp thực hiện:**   * Về lý thuyết:   + Tìm hiểu các tài liệu khoa học liên quan đến khái niệm và cách thức hoạt động của hệ thống hỏi đáp và chatbot.  + Tìm hiểu về web ngữ nghĩa và cách xây dựng cở sở tri thức sử dụng ontology.  + Tìm hiểu một số công cụ, nền tảng dùng để xây dựng chatbot phổ biến đang được phát triển.  + Dựa vào kiến thức cá nhân để mô hình hóa hệ thống, phân tích cách thức thiết kế.   * Về kĩ thuật:   + Sử dụng nển tảng mã nguồn mở RASA để phân tích đoạn chat của người dùng và xây dựng phản hồi.  + Ngôn ngữ lập trình: python, vueJS, owl, SPARQL.  + Sử dụng Restful API để xây dựng ứng dụng web.  **Mục tiêu của đề tài:**   * Tìm hiểu và xây dựng chatbot như một nhân viên chăm sóc khách hàng, hỗ trợ giải đáp những thắc mắc của người dùng về các phần mềm trong lĩnh vực kĩ thuật tập trung vào câu hỏi what (câu hỏi về khái niệm, định nghĩa) và câu hỏi how (câu hỏi về cách thức thực hiện). * Tìm hiểu về semantic web và ứng dụng nó làm cơ sở tri thức cho đề tài. * Tìm hiểu về các công cụ xây dựng chatbot, các thuật toán cần thiết để nhận diện, phân tích câu hỏi của người dùng và xây dựng phản hồi. * Xây dựng ứng dụng minh họa trên nền tảng web sử dụng Adobe Photoshop làm dữ liệu minh họa. |

**Kế hoạch thực hiện**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Từ ngày** | **Đến ngày** | **Nội dung công việc** | |
| 1 | 1/12/2018 | 28/2/2019 | Nhận đề tài. Lên kế hoạch thực hiện công việc.  Tìm kiếm và nghiên cứu các bài báo, khái niệm về hệ thống hỏi đáp, chatbot, xử lí ngôn ngữ tự nhiên … Xác định mục tiêu, hướng đi cụ thể của đề tài và phạm vi đề tài.  Nghiên cứu các dạng câu hỏi what, how trong lĩnh vực kĩ thuật để xác định hướng đi phù hợp.  Tìm hiểu các công cụ xây dựng chatbot.  Tìm hiểu về web ngữ nghĩa.  Viết chương 1 luận văn. | |
| 2 | 1/3/2019 | 31/3/2019 | Tổng hợp kiến thức đã nghiên cứu, xây dựng mục lục và viết chương 2 luận văn.  Xác định bài toán xây dựng chatbot cho câu hỏi what how với các phần mềm lĩnh vực kĩ thuật và  Tìm hiểu RASA – là công cụ để xây dựng chatbot và lựa chọn thuật toán sử dụng. | |
| 3 | 1/4/2019 | 20/4/2019 | Xây dựng cơ sở tri thức sử dụng ontology web ngữ nghĩa.  Chuẩn bị dữ liệu cho ontology.  Lựa chọn thuật toán, xây dựng giải pháp để nhận diện ý định người dùng và tạo phản hồi.  Viết chương 3 luận văn. | |
| 4 | 21/4/2019 | 15/5/2019 | Chuẩn bị dữ liệu train và viết phần API backend để xử lí câu hỏi của người dùng và tạo phản hồi.  Tìm hiểu và xây dựng server để kết nối ontology với phần backend.  Tìm hiểu và xây dựng giao diện web cho người dùng có thể tương tác với chatbot.  Viết tiếp chương 3 luận văn và một phần của chương 4. | |
| 5 | 16/5/2019 | 1/6/2019 | Tiếp tục kiểm tra và hoàn thiện sản phẩm.  Tìm hiểu các công cụ để deploy, publish ứng dụng và sử dụng chúng để publish ứng dụng cho mọi người có thể vào sử dụng chatbot.  Hoàn thiện chương 4 luận văn. | |
| 6 | 2/6/2019 | 15/6/2019 | Tìm hiểu cách đánh giá chatbot và bắt đầu chuẩn bị dữ liệu, mẫu đánh giá để lấy ý kiến phản hồi.  Tổng kết đánh giá sản phẩm và những kiến thức thu được trong quá trình làm đề tài.  Viết chương 5 luận văn. | |
| 7 | 16/6/2019 | 19/6/2019 | Hoàn thiện báo cáo cuốn. | |
| **Xác nhận của GVHD**  **TS. Phạm Nguyễn Cương** | | | | **Ngày …… tháng …… năm ………**  **Nhóm SV thực hiện**  **Trần Thị Ly Ly Trần Hiền Lương** |

**MỤC LỤC**

[DANH MỤC BẢNG 19](#_Toc11966282)

[DANH MỤC HÌNH ẢNH 19](#_Toc11966283)

[DANH MỤC TỪ VIẾT TẮT 24](#_Toc11966284)

[CHƯƠNG 1: GIỚI THIỆU 25](#_Toc11966285)

[1.1 Giới thiệu đề tài: 25](#_Toc11966286)

[1.2 Lí do chọn đề tài 28](#_Toc11966287)

[1.3 Mục tiêu đề tài 28](#_Toc11966288)

[1.4 Phạm vi đề tài 29](#_Toc11966289)

[1.5 Bố cục luận văn 29](#_Toc11966290)

[CHƯƠNG 2: HỆ THỐNG HỎI ĐÁP (QUESTION – ANSWERING SYSTEM) VÀ CHATBOT 30](#_Toc11966291)

[2.1 Hệ thống hỏi đáp – Question answering system (QAS) 30](#_Toc11966292)

[2.1.1 Một số khái niệm cơ bản 30](#_Toc11966293)

[2.1.1.1 Truy xuất thông tin - Information Retrieve (IR) 30](#_Toc11966294)

[2.1.1.2 Xử lí ngôn ngữ tự nhiên - Natural language processing (NLP) 31](#_Toc11966295)

[2.1.1.3 Tổng quan về QAS 33](#_Toc11966296)

[2.1.2 Phân loại 35](#_Toc11966297)

[2.1.2.1 Phân loại dựa trên miền kiến thức ứng dụng (Classification based on application domain) 36](#_Toc11966298)

[2.1.2.2 Phân loại dựa trên loại câu hỏi (Classification based on types of questions) 37](#_Toc11966299)

[2.1.2.3 Phân loại dựa trên loại phương pháp phân tích được thực hiện trên câu hỏi (Classification base on type of analysis done on question) 41](#_Toc11966300)

[2.1.2.4 Phân loại dựa trên loại dữ liệu nguồn (Classification based on types of data sources) 45](#_Toc11966301)

[2.1.2.5 Phân loại dựa trên kĩ thuật được dùng trong QAS (Techniques used in QAS) 47](#_Toc11966302)

[2.1.2.6 Phân loại dựa trên những dạng câu trả lời được tạo bởi QAS (Classification based on forms of answer generated by QAS) 49](#_Toc11966303)

[2.2 Chatbot 50](#_Toc11966304)

[2.3 Kết chương 51](#_Toc11966305)

[CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ GIẢI PHÁP CHO ỨNG DỤNG CHATBOT CỦA ĐỀ TÀI 52](#_Toc11966306)

[3.1 Phân tích yêu cầu đề tài 52](#_Toc11966307)

[3.2 Tổng quan về ứng dụng 56](#_Toc11966308)

[3.3 Xây dựng cơ sở tri thức dựa vào Ontology 59](#_Toc11966310)

[3.3.1 Lý thuyết về web ngữ nghĩa 59](#_Toc11966311)

[3.3.2 Ontology 60](#_Toc11966312)

[3.3.2.1 Khái niệm 60](#_Toc11966313)

[3.3.2.2 Cấu trúc 61](#_Toc11966314)

[3.3.3 Mô hình đề xuất cho hệ thống hỏi đáp tập trung vào câu hỏi “what” và “how” 62](#_Toc11966315)

[3.3.4 Khả năng mở rộng 68](#_Toc11966316)

[3.3.5 Những lợi ích trong mô hình đã đề xuất 69](#_Toc11966317)

[3.4 Xử lí tin nhắn đầu vào và xây dựng phản hồi với RASA 70](#_Toc11966318)

[3.4.1 Giới thiệu chung về Rasa 70](#_Toc11966319)

[3.4.2 Các thành phần chính của Rasa 71](#_Toc11966320)

[3.4.2.1 Mô hình (Architecture) 71](#_Toc11966321)

[3.4.2.2 Hành động (Action) 72](#_Toc11966322)

[3.2.2.3 Rasa NLU (Xử lý ngôn ngữ tự nhiên) 73](#_Toc11966323)

[3.2.2.4 Chính sách (Policy) 73](#_Toc11966324)

[3.4.3 Rasa NLU 73](#_Toc11966325)

[3.4.3.1 Pipeline 74](#_Toc11966326)

[3.4.3.2 Phân loại ý định và rút trích thực thể 76](#_Toc11966327)

[3.2.4 Rasa Core 86](#_Toc11966328)

[CHƯƠNG 4: CÀI ĐẶT ỨNG DỤNG 90](#_Toc11966329)

[4.1 Quá trình chuẩn bị 94](#_Toc11966330)

[4.1.1 Về dữ liệu ontology 94](#_Toc11966331)

[4.1.2 Về dữ liệu training xử lí câu hỏi 94](#_Toc11966332)

[4.2 Quá trình cài đặt chi tiết 96](#_Toc11966333)

[4.2.1 Cài đặt ontology 96](#_Toc11966334)

[4.2.2 Cài đặt RASA 99](#_Toc11966335)

[4.2.2.1 Cài đặt Rasa NLU 99](#_Toc11966336)

[4.2.2.2 Cài đặt Rasa Core: 103](#_Toc11966337)

[4.2.3. Kết nối ontology với RASA sử dụng thư viện Owlready2 107](#_Toc11966338)

[4.2.4 Cài đặt giao diện người dùng 114](#_Toc11966339)

[4.3 Phát hành, công bố sản phẩm 119](#_Toc11966340)

[4.3.1 Công bố ontology 120](#_Toc11966341)

[4.3.1.1 Công bố ontology để mọi người có thể vào xem 120](#_Toc11966342)

[4.3.1.2 Công bố ontology để mọi người có thể truy xuất và chỉnh sửa dữ liệu 125](#_Toc11966343)

[4.3.2 Phát hành ứng dụng 126](#_Toc11966344)

[1. API backend 126](#_Toc11966345)

[2. Giao diện người dùng 127](#_Toc11966346)

[CHƯƠNG 5: THỰC NGHIỆM VÀ ĐÁNH GIÁ SẢN PHẨM CHATBOT 128](#_Toc11966347)

[5.1 Đánh giá thuật toán Rasa NLU 128](#_Toc11966348)

[5.2 Chỉ số chất lượng hội thoại (dialogue quality metric) 132](#_Toc11966349)

[5.3 Mức độ hài lòng của người dùng (users' satisfaction) 133](#_Toc11966350)

[5.3.1 Thời gian hồi đáp 134](#_Toc11966351)

[5.3.2 Độ phù hợp của câu trả lời 136](#_Toc11966352)

[5.3.3 Độ dễ sử dụng: 137](#_Toc11966353)

[5.3.4 Giao diện người dùng 139](#_Toc11966354)

[5.3.5 Sự hài lòng của người dùng sau khi sử dụng chatbot 140](#_Toc11966355)

[CHƯƠNG 6: TỔNG KẾT, ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN 142](#_Toc11966356)

[6.1 Kết quả đạt được 142](#_Toc11966357)

[6.1.1 Về lý thuyết 142](#_Toc11966358)

[6.1.2 Về thực hành 143](#_Toc11966359)

[6.2 Hạn chế 143](#_Toc11966360)

[6.3 Hướng phát triển 143](#_Toc11966361)

[DANH MỤC THAM KHẢO 145](#_Toc11966362)

# DANH MỤC BẢNG

[Table 1: Phân tích các yếu tố của từng kĩ thuật 48](#_Toc10751688)

[Table 2 So sánh 2 công cụ xây dựng chatbot là DialogFlow và RASA 71](#_Toc10751689)

[Table 3: Thống kê số lượng dữ liệu ontology đã chuẩn bị 94](#_Toc10751690)

# DANH MỤC HÌNH ẢNH

[Hình 1. 1 Khảo sát được hỏi trên 1000 người dùng internet về vấn đề của họ đối với các dịch vụ online mà họ đang sử dụng ví dụ như search engine, websites, mobile apps… (Nguồn tham khảo: the 2018 State of chatbot Report [25]) 25](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500520)

[Hình 2. 1 Một ví dụ về Parse tree 32](#_Toc11500462)

[Hình 2. 2 Sơ đồ tổng quát của một mô hình QAS 33](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500463)

[Hình 2. 3 Các tiêu chí phân loại một QAS 35](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500464)

[Hình 2. 4 Các tiêu chí phân loại một QAS 35](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500465)

[Hình 2. 5 Sơ đồ các phân loại QAS dựa trên miền kiến thức ứng dụng 36](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500466)

[Hình 2. 6 Sơ đồ các phân loại QAS dựa trên miền kiến thức ứng dụng 36](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500467)

[Hình 2. 7 Sơ đồ các phân loại QAS dựa trên loại câu hỏi 37](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500468)

[Hình 2. 8 Sơ đồ các phân loại QAS dựa trên loại câu hỏi 37](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500469)

[Hình 2. 9 Sơ đồ phân loại QAS dựa trên loại phương pháp phân tích được thực hiện trên câu hỏi. 41](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500470)

[Hình 2. 10 Sơ đồ phân loại QAS dựa trên dữ liệu nguồn 45](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500471)

[Hình 2. 11 Sơ đồ phân loại QAS dựa trên những dạng câu trả lời được tạo ra bởi QAS. 49](E:\\Chatbot\\BCLV_2.0_main.docx" \l "_Toc11500472)

[Hình 3. 1 Biểu đồ tròn thống kê các phần trăm theo loại câu hỏi về Adobe Photoshop dựa trên lần xuất hiện đầu tiên của từ để hỏi trong câu hỏi (tổng 1,322 câu hỏi) [10] 53](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535811)

[Hình 3. 2 Mô hình hệ thống của ứng dụng 58](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535812)

[Hình 3. 3Mô hình ontology đề xuất cho đề tài 62](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535813)

[Hình 3. 4 Ví dụ sử dụng ontology đề xuất với câu hỏi “What is paragraph style?” 66](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535814)

[Hình 3. 5 Ví dụ sử dụng ontology đề xuất với câu hỏi “How to install Photoshop CS5 on Ubuntu?” 67](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535815)

[Hình 3. 6 Ví dụ sử dụng ontology đề xuất với câu hỏi “How to paint 3d?” 68](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535816)

[Hình 3. 7 Mô hình hoạt động của RASA 72](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535817)

[Hình 3. 8 Ví dụ về pipeline 74](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535818)

[Hình 3. 9 Luồng hoạt động của pipeline với 3 thành phần 75](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535819)

[Hình 3. 10 Ví dụ về supervised Embeddings với câu how to paint 3d image? 77](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535820)

[Hình 3. 11 Quy tắc cơ bản để chọn trình phân loại ý định 79](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535821)

[Hình 3. 12 Ví dụ về dữ liệu huấn luyện cho intent what\_object 81](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535822)

[Hình 3. 13 Quy tắc cơ bản của việc chọn thành phần ner 84](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535823)

[Hình 3. 14 output sau khi dùng tool đánh giá pipeline của Rasa NLU 85](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535824)

[Hình 3. 16 Mô hình Rasa core 86](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535825)

[Hình 3. 15 Các thành phần của RASA Core 86](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535826)

[Hình 3. 17 Các thành phần của RASA Core 88](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11535827)

[Hình 4. 1 Hình ảnh của dữ liệu được đánh nhãn 95](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537396)

[Hình 4. 2 Cấu trúc các class trong ontology 96](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537397)

[Hình 4. 3 Ành chụp dữ liệu ontology chuẩn bị cho câu hỏi “how mack local adjustments in camera raw”. 97](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537398)

[Hình 4. 4 Ảnh chụp dữ liệu ontology chuẩn bị cho process “Apply local adjustments with adjustment brush tool” là một process của câu hỏi how trong hình 4.6 98](#_Toc11537399)

[Hình 4. 5 Đoạn code cấu hình pipeline 99](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537400)

[Hình 4. 6 Dữ liệu train đã đánh nhãn cho ý định ask\_how 100](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537401)

[Hình 4. 7 Các slots và intents 101](#_Toc11537402)

[Hình 4. 9 Một số mẫu hội thoại định nghĩa sẵn 102](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537403)

[Hình 4. 10 Ảnh chụp độ tin cậy của câu hỏi hello, tell me how to installs photoshop khi sử dụng regex features. 102](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537404)

[Hình 4. 11 Ảnh chụp độ tin cậy của câu hỏi hello, tell me how to installs photoshop khi không sử dụng regex features. 103](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537405)

[Hình 4. 12 Đoạn code sử dụng MemoizationPolicy 103](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537406)

[Hình 4. 13 Đoạn code sử dụng keras policy và fallback policy 104](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537407)

[Hình 4. 14 Data stories 106](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537408)

[Hình 4. 15 Ảnh chụp kết quả trả về từ ontology cho câu hỏi what is layer. 108](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537409)

[Hình 4. 16 Hàm chung để truy vấn cho câu hỏi What với đầu vào là danh sách các Object và danh sách các Context dạng mảng các chuỗi string 108](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537410)

[Hình 4. 17 Hàm chung để truy vấn cho câu hỏi How với đầu vào là danh sách các Object, danh sách các Context và danh sách các operation dạng mảng các chuỗi string 111](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537411)

[Hình 4. 18 Ảnh chụp kết quả trả về từ ontology cho câu hỏi How to download photoshop CC 112](#_Toc11537412)

[Hình 4. 19 Hàm chung lấy các Task của Process dựa theo tên của process 113](#_Toc11537413)

[Hình 4. 20 Ảnh chụp kết quả trả về từ ontology cho process Projection Panting 113](#_Toc11537414)

[Hình 4. 21 Ảnh chụp màn hình giao diện chào hỏi và trả lời câu hòi của người dùng có trả về hình ảnh minh họa. 115](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537415)

[Hình 4. 22 Một phần đoạn code gọi API từ server backend 116](#_Toc11537416)

[Hình 4. 23 Ảnh chụp màn hình giao diện chat khi người dùng hỏi một câu hỏi how gồm nhiều cách xử lý khác nhau. 117](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537417)

[Hình 4. 24 Ảnh chụp màn hình giao diện chatbot trả lời cách xử lí gồm 4 bước, trả lời câu hỏi có video hướng dẫn và trả lời câu hỏi thiếu dữ liệu. 118](#_Toc11537418)

[Hình 4. 25 Ảnh chụp giao diện chatbot đưa ra câu trả lời có hình ảnh sau khi người dùng xác nhận lại câu hỏi thiếu dữ kiện ở hình 4.18 và chào kết thúc đoạn hội thoại 119](#_Toc11537419)

[Hình 4. 26 Thông tin truy cập FPT 121](#_Toc11537420)

[Hình 4. 27 Cấu trúc thư mục hosting (mở bằng FileZilla) 121](#_Toc11537421)

[Hình 4. 28 Ảnh chụp cấu hình chi tiết file .htaccess 122](#_Toc11537422)

[Hình 4. 29 Thông tin domain đăng kí trên PURL 123](#_Toc11537423)

[Hình 4. 30 Hình ảnh khi mở đường link trên web 124](#_Toc11537424)

[Hình 4. 31 Hình ảnh khi mở đường link bằng công cụ protégé 125](#_Toc11537425)

[Hình 4. 32 Một số API truy cập ontology hiện có 126](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537426)

[Hình 4. 33 Ảnh chụp trang firebase quản lý project ChatbotUI mà nhóm đã đăng kí và deploy 127](#_Toc11537427)

[Hình 4. 34 Kết quả đánh giá trình phân loại ý định: Tensorflow classifier 129](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537428)

[Hình 4. 35 Ma trận kết quả dự đoán với từng ý định người dùng 131](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537429)

[Hình 4. 36 Biểu đồ phân phối kết quả độ chính xác dự đoán đúng /sai tensorflow\_embedding 131](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537430)

[Hình 4. 37 Biểu đồ khảo sát đánh giá của người dùng về chất lượng hội thoại (dialogue quality metric) 132](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537431)

[Hình 4. 38 Biểu đồ thống kê đánh giá của người dùng về thời gian hồi đáp trên thang điểm 5 134](#_Toc11537432)

[Hình 4. 39 Thời gian phản hồi một số trường hợp với hệ thống hiện tại khi thực hiện chạy trực tiếp trên server. 135](#_Toc11537433)

[Hình 4. 40 Biểu đồ thống kê đánh giá của người dùng về mức độ phù hợp của câu trả lời trên thang điểm 5 136](#_Toc11537434)

[Hình 4. 41 Biểu đồ thống kê đánh giá của người dùng về độ dễ sử dụng của chatbot trên thang điểm 5 138](C:\\Users\\Ly Ly\\Downloads\\BCLV_2.1_main.docx" \l "_Toc11537435)

[Hình 4. 42 Biểu đồ thống kê đánh giá của người dùng về độ dễ sử dụng của chatbot trên thang điểm 5 139](#_Toc11537436)

[Hình 4. 43 Biểu đồ thống kê đánh giá của người dùng về sự hài lòng thoải mái khi nói chuyện cùng chatbot trên thang điểm 5. 140](#_Toc11537437)

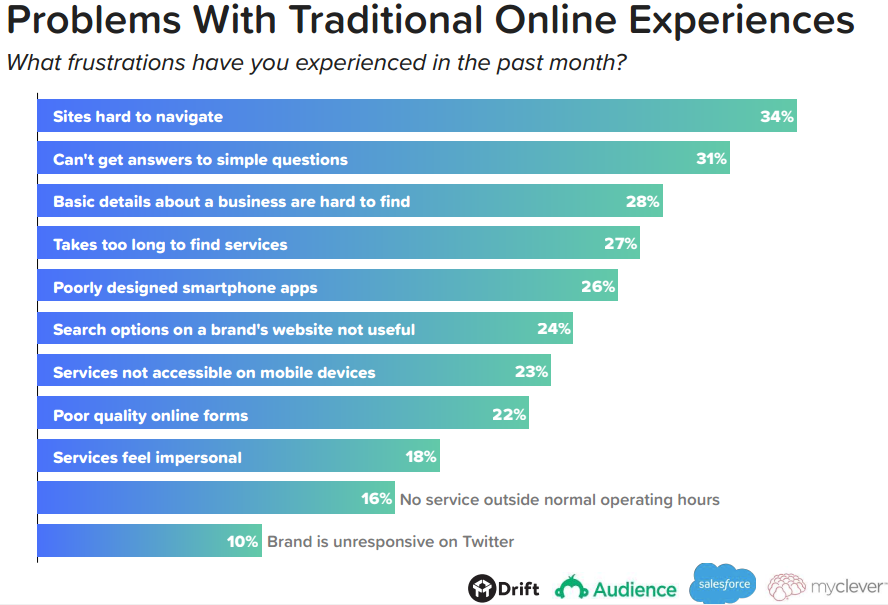
# DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
|  | **Tên đầy đủ** | **Ý nghĩa** |
| AI | Artificial intelligence | Trí thông minh nhân tạo. |
| NLP | Natural language processing | Xử lí ngôn ngữ tự nhiên. |
| NLU | Natural language understanding | Bài toán nhằm hiểu nghĩa của ngôn ngữ tự nhiên. |
| ML | Machine leaining | Máy học. |
| QAS | Question answering System | Hệ thống hỏi đáp. |
| KBS | Knowledge Base | Nguồn kiến thức cơ sở. |
| API | Application Programming Interface | Giao diện lập trình ứng dụng, phần mềm trung gian cho phép các ứng dụng giao tiếp với nhau. |

# CHƯƠNG 1: GIỚI THIỆU

*Mục tiêu của chương một là để giới thiệu đề tài và lý do của việc chọn lựa đề tài. Tiếp đến là phần trình bày mục tiêu cần đạt được của luận văn. Cuối cùng là trình bày về bố cục của toàn bộ cuốn luận.*

* 1. **Giới thiệu đề tài:**

Trong thời đại Internet phát triển như hiện nay, hầu hết các công ty phần mềm đều cần có một trang thông tin điện tử kèm theo dịch vụ hỏi đáp trực tuyến, nơi mà khách hàng (cả người đã từng sử dụng dịch vụ lẫn những khách hàng tiềm năng) có thể lên trang này hỏi thêm về thông tin công ty, dịch vụ, các điều khoản, chính sách, cách sử dụng, …

Hình 1. Khảo sát được hỏi trên 1000 người dùng internet về vấn đề của họ đối với các dịch vụ online mà họ đang sử dụng ví dụ như search engine, websites, mobile apps… (Nguồn tham khảo: the 2018 State of chatbot Report [31])

Theo kết quả khảo sát ở Hình 1.1, có thể thấy dịch vụ online truyền thống đang không đáp ứng đủ nhu cầu của người dùng, một số vấn đề dẫn đầu như *Sites hard to navigate (*khó điều hướng trang) – 34%, *can’t get answer to simple question (*không lấy được câu trả lời cho câu hỏi đơn giản*)* – 31%, *basic details about a business are hard to find (*thông tin cơ bản của 1 doanh nghiệp như là địa chỉ, số điện thoại, thì khó tìm*)*,… Với những vấn đề trên cùng xu hướng thế giới hiện nay đang hướng đến chatbot dưới hình thức một cô nhân viên chăm sóc khách hàng tự động, có khả năng ghi nhớ chính xác, không bị cảm xúc chi phối trong quá trình hỗ trợ khách hàng, có thể làm việc 24/7 không ngừng nghỉ và trả lời thắc mắc cho nhiều khách hàng một lúc nhanh chóng, giúp doanh nghiệp tiết kiệm rất nhiều chi phí đào tạo, kiểm tra, trả lương cho nhân viên chăm sóc khách hàng. Bên cạnh đó, web ngữ nghĩa với rất nhiều những lợi ích to lớn về lâu dài, đó là tạo nên một cơ sở tri thức có ngữ nghĩa khiến máy tính có thể hiểu được từ đó giúp cho việc tìm kiếm thông tin trên internet trở nên nhanh chóng và chính xác hơn thay vì chỉ tìm kiếm dựa trên việc so khớp các từ khóa (xem thêm chương 3.1). Web ngữ nghĩa tuy không phải là một lĩnh vực mới nhưng chưa được ứng dụng nhiều. Với những lí do trên, chúng tôi cho rằng việc tìm hiểu, nghiên cứu để kết hợp giữa Ontology (của web ngữ nghĩa – xem thêm ở chương 2.1) với công nghệ chatbot, tạo nên hệ thống hỏi đáp hoàn chỉnh là một chủ đề rất có ý nghĩa và đem lại nhiều giá trị.

Chúng tôi chọn dữ liệu Adobe Photoshop cho môi trường thực nghiệm là vì đây là một phần mềm rất nổi tiếng và phổ biến, trên các diễn đàn số lượng câu hỏi về photoshop khá nhiều và vì đây là một phần mềm về thiết kế đồ họa nên các bước sử dụng nó khá rõ ràng, tài liệu hướng dẫn cũng rất chi tiết. Có rất nhiều loại câu hỏi mà người dùng có thể hỏi như what, who, which, where, why, how … tuy nhiên trong phạm vi của luận văn này chúng tôi tập trung vào câu hỏi what và how (thuộc loại factoid question là câu hỏi đơn giản thuộc về trả lời lý thuyết, cách thực hiện) để tạo tiền đề vững chắc cho chatbot có thể dễ dàng mở rộng ra thêm chức năng xử lí các loại câu hỏi khác [1]. Lấy ví dụ về câu hỏi why, đây là câu hỏi dạng nguyên nhân – kết quả, để trả lời được câu hỏi dạng này, trước tiên chúng ta phải có một cơ sở tri thức chuẩn (đủ để trả lời cho câu hỏi how và what) và sau đó phải làm rất nhiều bước xử lí ngôn ngữ phức tạp khác.

* 1. **Lí do chọn đề tài**

Nhằm mục đích hệ thống hóa các kiến thức của bản thân trong quá trình học tập vào trong một sản phẩm có ý nghĩa thực tế, có ích trong tương lai. Hơn thế nữa, với xu thế hiện nay, chatbot đang là một cụm từ rất nổi, đang dần len lỏi vào các lĩnh vực xã hội như bán hàng, dịch vụ, tư vấn, tra cứu thông tin, … chúng tôi lựa chọn đề tài: “Xây dựng chatbot hỗ trợ trả lời câu hỏi what – how cho các sản phẩm trong lĩnh vực kĩ thuật”.

Vốn là sinh viên khối ngành kĩ thuật, chúng tôi hiểu được nhu cầu tra cứu thông tin về các phần mềm kĩ thuật nhằm hỗ trợ trong quá trình học tập, làm việc là vô cùng lớn. Vì vậy ứng dụng chatbot hy vọng sẽ là một kênh hỗ trợ tốt cho mọi người có thể hỏi đáp thông tin một cách nhanh chóng và chính xác, là bước khởi đầu cho việc tạo nên một mạng lưới thông tin có ngữ nghĩa về các phần mềm kĩ thuật.

* 1. **Mục tiêu đề tài**

Mục tiêu của luận văn là tìm hiểu về hướng nghiên cứu hệ thống hỏi đáp (question – answering system) và ứng dụng các phương pháp phù hợp vào việc xây dựng chatbot hỗ trợ hỏi đáp cho các sản phẩm trong lĩnh vực kĩ thuật (mà chúng tôi sử dụng adobe photoshop làm ứng dụng minh họa), chỉ tập trung vào câu hỏi what, how. Cụ thể:

* Về lí thuyết:

+ Nắm rõ các khái niểm về web ngữ nghĩa và cách xây dựng ontology.

+ Tìm hiểu các công cụ hỗ trợ xây dựng chatbot và so sánh ưu khuyết điểm giữa các công cụ.

+ Tìm hiểu về lĩnh vực xử lí ngôn ngữ tự nhiên để lựa chọn các thuật toán nhằm ứng dụng vào đề tài một cách phù hợp.

+ Tìm hiểu về lĩnh vực hệ thống hỏi đáp (Question – Answering System) và tập trung vào nhánh nghiên cứu chính của đề tài là câu hỏi what và how.

* Về thực hành:

+ Đề xuất mô hình ontology cho hệ thống hỏi đáp tập trung vào câu hỏi what, how và xây dựng nó dựa theo tài nguyên của Adobe Photoshop.

+ Lựa chọn công cụ phù hợp để sử dụng phân tích ý định của người dùng, nhận diện thực thể, từ đó làm backend API Restful Services cho hệ thống chatbot.

+ Tạo ra chatbot hoàn chỉnh chạy trên hệ thống messenger và chắc chắn rằng chatbot có thể bao quát hầu hết trường hợp câu hỏi từ người dùng.

* 1. **Phạm vi đề tài**
* Phạm vi đề tài sẽ tập trung vào một số mục như: Tìm hiểu khái niệm, phân loại hệ thống hỏi đáp và chatbot, từ đó giúp làm rõ 2 khái niệm này.
* Tìm hiểu khái niệm về web ngữ nghĩa, từ đó áp dụng tạo ra cơ sở tri thức cho đề tài xây dựng chatbot cho các sản phẩm phần mềm trong lĩnh vực kĩ thuật tập trung vào câu hỏi what và câu hỏi how.
* Tìm hiểu các khái niệm, thuật toán nhằm phân tích câu hỏi của người dùng, từ đó xây dựng phản hồi phù hợp áp dụng vào đề tài.
  1. **Bố cục luận văn**
* Chương 1: Giới thiệu:
  + Khái quát chung về đề tài và lí do chọn đề tài.
  + Mục tiêu hướng đến của đề tài sau khi hoàn thành.
* Chương 2: Hệ thống Hỏi – đáp và chatbot:
  + Tổng quan về QAS, khái niệm và phân loại.
  + Giới thiệu sơ về chatbot nhằm làm rõ khái niệm QAS và chatbot.
* Chương 3: Phân tích và thiết kế giải pháp cho ứng dụng chatbot hỗ trợ hỏi đáp trong lĩnh vực kỹ thuật
  + Phân tích rõ hơn về đề tài và mô hình hệ thống ứng dụng.
  + Lý thuyết về web ngữ nghĩa và ontology.
  + Mô hình ontology đề xuất cho hệ thống hỏi đáp.
  + Tổng quan về công cụ RASA sử dụng để phân tích dữ liệu đầu vào và xây dựng phản hồi.
* Chương 4: Cài đặt ứng dụng
  + Quá trình xây dựng và publish cơ sở tri thức sử dụng ontology.
  + Quá trình xây dựng và publish phần server xử lí tin nhắn đầu vào và tạo phản hồi.
  + Quá trình xây dựng và publish phần giao diện người dùng dưới dạng web.
  + Kết nối 3 thành phần trên qua Restful API service.
* Chương 5: Đánh giá, thực nghiệm
* Trình bày kết quả đánh giá model sử dụng cho hai thuật toán nhận diện ý định (intent-classification) và nhận diện thực thể (name entity recognization)
* Trình bày kết quả đánh giá chatbot sau khi thực hiện khảo sát trên thực tế cho một số người dùng thử và nhận xét.
* Chương 6: Tổng kết, đánh giá và hướng phát triển
  + Trình bày kết quả và kinh nghiệm đạt được sau khi làm đề tài và nêu những vấn đề, khó khăn đã gặp phải, đề xuất hướng khắc phục. Bên cạnh đó, phần này cũng nêu lên hướng phát triển của đề tài và những đề xuất nhằm cải tiến ứng dụng trong tương lai.

# CHƯƠNG 2: HỆ THỐNG HỎI ĐÁP (QUESTION – ANSWERING SYSTEM) VÀ CHATBOT

*Mục tiêu của chương hai là khái quát lý thuyết về một hệ thống hỏi đáp, các cách phân loại và nêu rõ đề tài của nhóm thuộc loại nào. Sau đó là khái quát về chatbot và phân tích mối liên quan giữa hệ thống hỏi đáp và chatbot, liệu nó là hai khái niệm giống nhau, khác nhau hay bỗ trợ cho nhau?*

## 2.1 Hệ thống hỏi đáp – Question answering system (QAS)

#### 2.1.1 Một số khái niệm cơ bản

Để giúp mọi người dễ dàng theo dõi, ở phần này chúng tôi xin nêu ra một số khái niệm cơ bản cũng như những thuật ngữ chuyên ngành mà sẽ nhắc lại ở các phần khác của cuốn luận.

#### 2.1.1.1 Truy xuất thông tin - Information Retrieve (IR)

IR là ngành khoa học nghiên cứu về việc tìm kiếm truy xuất thông tin từ tài liệu hoặc databases. Với một công cụ IR, một từ khóa hay một cụm từ khóa bất kì người dùng đưa ra thì hệ thống IR sẽ tìm kiếm và đưa ra một tập các tài liệu phù hợp – những tài liệu có thông tin liên quan đến từ khóa ban đầu [2]. Những tài liệu này sẽ được sắp xếp theo thứ tự phù hợp tùy theo thuật toán của mỗi công cụ, thường là theo thời gian, số lượng từ khóa, từ khóa có trong tiêu đề, hag tag…

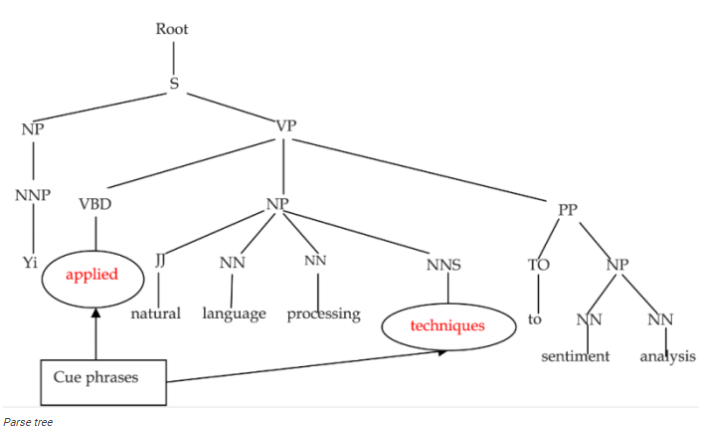
IR thường được chia thành các hướng nghiên cứu về công cụ tìm kiếm (Search Engine như Google, Yahoo, Bing…), hệ thống tư vấn (Recommender System - Hệ thống gợi ý thường được dùng trong các web so sánh, hay các web thương mại, dịch vụ, bất động sản)…

#### 2.1.1.2 Xử lí ngôn ngữ tự nhiên - Natural language processing (NLP)

Theo Wikipedia [3], NLP là một lĩnh vực con của ngành khoa học máy tính (Computer Science), khoa học về thông tin, dữ liệu (Information Engineering) và trí thông minh nhân tạo (Artificial Intelligence) quan tâm đến sự tương tác giữa máy tính và ngôn ngữ tự nhiên của con người, lĩnh vực này nghiên cứu cách để chương trình máy tính xử lí và phân tích một số lượng lớn dữ liệu ngôn ngữ tự nhiên.

Để xử lí ngôn ngữ tự nhiên cũng như để xây dựng một QAS mà sẽ trình bày ở phần tiếp theo, chúng ta sẽ cần phải hiểu về những thuật ngữ cơ bản sau:

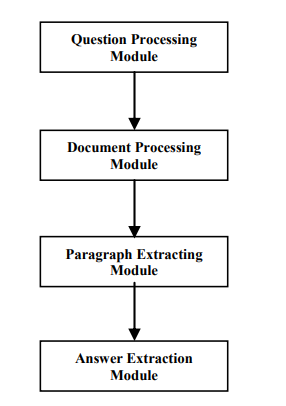
* **Tiền xử lí dữ liệu (Pre – processing)**: là quá trình tiền xử lí dữ liệu, xóa bỏ những kí tự không cần thiết (như is, a, an, the …), tách từ (word segmentation), tích hợp (integreation), chuyển đổi (transformation) và giảm số chiều (reduction). Quá trình này giúp chuyển một câu đầy đủ ngữ pháp thành một câu ngắn gọn hơn nhưng vẫn đầy đủ ý. Ví dụ với câu: *I want to go to Nha Trang at weekend* qua quá trình này sẽ còn *I want go Nha Trang weekend*. Hay câu *Yi applied natural language processing techniques to sentiment analysis* thì sẽ còn *Yi apply natural language process technique sentiment analysis*
* **Phân tích từ loại (Parser)**: là quá trình phân tích ngữ pháp của câu, bước này thường được thực hiện ngay sau bước pre – processing. Parser bao gồm có Part Of Speech (POS) – gán nhãn từ loại, gán nhãn danh từ, động từ, tính từ, Pharse – gán nhãn cụm từ, một số từ sẽ có ý nghĩa khác khi đi chung với nhau, Grammatical Relation – gán nhãn quan hệ ngữ pháp và Parse Tree – gán nhãn cây cú pháp. Lấy lại ví dụ cho câu *Yi applied natural language processing techniques to sentiment analysis*. Qua bước POS ta sẽ có apply là VBD (động từ ở thì quá khứ), techniques là NNS (danh từ số nhiều). Pharse ở đây có NP tức cụm danh từ - natural language processing techniques, cây Parse tree của câu trên được biểu diễn như hình bên dưới:



Hình 2. 1 Một ví dụ về cây ngữ nghĩa (Parse tree)

* **Chiết xuất thông tin (Information Extraction)**: là tiến trình trích xuất thông tin có cấu trúc từ dữ liệu phi cấu trúc/ bán cấu trúc.
* **Nhân diện tên thực thể (Name Entity Recognition** **- NER)**: là quá trình trích xuất những thông tin cần thiết từ đoạn văn bản, những thông tin này có thể là tên người, tên tổ chức, các từ chỉ nội thất hay xe cộ, …
* **Stemming and Lemmatization:** Theo [4]: “*Trong quá trình xử lý ngôn ngữ tự nhiên, sẽ có lúc bạn muốn chương trình của bạn có thể nhận ra rằng “ask” và “asked” chỉ là một động từ thuộc nhiều thì khác nhau. Đó là ý tưởng của việc giảm các hình thái khác nhau của một từ vào từ gốc. Các từ có nguồn gốc từ nhau có thể được ánh xạ tới một từ hoặc ký hiệu trung tâm, đặc biệt nếu chúng có cùng ý nghĩa cốt lõi*”. Stemming và lemmatization là hai cách làm khác nhau, tuy nhiên mục đích của chúng thì chỉ có một, đó là biến các từ có cùng ý nghĩa nhưng khác hình thái về chung một hình thái duy nhất.

#### 2.1.1.3 Tổng quan về QAS

QAS là hệ thống mà có thể tìm kiếm câu trả lời dựa trên câu hỏi của người dùng. QAS là sự kết hợp của IR và NLP, nó là một hệ thống để tìm kiếm, truy xuất thông tin qua việc xử lí ngôn ngữ tự nhiên, từ đó tìm ra duy nhất một câu trả lời hay phản hồi phù hợp nhất. Một QAS thường được cấu thành như sau [5]:

Hình 2. Sơ đồ tổng quát của một mô hình QAS [5]

Trong đó:

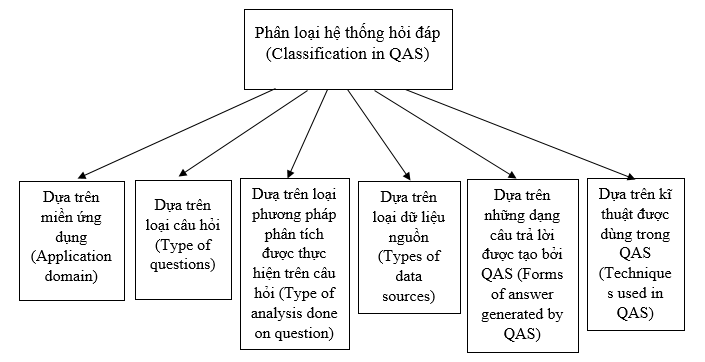
**Question Processing Module**: đây là module có nhiệm vụ chuyển đổi từ ngôn ngữ tự nhiên thành các thông tin cần thiết cho việc tạo câu truy vấn dùng trong bước tiếp theo. Nó cũng có nhiệm vụ phát hiện loại câu trả lời mà người dùng muốn ví dụ với câu *where was Obama born?* thì loại câu trả lời mong đợi là một địa điểm.

**Document Processing Module**: là tiến trình truy xuất dữ liệu từ tập dữ liệu ban đầu mà có thể chứa câu trả lời cho câu truy vấn từ bước trên. Nó bao gồm thuật toán tạo câu truy vấn (query generation algorithm) và công cụ tìm kiếm từ (text search engine). thuật toán tạo câu truy vấn sẽ chuyển những thông tin, từ khóa chính của người dùng để tạo ra câu truy vấn chứa những ràng buộc thích hợp. Câu truy vấn này sẽ được đưa qua cho công cụ tìm kiếm từ để lấy được tập dữ liệu mà có vẻ như có chứa câu trả lời.

**Paragraph Extraction Module**: từ những dữ liệu có vẻ như chứa câu trả lời lấy được từ bước trên, module này sử dụng một vài thuật toán (tùy thuộc vào yêu cầu hệ thống) để so sánh, gán nhãn, tính điểm và cuối cùng đưa ra đoạn có điểm cao nhất.

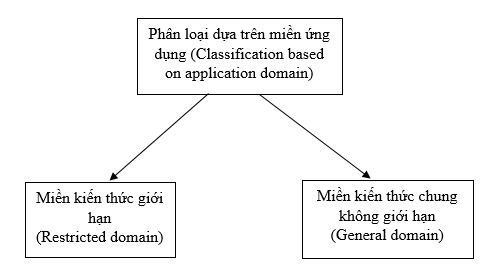
**Answer extraction Module**: Từ đoạn có điểm số cao nhất, module này sẽ phân tích và chiết xuất ra câu trả lời phù hợp, ví dụ như ở phần Question Processing Module ta biết loại câu trả lời cần đưa ra là một địa điểm, vậy ở module này ta chỉ cần trả lời địa điểm mà Obama sinh ra với ngữ pháp thích hợp – He was born in Hawai, US.

### 2.1.2 Phân loại

Có nhiều tiêu chí phân loại QAS, chúng tôi sẽ trình bày tất cả các tiêu chí và các loại trong từng tiêu chí. Cuối mỗi tiêu chí chúng tôi sẽ trình bày phân loại của đề tài và lí do tại sao đề tài lại được xếp vào phân loại đó*.*

Hình 2. 3 Các tiêu chí phân loại một QAS

#### 2.1.2.1 Phân loại dựa trên miền kiến thức ứng dụng (Classification based on application domain)

 Tiêu chí này dựa trên miền tri thức mà QAS tập trung vào, gồm 2 loại là miền kiến thức không giới hạn (general domain) và miền kiến thức giới hạn (restricted domain).

Hình 2. 5 Sơ đồ các phân loại QAS dựa trên miền kiến thức ứng dụng

##### 2.1.2.1.1 Miền kiến thức không giới hạn (General domain QAS)

General domain QAS là phân loại mà miền kiến thức không có giới hạn. Câu hỏi của người dùng có thể là bất cứ thứ gì từ lịch sử, địa lí, xã hội, …

* + Ưu điểm:
* Phù hợp với lượng lớn người dùng. Người dùng có thể hỏi bất kì câu hỏi nào mà họ muốn.
* Không yêu cầu một kho kiến thức chuyên môn cụ thể mà thường sẽ lấy kiến thức thông thường từ nhiều nguồn khác nhau. Có thể tận dụng Wikipedia hoặc các bài báo làm nguồn dữ liệu.
* Người dùng không cần phải biết chính xác keywords khi hỏi.
  + Nhược điểm
* Chất lượng câu trả lời thấp. Tỉ lệ câu trả lời phù hợp với người dùng không cao, đôi khi câu này phù hợp với người này nhưng với người khác thì không.
* Khó trả lời tốt với những câu hỏi mang kiến thức chuyên môn cao.

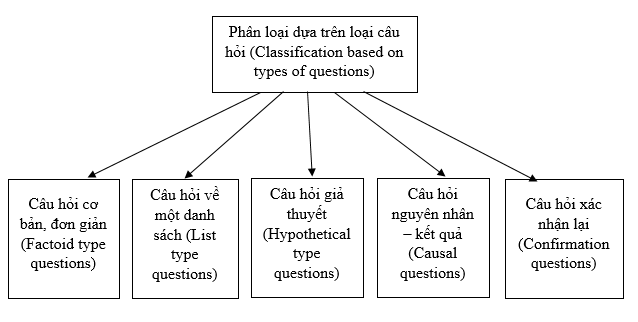
##### 2.1.2.1.2 Miền kiến thức giới hạn (Restricted domain QAS)

Restricted domain QAS là phân loại mà miền kiến thức là hạn chế. Chỉ gói gọn trong một lĩnh vực chuyên môn nhất định.

* + Ưu điểm:
* Phù hợp với những người dùng có chuyên môn thuộc lĩnh vực mà QAS hướng tới. Họ có thể có được câu trả lời cụ thể và chính xác, không bị nhập nhằng từ đồng nghĩa với các lĩnh vực khác.
* Chất lượng câu hỏi cao, cụ thể do hướng đúng đối tượng.
  + Nhược điểm:
* Chỉ có thể trả lời một lượng hạn chế các câu hỏi.

=> *Sản phẩm của chúng tôi thuộc Restricted domain QAS, miền kiến thức tập trung ở đây là kiến thức liên quan đến các phần mềm kĩ thuật (cụ thể là AdobePhotoshop), hướng đến cộng đồng người dùng có sử dụng sản phầm phần mềm đó.*

#### 2.1.2.2 Phân loại dựa trên loại câu hỏi (Classification based on types of questions)

Tiêu chí này nhằm phân loại QAS theo độ phức tạp của loại câu hỏi đầu vào, đi từ đơn giản đến phức tạp ta có: factoid type questions, list type questions, hypothetical type questions, causal questions, confirmation questions*.*

Hình 2. 7 Sơ đồ các phân loại QAS dựa trên loại câu hỏi

##### 2.1.2.2.1 Câu hỏi cơ bản, đơn giản (Factoid type questions - what, when, which, who, how)

Là những câu hỏi đơn giản, ngắn gọn, thực tế. Yêu cầu câu trả lời là ngắn gọn, chính xác và đúng đối tượng – ví dụ: *who is CEO of facebook? - Mark Zuckerberg*.

* Ưu điểm:
* Loại câu trả lời mong đợi cho câu hỏi này thường là tên thực thể, có thể dễ dàng lấy được trong tài liệu bằng một số thuật toán nhận diện thực thể. Loại câu trả lời là tên riêng, địa chỉ, hay thời gian, địa điểm phụ thuộc vào loại câu hỏi who, when, which … vì vậy có thể đạt được độ chính xác khá cao.
* Nhiều QASs hiện tại đạt được hiệu năng khá tốt trong xử lí câu hỏi dạng này.
* Hiện nay đã có một lượng lớn dữ liệu cho câu hỏi dạng này.
* Đối với câu hỏi dạng này, không cần phải thực hiện xử lí ngôn ngữ tự nhiên quá phức tạp để trích xuất câu trả lời.
* Có thể tận dụng Wikipedia hoặc các bài báo làm nguồn dữ liệu.
* Nhược điểm:
* Một vấn đề đang được nghiên cứu hiện nay là việc nhận diện câu hỏi của người dùng là câu hỏi dạng factoid và dạng mở rộng của nó một cách tự động.
* Descriptive type questions: là một nhánh của factoid, dạng câu hỏi yêu cầu tìm kiếm một định nghĩa hoặc mô tả của một sự kiện hay một thực thể. Nó thường bắt đầu với “what is”. Câu trả lời cho câu hỏi này có thể bị nhiễu loạn bởi nó có thể là bất kì sự kiện hoặc thực thể.
* Fuzzy questions: là những câu hỏi nhập nhằng, không thể xử lí thông tin mà người dùng cần một cách chính xác để đưa ra câu trả lời phù hợp, nó thường là những câu hỏi mà có tính từ đi kèm. Ví dụ: ai là người phụ nữ đẹp nhất trong công ty?
* Relationship or information extraction (IE): việc chiết xuất thông tin hoặc các mối quan hệ giữa những thực thể với nhau để tìm ra câu trả lời phù hợp là một vấn đề cần phải suy nghĩ.
* Dialog questions: những câu hỏi thiếu thông tin hoặc là sai cú pháp, làm sao để hệ thống nhận diện được yêu cầu của người dùng để đưa ra phản hồi phù hợp.
* Badly worded questions or ambiguous questions: những câu hỏi mà người dùng sử dụng từ đồng âm, teen code, nhập nhằng như là *what makes him coooool?*

##### 2.1.2.2.2 Câu hỏi về một danh sách (List type questions)

Là dạng câu hỏi mà yêu cầu câu trả lời là một danh sách các thực thể. Có thể ví loại câu hỏi này như factoid question nhưng được hỏi đi hỏi lại nhiều lần, mà mỗi lần lại cần câu trả lời khác nhau. Ví dụ hỏi danh sách những tỉ phú thế giới năm 2019? – Bill gate, Mark Zuckerberg, Phạm Nhật Vượng….

* Ưu điểm:
* Loại câu trả lời mong đợi cho câu hỏi này thường là danh sách các thực thể. Vì vậy có thể đạt được độ chính xác khá cao.
* Có thể sử dụng lại công nghệ cho factoid type question áp dụng vào list type questions.
* Không yêu cầu xử lí ngôn ngữ tự nhiên quá nhiều cho việc trích xuất câu trả lời.
* Khó khăn:
* Cần để ý vấn đề về thiết lập ngưỡng giá trị trả về.

##### 2.1.2.2.3 Câu hỏi giả thuyết (Hypothetical type questions)

Những câu hỏi giả thuyết (hypothetical) này hỏi các thông tin liên quan đến bất cứ sự kiện giả thiết nào. Dạng câu hỏi này thường bắt đầu với “*what would happen if – chuyện gì sẽ xảy ra nếu*”. QASs yêu cầu phải có các kĩ thuật truy xuất kiến thức để tạo câu trả lời. Câu trả lời cho dạng này chỉ là một câu hỏi chủ quan chứ không thể có một đáp án đúng chính xác.

* Ưu điểm:
* Một số người dùng chuyên gia sẽ thích tìm kiếm câu trả lời tối ưu cho câu hỏi giả thuyết, bởi câu hỏi dạng này thường yêu cầu kiến thức rộng và khả năng lập luận tốt.
* Nhược điểm:
* Độ chính xác thấp.
* Công nghệ áp dụng thành công cho factoid type questions thường không áp dụng lại được.
* Độ tin cậy thấp, tùy thuộc vào người dùng và ngữ cảnh.

##### 2.1.2.2.4 Câu hỏi nguyên nhân – kết quả (Causal questions - how, why)

Câu hỏi nguyên nhân – kết quả (causal questions) là dạng câu hỏi mà yêu cầu sự giải thích về một vấn đề, thực thể nào đó. Câu trả lời không đơn giản như factoid type question mà nó đòi hỏi một kĩ thuật cao trong xử lí ngôn ngữ tự nhiên để phân tích văn bản ngữ dụng và trình độ ngôn từ nhằm tạo ra câu trả lời.

* Ưu điểm:
* Những câu hỏi của người dùng đều sẽ liên quan đến những sự kiện và đối tượng cụ thể, do khi hỏi câu hỏi dạng này, người dùng đã có hiểu biết về các sự kiện, đối tượng.
* Nhược điểm:
* Vấn đề trong việc xác định sự liên quan hay tìm câu trả lời đơn nhất cho câu hỏi bởi câu trả lời có thể là một câu, một đoạn văn thậm chí là một tập tài liệu. Một câu trả lời có thể bị hiểu theo nhiều cách khác nhau tùy theo sự phiên dịch. Lấy ví dụ với câu why x took a lecture in class 2 – *Tại sao X lấy bài giảng của lớp 2?* Có thể có 3 cách phiên dịch khác nhau như sau:
  + Why X? – *Tại sao người lấy bài giảng của lớp 2 lại là X?*
  + Why took lecture? – *Tại sao lại lấy bài giảng?*
  + Why in class 2? – *Tại sao lại lấy của lớp 2 mà không phải là lớp 1?*
* Vấn đề liên quan đến hiệu quả truy xuất models trong why QAS là hầu hết các mô hình hiện tại dựa trên bag of words model – một mô hình thường dùng trong phân loại văn bản (text classification), thống kê số lần xuất hiện của các từ trong văn bản. Model này còn nhiều vấn đề trong xử lí các từ đồng nghĩa (synonymy), đồng âm (homonymy), đa nghĩa (polysemy). Hơn nữa, câu trả lời có thể từ vài câu đến vài đoạn và việc nhận định mối quan hệ nghị luận trong tài liệu nguồn là tối quan trọng để có thể tạo ra câu trả lời tốt.

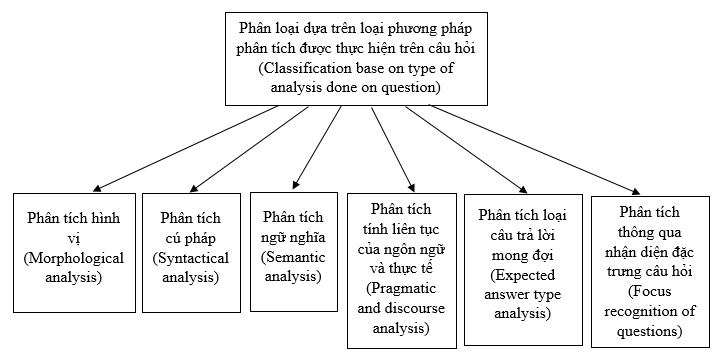
##### 2.1.2.2.5 Câu hỏi xác nhận lại (Confirmation questions)

Câu hỏi xác nhận lại (confirmation question) là những câu hỏi nhằm xác nhận lại thông tin, nó yêu cầu câu trả lời là đúng hoặc sai. Hệ thống yêu cầu phải có cơ chế suy luận, kiến thức rộng và khả năng lập luận tốt để tạo ra câu trả lời.

* Ưu điểm:
* Một số người dùng chuyên gia có thể thích tìm kiếm thông tin mà yêu cầu kiến thức rộng và lập luận tốt để học kiến thức mới.
* Nhược điểm:
* Đây là dạng câu hỏi khá khó, nhiều thách thức đặt ra đối với các dự án QASs cho dạng câu hỏi này.

=> *Hệ thống của chúng tôi tập trung vào câu hỏi What – How thuộc nhánh Factoid type question. Là những câu hỏi cơ bản nhất, tạo nguồn dữ liệu cơ sở để các nghiên cứu sau này có thể mở rộng ra các loại câu hỏi khác.*

#### 2.1.2.3 Phân loại dựa trên loại phương pháp phân tích được thực hiện trên câu hỏi (Classification base on type of analysis done on question)

Có nhiều cách để phân tích nội dung đầu vào của một văn bản. Tiêu chí này phân loại QAS dựa trên loại phương pháp phân tích được dùng, phần này sẽ giải thích ý nghĩa từng phương pháp và ưu khuyết điểm của mỗi loại, sau đó sẽ trình bày ba kĩ thuật chính thường được dùng để phân tích câu hỏi*.*

Hình 2. 9 Sơ đồ phân loại QAS dựa trên loại phương pháp phân tích được thực hiện trên câu hỏi.

##### 2.1.2.3.1 Phân tích hình vị (Morphological analysis)

Phân tích hình vị nhằm mục đích chia tách các từ thành những hình vị độc lập và gán lớp cho mỗi hình vị. Ví dụ “plays”, “playes”, “played” sẽ được phân vào lớp “play”. Phương pháp này sử dụng stemming và lemmatizations – cách biến đổi một từ về dạng nguyên mẫu –như đã giải thích ở phần trước để phân tích hình vị của từ..

* Ưu điểm:
* Những từ ngữ dư thừa sẽ dược loại bỏ trong suốt quá trình truy xuất thông tin.
* Nhược điểm:
* Đôi khi quá trình stemming và lemmatizations sẽ khiến các từ bị sai về mặt ngữ nghĩa. Ví dụ với từ “compute” và “computer” qua quá trình stemming và lemmatizations có thể bị sai khi cùng được phân vào lớp “compute” (điều này là do quá trình stemming thường không quan tâm đến ngữ nghĩa, mà sẽ loại từ đi theo cách thông thường nhất, do đó những từ có đuôi –er như programer, manager sẽ bị bỏ đi chữ r thành programe, manage)

##### 2.1.2.3.2 Phân tích cú pháp (Syntactical analysis)

Là phương pháp phân tích ngữ pháp, loại từ của các từ trong câu hỏi hoặc tài liệu nguồn. Thông thường, một câu sau khi phân tích sẽ gồm các thành phần chính sau (noun – danh từ, verbs – động từ, adjectives – tính từ, adverbs – trạng từ). Đối với loại phân tích này, QASs sẽ tạo ra cây ngữ nghĩa (parse tree – như hình 2.1) sau quá trình xử lí câu hỏi hoặc tài liệu. Hệ thống có loại phân tích này thì sẽ giảm được tập tài liệu tìm kiếm. Ví dụ với câu *which play did xyz work?* Thì hệ thống sẽ tìm những câu có chứa từ “play” dưới dạng là động từ thay vì danh từ.

* Ưu điểm:
* Những từ ngữ dư thừa sẽ dược loại bỏ trong suốt quá trình truy xuất thông tin.
* Nhược điểm:
* Hiện vẫn chưa có bất kì một thuật toán nào có thể phân tích ngữ pháp đúng 100%, vì vậy đôi khi việc phân tích theo cú pháp, ngữ pháp vẫn có thể bị sai.

##### 2.1.2.3.3 Phân tích ngữ nghĩa (Semantic analysis)

Là loại phân tích đoán nghĩa của câu hỏi dựa trên các từ sử dụng trong câu. Nó thường sử dụng cây ngữ nghĩa từ giai đoạn phân tích cú pháp (syntatical analysis) để phiên dịch ra nghĩa của câu. Một trong những nhiệm vụ của giại đoạn phân tích ngữ nghĩa (semantic analysis) là nhận diện và đánh nhãn các đối số trong văn bản (semantic role labeling).

* Ưu điểm:

- Giải quyết vấn đề phân lớp ngữ nghĩa cho câu hỏi và tìm kiếm loại câu trả lời.

- Cung cấp giải pháp tìm kiếm câu trả lời hiệu quả thay vì tìm kiếm chỉ dựa trên từ khóa.

* Nhược điểm:

- Những hệ thống hiện tại mới chỉ phân tích nghĩa ở cấp độ câu chứ chưa có phân tích ở cấp độ tài liệu.

- Để phân tích ngữ nghĩa của câu thì cần phải sử dụng các thuật toán nhận diện tên thực thể, khai thác quan hệ, gán nhãn từ loại,… vì thế độ chính xác của những thuật toán này sẽ ảnh hưởng đến việc phân tích ngữ nghĩa, nhất là với những văn bản có độ khó cao (hiện các thuật toán kể trên vẫn chưa thể chính xác 100%)

##### 2.1.2.3.4 Phân tích tính liên tục của ngôn ngữ và thực tế (Pragmatic and discourse analysis)

Pragmatic analysis: có thể xem đây là loại mở rộng ở cấp cao hơn so với syntactical analysis. Trong khi syntactical analysis là chức năng của một đối số như 1 câu, thì pragmatic analysis là chức năng quan tâm đến cách nói và ngữ cảnh mà câu này được diễn đạt. Ví dụ: “I need a mobile with a good camera. I find oppo in the market. Why should I buy it?” - Ở đây “it” đúng hơn là một cái mobile hãng oppo, chứ không phải là oppo.

Discourse analysis: discourse là một bài tiểu luận hay một cuộc đàm thoại. Với loại phân tích này, hệ thống nhận diện cấu trúc thông qua sự kết nối văn bản như loại mối quan hệ tồn tại giữa các câu (chú thích - elaboration, giải thích - explanation, tương phản - contrast) trong văn bản.

Pragmatic and discourse analysis thường được dùng khi tìm kiếm câu trả lời dài cho câu hỏi phức tạp (complex questions như là why hay how). Nó thực hiện các nhiệm vụ sau đây:

* Anaphora resolution – Thay thế đại từ (It, he, she…) bằng danh từ miêu tả chính xác đại từ đó trong câu.
* Discourse structure recognition – nhận diện kết nối hợp lí giữa các câu trong văn bản. Ví dụ một bài báo sẽ được phân thành chủ đề, câu chuyện chính, những sự kiện trước câu chuyện chính, đánh giá, …. Thường được dùng trong trường hợp câu hỏi ý kiến (opinionated), nguyên nhân – kết quả (causal), xác nhận (yes – no), giả thiết (hypotheical).
  + Ưu điểm:
* Thích hợp cho loại câu hỏi phức tạp như causal question hay hypothetical question. Các quan hệ như giải thích, tương phản tồn tại giữa các câu sẽ giúp việc tìm ra câu trả lời chính xác hơn.
* Giúp ích cho việc suy đoán nghĩa của văn bản.
  + Nhược điểm:
* Công nghệ hiện tại sử dụng inter sentential và intra sentential để cấu trúc câu. Vì vậy, công nghệ vẫn còn khá xa vời với ý tưởng discourse analysis trong văn bản.
* Để phân tích ngữ nghĩa của câu thì cần phải sử dụng các thuật toán nhận diện tên thực thể, khai thác quan hệ, gán nhãn từ loại… vì thế việc discourse analysis càng trở nên khó khăn hơn.

##### 2.1.2.3.5 Phân tích loại câu trả lời mong đợi (Expected answer type analysis)

QASs xác định loại câu trả lời được yêu cầu dựa trên loại câu hỏi. Ví dụ câu hỏi who is author of book X? thì loại câu trả lời mong đợi ở đây là person. Vậy, expected answer type analysis giúp tạo phản hồi cho factoid type question và list type question.

##### 2.1.2.3.6 Phân tích dựa trên nhận diện đặc trưng của loại câu hỏi (Focus recognition of questions)

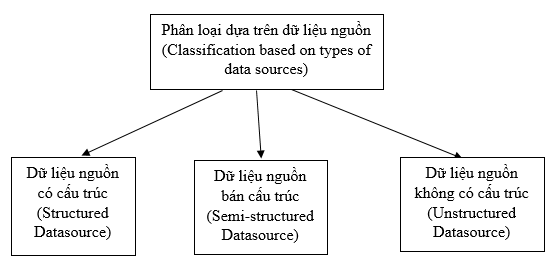
Nhận diện đặc trưng của câu hỏi là rất quan trọng trong việc tìm ra câu trả lời chính xác. Xét câu “if I need mobile with good camera and nice sound quality. I found nokia in market. *Why should I buy Nokia?*” có đặc trưng là các cụm từ “*need mobile, nice sound quality, buy nokia, good camera*”.

* + - Mỗi loại câu hỏi sẽ cần các kĩ thuật xử lí khác nhau để nhận diện đặc trưng. Có nhiều khía cạnh trong việc phân tích ngôn ngữ tự nhiên của câu hỏi như là target extraction, pattern extraction và parsing.
* Để thực hiện các phân tích nêu trên, có 3 phương pháp tiếp cận, đó là:
* Tiếp cận dựa trên thống kê (Statistical based approach): Sử dụng việc thống kê các quan hệ tồn tại trên câu hỏi và tài liệu. Nó bao gồm các lĩnh vực mô hình xác xuất (probabilistic modeling), đại số tuyến tính (linear algebra), lý thuyết thông tin (information theory). Cách tiếp cận này yêu cầu lượng dự liệu lớn để hệ thống có thể học. Khi học, hệ thống có thể tạo ra những kết quả đầy hức hẹn.
* Tiếp cận dựa trên nguyên tắc khớp mẫu (Rule based pattern matching approach): sử dụng luật ngôn ngữ và kiến thức của con người trong tiến trình truy xuất thông tin. Ở cách tiếp cận này, một mô hình xác định trước được xây dựng cho các câu hỏi và câu trả lời.
* Tiếp cận hỗn hợp (Hybrid approach): kết hợp giữa tiếp cận dựa trên thống kê và tiếp cân dựa trên nguyên tắc khớp mẫu.

=> *Ở phạm vị luận văn, chúng tôi sử dụng RASA để phân tích câu hỏi với thuật toán có sẵn của nó, các loại phân tích chúng tôi sử dụng là morphological analysis, syntactical analysis, semantic analysis … và hình thức tiếp cận được sử dụng là tiếp cận dựa trên thống kê (Statistical based approach)*

#### 2.1.2.4 Phân loại dựa trên loại dữ liệu nguồn (Classification based on types of data sources)

Để xây dựng nên một QAS thì dữ liệu nguồn là cái không thể thiếu được. Có 3 loại dữ liệu nguồn: có cấu trúc (structured data source), bán cấu trúc (semi-structured data source) và không có cấu trúc (unstructured data source).



Hình 2. 10 Sơ đồ phân loại QAS dựa trên dữ liệu nguồn

##### 2.1.2.4.1 Dữ liệu nguồn có cấu trúc (Structured data source)

##### Là loại dữ liệu được cấu trúc theo một bộ ngữ nghĩa xác định. Những thực thể tương tự được gom vào trong cùng mối quan hệ. Những thực thể có cùng mối quan hệ thì cũng có những thuộc tính giống nhau. Sự mô tả tất cả thực thể trong một đơn vị gọi là schema.

* + Ưu điểm:
* Độ tin cậy của câu trả lời sẽ cao hơn vì thông tin được lưu có cấu trúc.
* Không yêu cầu phải xử lí ngôn ngữ tự nhiên quá phức tạp.
  + Nhược điểm:
* Lượng thông tin được lưu hạn chế, không nhiều.
* Có thể có sự đối chiếu, tham chiếu trong các nguồn dữ liệu.
* Tốn nhiều công sức để xây dựng nguồn dữ liệu.
* Hệ thống phải chuyển câu hỏi thành câu truy vấn xuống cơ sở dữ liệu.

##### 2.1.2.4.2 Dữ liệu nguồn bán cấu trúc (Semi-structured data source)

Đối với dữ liệu bán cấu trúc, không tồn tại vách ngăn giữa dữ liệu được lưu trữ và schema (mô tả cấu trúc các thực thể trong nguồn dữ liệu). Tức là ta có thể tự mô tả cấu trúc dữ liệu mà không cần tuân theo schema cố định. Ví dụ có schema gồm 1 lớp miêu tả người Person có các thuộc tính (attribute) là name, address, day of birth. Thì đối với dữ liệu có cấu trúc, với mỗi thực thể ta chỉ có thể lưu đúng 3 thông tin là 3 thuộc tính nêu trên nhưng với dữ liệu bán cấu trúc, đối với mỗi thực thể nếu cần thêm bất kì thông tin nào như partposs, phone ta đều có thể ghi vào dữ liệu nguồn.

* Ưu điểm:
* Sự miêu tả thông tin trong dữ liệu nguồn không bị hạn chế bởi schema.
* Cung cấp một cấu trúc linh hoạt để trao đổi dữ liệu giữa các loại cơ sở dữ liệu khác nhau.
* Có thể chuyển đổi từ dữ liệu có cấu trúc sang dữ liệu bán cấu trúc (cho mục đích duyệt web).
* Nhược điểm:
* Có thể có sự đối chiếu, tham chiếu trong các nguồn dữ liệu.
* Tốn nhiều công sức để xây dựng nguồn dữ liệu.

##### 2.1.2.4.3 Dữ liệu nguồn không có cấu trúc (Un-structured data source)

Là loại dữ liệu mà không được cấu trúc theo một ngữ nghĩa nào, dữ liệu có thể tồn tại ở bất cứ định dạng nào. QAS với dữ liệu nguồn không có cấu trúc cần phải áp dụng các thuật toán NLP và IR để tìm ra câu trả lời.

* Ưu điểm:
* Thông tin có thể được thêm và chỉnh sửa một cách dễ dàng.
* Nhược điểm:
* Sự miêu tả thông tin trong dữ liệu không có cấu trúc là một vấn đề lớn.
* Độ tin cậy của câu trả lời không cao.

=> Đối với sản phẩm trong đề tài chúng tôi sử dụng web ngữ nghĩa, với bộ ontology mới do nhóm đề xuất thì đề tài của chúng tôi thuộc loại dữ liệu nguồn có cấu trúc (structured data source)

#### Phân loại dựa trên kĩ thuật được dùng trong QAS (Techniques used in QAS)

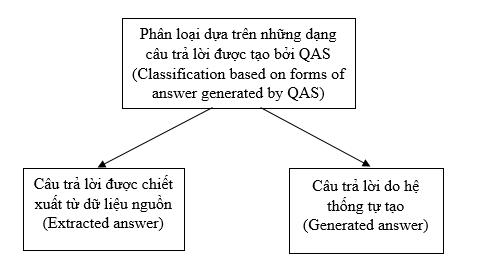
Có 4 kĩ thuật được dùng trong QAS, đó là:

* QAS based on data mining – dựa trên khai thác dữ liệu
* QAS based on information retrieval – dựa trên truy xuất thông tin
* QAS based on natural language understanding – dựa trên phân tích ngôn ngữ tự nhiên
* QAS based on knowledge retrieval and discovery – dựa trên tìm hiểu và khám phá miền kiến thức
* Bảng bên dưới là phân tích các yếu tố của từng kĩ thuật được phân loại theo tài liệu [6], hệ thống của chúng tôi không hoàn toàn tuân theo một loại kĩ thuật nào nhưng nó gần nhất với Knowledge retrieval and discovery, có điều do khả năng và thời gian hạn chế, hệ thống của chúng tôi không bao gồm quá trình tự tạo ra nguồn tri thức.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kỹ thuật | Data mining | Information retrieval | Natural language understanding | Knowledge retrieval and discovery |
| Tìm kiếm (searching) | Tìm kiếm dữ liệu thực tế | Truy vấn dữ liệu thực tế | Truy vấn thông tin có đánh giá chủ quan hoặc dựa trên thực tế | Hiểu, tạo ra nguồn tri thức và tìm kiếm đáp án đúng. |
| Khớp mẫu (matching) | Chính xác | Phù hợp nhất | Phù hợp nhất | Phù hợp và chính xác nhất |
| Kỹ thuật | AI và database | IR và NLP | NLP và NLU | NLU, knowledge acquisition, mining |
| Dạng câu trả lời | Ngắn | Hỗn hợp | Hỗn hợp | Hỗn hợp |
| Loại câu hỏi | Câu hỏi tìm kiếm đơn giản | Wh question – what, where, which, when | Câu hỏi định nghĩa | Câu hỏi phức tạp – how, why, yes-no, … |
| Sự liên quan (relevancy) | Khách quan | Chủ quan | Chủ quan | Chủ quan |
| Kĩ thuật | Pattern matching (khớp mẫu), syntactic analysis | Relevancy ranking (xếp hạng sự liên quan), pattern matching, syntactic analysis | Relevancy ranking, pattern matching, syntactic, semantic analysis | Understanding knowledge, creating knowledge (Hiểu và tạo nguồn tri thức), discourse and pragmatic analysis, application of deductive tecniques (ứng dụng của công nghệ suy diễn) |
| Nguồn tri thức | Database | Syntactic Web | Syntactic and pragmatic web | Semactic and pragmatic web |
| Mô hình dùng trong truy xuất | Bag of word | Bag of word | Bag of concepts | Bag of knowledge |
| Độ tin cậy | Rất tốt do schema được thiết kế bởi các chuyên gia trong lĩnh vực đó | Thấp do có quá nhiều thông tin giả, không chính thống trên web | Thấp | Tốt |

Table 1: Phân tích các yếu tố của từng loại kĩ thuật sử dụng trong QAS.

#### Phân loại dựa trên những dạng câu trả lời được tạo bởi QAS (Classification based on forms of answer generated by QAS)

Tiêu chí này phân loại dựa trên câu trả lời thuộc loại extracted answer (chiết xuất từ dữ liệu nguồn) hay generated answer (hệ thống tự tạo ra sau khi phân tích đầy đủ thông tin).

Hình 2. 11 Sơ đồ phân loại QAS dựa trên những dạng câu trả lời được tạo ra bởi QAS.

##### 2.1.2.6.1 Câu trả lời được chiết xuất từ dữ liệu nguồn (Extracted answer)

* Câu trả lời dưới dạng các câu: tài liệu nguồn phải được chia ra thành cụm những câu riêng biệt. Cụm câu nào được hệ thống đánh giá tốt nhất sẻ dùng để phản hồi cho người dùng. Thông thường nó được dùng cho factoid questions hoặc yes-no type questions. QASs gặp vấn đề trong việc tìm ra ranh giới của cụm câu trả lời từ các loại tài liệu không chính thức như blogs, các trang mạng xã hội, …
* Câu trả lời dưới dạng đoạn văn bản: tài liệu nguồn phải được chia ra thành các đoạn riêng biệt. Đoạn nào được hệ thống đánh giá tốt nhất sẻ dùng để phản hồi cho người dùng. Thường được dùng cho causal questions hoặc hypotherical type questions. Vấn đề lớn nhất cho chiết xuất câu trả lời dạng đoạn văn bản là việc phân tích chủ đề đoạn.
* Câu trả lời dưới dạng đa phương tiện: phản hồi trả về cho người dùng sẽ ở dạng các tệp tin đa phương tiện như audio, video, sound clip.

##### 2.1.2.6.2 Câu trả lời do hệ thống tự tạo (Generated answer)

* Câu trả lời dạng yes – no: câu trả lời sẽ được hệ thống đưa ra thông qua xác minh và biện luận (verification and justification).
* Câu trả lời là ý kiến, thái độ hoặc đánh giá được tạo ra bởi hệ thống đánh giá đối tượng hoặc tính năng của đối tượng.
* Câu trả lời hội thoại: tạo ra câu trả lời cho câu hỏi của người dùng dưới hình thức như một cuộc hội thoại.

=> *Hệ thống của chúng tôi là chatbot vì vậy sẽ trả lời câu hỏi của người dùng dưới hình thức một cuộc hội thoại (của generate answer) kết hợp với câu trả lời dưới dạng các câu (của extracted answer).*

## 2.2 Chatbot

Theo Wikipedia [7], “*chatbot (tên khác là smartbots, talkbot, chatterbot, Bot, IM bot, interactive agent, Conversational interface hay Artificial Conversational Entity) là một chương trình máy tính hoặc trí thông minh nhân tạo thực hiện cuộc trò chuyện bằng giọng nói hay văn bản. Các chương trình như vậy thường được thiết kế để mô phỏng cách con người đối thoại với nhau. Một số chatbot sử dụng hệ thống xử lý ngôn ngữ tự nhiên phức tạp, nhưng nhiều hệ thống đơn giản hơn chỉ quét từ khóa trong đầu vào, sau đó trả lời với từ khóa phù hợp nhất hoặc xây dựng câu trả lời từ cơ sở dữ liệu*”.

Nhiều người thường lầm lẫn QAS và chatbot là một, nhưng theo [8], “*chatbot và QAS có những cách tiếp cận, phương thức, thuật toán và mô hình khác nhau. Chatbot có thể có nhiều chức năng (tùy thuộc vào vấn đề cần giải quyết) hơn là QAS và cũng dễ dàng để xây dựng nhờ các công cụ được xây dựng sẵn như Diaglogflow.ai, wit.ai, LUIS, amazon lex and IBM Watson, … nhưng nó không thể thông minh thật sự mà nó chỉ có thể làm những việc mà người xây dựng cần. Chatbot được biết đến như là tác nhân đàm thoại, một dịch vụ được cung cấp bởi các quy tắc hoặc trí tuệ nhân tạo (một phần nhỏ) mà bạn tương tác thông qua giao diện trò chuyện.”*

Hay theo [9], “*chatbot không chỉ là một QAS mà nó còn thực hiện rất nhiều nhiệm vụ tùy theo chúng ta thiết kế như là chào hỏi, order bánh, order quần áo, đăng kí môn học, … Phần quan trọng nhất trong xây dựng chatbot là thiết kế cuộc đàm thoại. Cuộc đàm thoại xác định giữa người dùng và bot, lý tưởng nhất là cuộc đàm thoại phải luôn được điều khiển bởi bot*”.

Có thể nói chatbot là một QAS với quy mô nhỏ và được xây dựng bằng các công cụ hỗ trợ nhằm tạo nên một cuộc đàm thoại giữa người dùng và bot trên một giao diện chat. Chatbot bỗ trợ cho QAS, khiến hệ thống trở nên thân thiện hơn và có thể thực hiện một số thao tác trên hệ thống nhằm đáp ứng nhu cầu của người dùng.

## 2.3 Kết chương

Chương này đã đưa ra cái nhìn tổng quan về một hệ thống hỏi đáp (QAS) và chatbot. Ứng dụng thực hiện trong đề tài là một chatbot hay một hệ thống hỏi đáp tập trung vào câu hỏi what, how (factoid question), thuộc miền kiến thức hạn chế trong lĩnh vực các sản phẩm phần mềm (restricted domain), sử dụng hai phương pháp phân tích cơ bản là phân tích hình thái (morphological analysis), phân tích cú pháp (syntactical analysis) và cố gắng để chatbot có thể thông minh hơn với phương pháp phân tích ngữ nghĩa (semantic analysis). Hệ thống sử dụng phương pháp tiếp cận dựa trên thống kê (statistical based approach) để phân tích cú pháp, ngữ nghĩa câu hỏi và tạo phản hồi bằng cả hai phương pháp chiết xuất từ dữ liệu nguồn (extracted answer) và tạo ra phản hồi tự động (generated answer) tùy thuộc vào ý định trong câu hỏi của người dùng mà hệ thống phân tích được.

Từ những khái niệm đã nêu ra ở chương này, sang chương tiếp theo chúng tôi sẽ phân tích rõ hơn về đề tài cũng như giải pháp để thực hiện đề tài với những phương pháp xây dựng QAS đã lựa chọn.

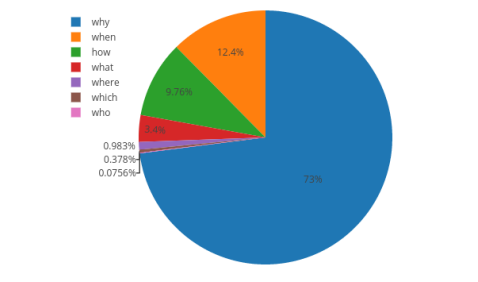
# CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ GIẢI PHÁP CHO ỨNG DỤNG CHATBOT CỦA ĐỀ TÀI

*Mục tiêu của chương này là phân tích kĩ hơn về đề tài, từ đó đưa ra giải pháp cho ứng dụng chatbot hỗ trợ hỏi đáp trong lĩnh vực kĩ thuật tập trung vào câu hỏi what và how. Chương này sẽ đi qua các bốn phần chính, đầu tiên là phân tích yêu cầu của đề tài, sau đó sẽ nói tổng quan về ứng dụng, hai phần cuối sẽ nói rõ hơn về chi tiết giải pháp sử dụng cho đề tài gồm cơ sở tri thức ứng dụng web ngữ nghĩa và xây dựng tin nhắn phản hồi dùng RASA.*

## 3.1 Phân tích yêu cầu đề tài

sản phẩm phần mềm với MS Word, MS Excel hay với Adobe Photoshop, Adobe Ilustrate Adobe Export PDF

Ứng dụng minh họa sẽ sử dụng Adobe Photoshop làm dữ liệu nguồn. Theo [10], chúng tôi có thống kê dữ liệu được thu thập từ các trang web và diễn đàn nhằm phân loại câu hỏi dựa trên lần xuất hiện đầu tiên của từ để hỏi (what, how, when, why…) trong câu hỏi như sau:



Hình 3. 1 Biểu đồ tròn thống kê các phần trăm theo loại câu hỏi về Adobe Photoshop dựa trên lần xuất hiện đầu tiên của từ để hỏi trong câu hỏi (tổng 1,322 câu hỏi) [10]

Từ biểu đồ trên có thể thấy được số lượng câu hỏi chiếm tỉ lệ nhiều nhất là why, tiếp đó là when, how và what. Câu hỏi when, theo như phân tích trong bài báo [11], thì câu hỏi when chiếm phần trăm lớn nhưng không có nghĩa là người dùng muốn câu trả lời cho câu hỏi “khi nào” mà là trong quá trình đặt câu hỏi, người dùng sẽ dùng từ when để mô tả trạng thái hiện tại của mình. Ví dụ *“When I try to resize image, I have an issue* …”, vì thế câu hỏi when thường là biến thể của câu hỏi why hoặc một vài câu hỏi khác nên chúng tôi sẽ bỏ qua trường hợp của câu hỏi này. Câu hỏi why chiếm phần trăm lớn nhất, bởi vì người dùng thường không dễ để tìm ra câu trả lời cho câu hỏi why (loại câu hỏi nguyên nhân – kết quả hay causal question), thường là lỗi xuất hiện khi người dùng đang thao tác với phần mềm vì vậy người ta sẽ thường có xu hướng đăng câu hỏi lên các diễn đàn để hỏi. Khác với dạng câu hỏi why, với những câu hỏi đơn giản (factoid question), khi có thắc mắc người dùng sẽ đọc tài liệu để tra cứu thông tin thay vì đăng bài hỏi. Câu hỏi why như đã nói ở chương 2, là dạng câu hỏi rất khó, hệ thống cần phải thực sự hiểu về dữ liệu, hiểu câu hỏi để đưa ra câu trả lời thích hợp. Ví dụ với câu hỏi “*Why has my photoshop commands change randomly?”* để trả lời được câu hỏi này hệ thống trước tiên cần phải hiểu *“What is photoshop commands?”, “How to change photoshop commands?”…* Trong biểu đồ trên, lượng câu hỏi what, how có thể không nhiều, nhưng dạng câu hỏi này là yếu tố cần thiết phải xây dựng được trước khi giải quyết câu hỏi why. Hơn nữa, nếu có một chatbot trả lời câu hỏi what và how, chắc chắn sẽ tiện hơn rất nhiều so với việc người dùng phải tra cứu tài liệu chuyên môn mỗi khi có thắc mắc.

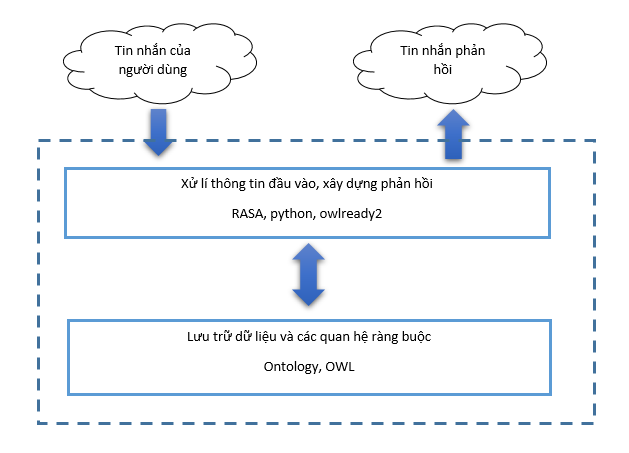
Hiện nay với dữ liệu được thu thập và phân tích trên trang Adobe Photoshop User Guide [12] (hướng dẫn sử dụng Adobe Photoshop). Toàn bộ dữ liệu được phân tích gồm tổng cộng 1891 câu hỏi, trong đó có 557 câu hỏi what (câu hỏi về định nghĩa) và 1334 câu hỏi how (câu hỏi về cách thức hoạt động).

Để xây dựng một ontology tập trung vào câu hỏi what, how,

* cỏwparaghraph style, camera raw, reshape or transform…

## 3.2 Tổng quan về ứng dụng

Sơ đồ bên dưới là thể hiện tổng quát về hệ thống chatbot: nhận tin nhắn của người dùng, xử lý và đưa ra phản hồi.



Hình 3. Mô hình hệ thống của ứng dụng

Với mô hình trên, chúng tôi tạo một website đơn giản bằng vueJS nhằm giao tiếp với người dùng (gửi nhận tin nhắn) – đây cũng chính là phần front-end của hệ thống. Chúng tôi xử dụng Restful API web service để kết nối giữa backend và frontend.

Về phần xử lí thông tin đầu vào và xây dựng phản hồi, chúng tôi dùng RaSa là một phần mềm mã nguồn mở nhằm giúp chúng tôi xây dựng chatbot, áp dụng các thuật toán xử lí ngôn ngữ tự nhiên để phân tích câu hỏi của người dùng và xây dựng các đoạn hội thoại. Đây là phần mềm được viết bằng python vì thế chúng tôi cũng dùng python để viết các hàm xử lí cần thiết.

Về phần lưu trữ dữ liệu và các quan hệ ràng buộc, chúng tôi sử dụng ontology được viết bằng ngôn ngữ owl. Sau đó tiến hành khởi chạy database này như một server riêng biệt để có thể lấy cũng như cập nhật dữ liệu một cách thuận tiện. Cũng nhờ đó mà phần back-end có thể kết nối để lấy dữ liệu một cách dễ dàng.

Ở hai phần tiếp theo chúng tôi sẽ nói rõ hơn về hai thành phần chính trong mô hình trên. Cụ thể ở phần 3.3 chúng tôi sẽ trình bày về web ngữ nghĩa và cách mà chúng tôi xây dựng cơ sở tri thức dựa vào ontology, ở phần 3.4 chúng tôi sẽ nêu tổng quan về phần mềm RASA cũng như cách chúng tôi áp dụng vào đề tài

## 3.3 Xây dựng cơ sở tri thức dựa vào Ontology

Theo [13], việc xây dựng ontolgy cho web ngữ nghĩa nói chung và ứng dụng chatbot nói riêng cũng sẽ đem lại một số lợi ích sau:

* **Tính chia sẻ:** giúp chia sẻ hiểu biết chung về cấu trúc thông tin giữa mọi người và các chương trình phần mềm.
* **Tái sử dụng:** Cho phép sử dụng lại kiến thức của lĩnh vực được trình bày.
* **Làm rõ nghĩa:** Mô tả chi tiết hơn về các đối tượng và quan hệ của chúng.
* **Phân loại:** Các mô hình dành cho hoạt động và định nghĩa.
* **Phân tích:** Đánh giá tính đúng đắn trong việc sử dụng các thuật ngữ.

Vì vậy, phần này sẽ nói về web ngữ nghĩa và cách mà chúng tôi lưu trữ dữ liệu và các quan hệ ràng buộc với ontology.

### 3.3.1 Lý thuyết về web ngữ nghĩa

Theo Wikipedia [14], thì “*Semantic Web (tiếng Việt: Web ngữ nghĩa) là một phong trào hợp tác được dẫn đầu bởi tổ chức W3C. Tiêu chuẩn này phát triển các định dạng dữ liệu chung trên World Wide Web. Bằng cách khích lệ sự bao hàm của nội dung ngữ nghĩa trong các trang web, Semantic Web hướng đến việc chuyển đổi các nội dung web hiện tại bao gồm phần lớn các văn bản không có cấu trúc hoặc bán cấu trúc thành "dữ liệu web" (dữ liệu có ngữ nghĩa). Ngăn xếp Semantic Web xây dựng trên RDF thuộc W3C. Theo W3C, "Semantic Web cung cấp một framework chung cho phép dữ liệu được chia sẻ và tái sử dụng thông qua các ứng dụng, các công ty và các biên giới cộng đồng". Thuật ngữ này được đặt ra bởi Tim Berners-Lee cho dữ liệu web có thể thực thi bởi máy tính.”* Có thể xem đây là một tiêu chuẩn tạo nên cơ sở tri thức cho web, giúp máy tính có thể hiểu được thông tin trên web và các lập trình viên, các nhà nghiên cứu, các doanh nghiệp có thể thuận tiện trong việc tách, chiết, sử dụng, mở rộng các tri thức này cho nhiều mục đích khác nhau. Vậy tóm lại Web ngữ nghĩa chính xác như cái tên của nó, là Web nhưng được tổ chức thông tin một cách có ngữ nghĩa.

Để rõ hơn về công dụng của web ngữ nghĩa, chúng tôi xin trích dẫn một đoạn trong [15], “*Các công cụ Search với cơ chế hoạt động dùng web spider lùng sục những trang web mới, phân tích dữ liệu trên trang web để “cố gắng” hiểu nội dung của nó, sau đó đánh chỉ số và lưu trữ các thông tin quan trọng và địa chỉ của trang web. Khi người search thông tin, họ sẽ đưa ra những từ khóa, nếu từ khóa khớp với những từ quan trọng của trang web nào đó thì trang đó sẽ là kết quả trả về của phép tìm kiếm. Nhưng cơ chế này hạn chế ở ngữ nghĩa. Vì nếu chỉ sử dụng những từ khóa để tìm kiếm thì do vấn đề ngôn ngữ (từ đa nghĩa…) mà có nhiều sự nhập nhằng và nhầm lẫn dẫn đến kết quả sai. Hơn nữa, việc hiểu nội dung của trang web thông qua phân tích text cũng không phải là phương pháp có độ chính xác cao. Vì vậy, mà cộng đồng những nhà nghiên cứu vẫn nung nấu một mong muốn “ngữ nghĩa” hóa hệ thống web. Nghĩa là trang web bây giờ sẽ chứa đựng thông tin chủ đạo trên trang đó và biểu diễn dưới dạng mà máy tính có thể hiểu được, xử lý được. Và nếu viễn cảnh đó thành công thì triết lý “thông tin mọi nơi” sẽ biến thành “thông tin tốt nhất mọi nơi”.”*

Đối với Word wide web, cấu trúc trang web được biểu diễn dưới dạng HTML, do đó chúng ta chỉ có thể hiểu được một phần nghĩa của trang web như là tittle, content mà hầu như không thể biết được trang web đó đang mô tả cái gì, lĩnh vực nào, có liên hệ ra sao với những trang web khác. Web ngữ nghĩa thì hơn thế, với sự hỗ trợ của ngôn ngữ RDF và các mở rộng của nó, chúng ta có thể thêm vào các dữ liệu trang web những mối quan hệ về mặt ngữ nghĩa như hasAuthor, isChildOf, hasPrice, … Tổ chức W3C có một nhánh nghiên cứu để tạo ra các cấu trúc ngữ nghĩa chung (ontology) cho từng lĩnh vực để chúng ta có thể áp dụng, tạo ra một thế hệ web ngữ nghĩa mới. Ngoài ra, mỗi tổ chức cá nhân đều có thể tạo ra một bản thể riêng (ontology) cho dữ liệu của mình.

### 3.3.2 Ontology

#### 3.3.2.1 Khái niệm

Theo [15] : “*trong khoa học máy tính, một ontology là một mô hình dữ liệu biểu diễn một lĩnh vực và được sử dụng để suy luận về các đối tượng trong lĩnh vực đó và mối quan hệ giữa chúng. Ontology cung cấp một bộ từ vựng chung bao gồm các khái niệm, các thuộc tính quan trọng và các định nghĩa về các khái niệm và các thuộc tính này. Ngoài bộ từ vựng, ontology còn cung cấp các ràng buộc, đôi khi các ràng buộc này được coi như các giả định cơ sở về ý nghĩa mong muốn của bộ từ vựng, nó được sử dụng trong một miền mà có thể được giao tiếp giữa người và các hệ thống ứng dụng phân tán hỗn tạp khác.”*

Cụm từ “ontology” được lấy từ triết học, nghĩa là “theory of existence”, “bản thể luận” hay “lý thuyết về sự tồn tại”. Có thể hiểu ontology là khái niệm hóa sự vật, sự việc. Với ontoloty ta có thể biết được tập hợp những cấu trúc cơ bản nhất của một sự vật sự việc và quan hệ giữa chúng, ví dụ như “xe ô tô” thì “gồm” “đinh vít”, “bánh xe”, “đèn xe”, “ghế”, “tay lái”, …., trong “đèn xe” lại “có” “vỏ đèn”, “tim đèn”, “màu sáng”, … vỏ đèn thì “làm bằng” “nhựa” hay “thủy tinh”.

Đối với web ngữ nghĩa, ontology chính là đại diện cho cơ sở tri thức. Một trang web được gọi là có ngữ nghĩa khi nó có ontology và được cấu trúc theo ontology của nó. Mỗi ontology mô tả một miền tri thức, nó có khả năng mở rộng, kết hợp với các ontology khác tạo nên một mạng lưới tri thức rộng lớn, đó chính là tương lai của Web sau này. Khi mà tất cả trang web đều có thể liên kết với nhau, hiểu được nghĩa của nhau và có khả năng kế thừa tri thức của nhau.

Với những phân tích bên trên, nhóm tác giả chọn ontology nhằm tạo ra một cơ sở tri thức mới cho hệ thống hỏi đáp QAS trên web ngữ nghĩa. Chúng tôi hy vọng những đóng góp của mình sẽ thúc đẩy việc phát triển của hệ thống QA – question dựa trên web ngữ nghĩa và những hệ thống hỏi đáp sau có thể kế thừa cơ sở tri thức của hệ thống trước để tạo nên những chatbot ngày càng thông minh mà không làm mất quá nhiều thời gian để xây dựng cơ sở tri thức (KBS) mới.

#### 3.3.2.2 Cấu trúc

Có nhiều cách chia thành phần Ontology, ở luận văn này, tôi xin trình bày thành phần Ontology mà cách chia của nó hướng đến đối tượng người lập trình và các phần mềm hỗ trợ soạn thảo ontology hướng đến. Bao gồm 3 thành phần:

##### 3.3.2.2.1 Class – Lớp

Là thành phần quan trọng nhất của một Ontology, nó thể hiện cấu trúc, đặc điểm của một ontology. Một lớp là tập hợp các Individual khác nhau nhưng có cùng một hoặc một vài đặc tính chung. Ví dụ: Lớp xe là tập hợp của tất cả các loại phương tiện giao thông phân biệt nhau bởi biển số xe, nhưng nó cùng có đặc tính là có bánh xe và là phương tiện để chở người, vật lưu thông trên đường.

Mỗi class gồm một hoặc nhiều object thể hiện thông tin về các đặc tính của Class đó.

Một class có thể có các SubClass là một class con của class hiện tại. Giống với khái niệm kế thừa trong hướng đối tượng, một SubClass sẽ bao gồm tất cả các đặc tính mà Class cha của nó có, kèm theo một vài đặc tính riêng biệt mà chỉ có các thể hiện (Individual) thuộc Subclass mới có.

##### 3.3.2.2.2 Individual – Thể hiện

Là một đơn vị dữ liệu trong ontology. Individual dựa trên cơ sở các Class, thể hiện thông tin của một cá thể riêng biệt. Cũng với ví dụ về lớp xe bên trên, thì Individual ở đây chính là một phương tiện giao thông với 1 biển số xe riêng biệt. Một phương tiện giao thông đó là một cá thể trong class xe và tất cả đặc điểm của loại xe này sẽ được nêu rõ trong Object.

##### 3.3.2.2.3 Các thuộc tính (Properties)

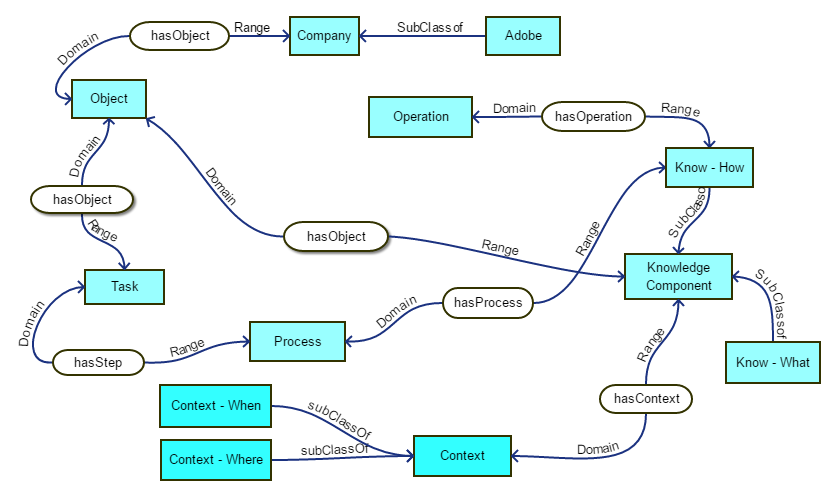
Các đối tượng trong ontology có thể được mô tả thông qua việc khai báo các thuộc tính của chúng. Mỗi một thuộc tính đều có tên và giá trị của thuộc tính đó. Các thuộc tính được sử dụng để lưu trữ các thông tin mà đối tượng có thể có. Ví dụ, đối với một cá nhân có thể có các thuộc tính: Họ\_tên, ngày\_sinh, quê\_quán, số\_cmnd…

Giá trị của một thuộc tính có thể có các kiểu dữ liệu phức tạp.

##### 3.3.2.2.4 Relation – Quan hệ

Relation biểu diễn mối quan hệ giữa các Individual với nhau, giữa các Class với nhau, đôi khi là mối quan hệ giữa các Relation với nhau. Mỗi quan hệ sẽ có “range” là miền giá trị của quan hệ, “domain” là đối tượng mà quan hệ liên kết tới.

### 3.3.3 Mô hình đề xuất cho hệ thống hỏi đáp tập trung vào câu hỏi “what” và “how”

 Như đã trình bày ở phần 3.1, chúng ta cần một ontology có các thành phần company, context, know - what, know – how, object…, dưới đây là sơ đồ ontology cho hệ thống hỏi đáp tập trung vào câu hỏi *what* và câu hỏi *how*, với các ô màu xanh đại diện cho *class*, ô màu trắng đại diện cho mối quan hệ giữa các *class*:

Hình 3. Mô hình ontology đề xuất cho đề tài

Sau đây chúng tôi sẽ trình bày rõ hơn về các class cũng như các mối quan hệ giữa chúng:

- **Company** là class thể hiện cho công ty chủ sở hữu các phần mềm kĩ thuật – vì ở đây chúng tôi dùng AdobePhotoshop làm minh họa nên “Company” sẽ có subclass là “Adobe”, AdobePhotoshop cũng như các sản phẩm khác của Adobe sẽ làm invidual của class “Adobe”. Trong trường hợp có các công ty khác cần thêm vào KBS như Microsoft chằng hạn, lúc đó class “Company” sẽ có thêm một class con có mối quan hệ subClassOf với “Company” là class “Microsoft”, tương tự các sản phẩm của Microsoft như Excel, Word sẽ là các invidual của class Microsoft. Company là class cha của các class như Adobe hay Microsoft, mà mỗi sản phẩm kĩ thuật của mỗi công ty sẽ có những object riêng không giống nhau và nó chỉ liên quan đến object đó chứ không phải bất kì object nào thuộc sản phẩm kĩ thuật khác. Vậy nên chúng tôi có mối quan hệ “hasObject” giữa company đến class “Object”. Điều này cũng có nghĩa là mỗi invidual trong class con (hay nói cách khác là mỗi sản phẩm kĩ thuật) sẽ có các mối quan hệ “hasObject” đến class “Object”.

- **Context** là ngữ cảnh trong câu hỏi của người dùng, có thể là phiên bản sử dụng, môi trường, tác nhân… Ở đây, có 2 loại Context là Context-When – ngữ cảnh liên quan đến thời gian, phiên bản (CS6, CS5) … và Context-Where – ngữ cảnh liên quan đến vị trí, môi trường cài đặt… (Window, Mac OS)

- **Object** là đối tượng mà người dùng muốn hỏi. Giả sử đối với câu hỏi what – là câu hỏi về định nghĩa. Khi người dùng hỏi “*what is layer*?” -> Object ở đây là “*Layer*”.

- **Knowledge Component** là miền kiến thức – nơi lưu trữ dữ liệu chính để phản hồi lại cho người dùng. Mỗi Knowledge Component có thể có một hoặc nhiều object và context (bắt buộc phải có ít nhất một object), vì vậy nó sẽ cần có mối quan hệ hasObject đến class “Object” và mối quan hệ “hasContext” đến class “Context”. Hiện tại “Knowledge Component” có hai SubClass là “Know – What” và “Know – How”.

+ **Know – What** chứa miền kiến thức liên quan đến câu hỏi what. Mỗi invidual được xem như một câu hỏi what.

+ **Know – How** chứa miền kiến thức liên quan đến câu hỏi How. Mỗi invidual được xem như một câu hỏi how. Như đã phân tích ở phần trước, câu hỏi How – Làm cái đó như thế nào – sẽ có thêm một đặc điểm là hành động, phương thức (Operation). VD: *How I can change Background in PS CS6*? Thì *change* là một operation, *Background* là object và *CS6* là context. Vì vậy, ngoài 2 mối quan hệ “hasObject” và “hasContext” kế thừa từ class cha “Knowledge Component” thì “Know - How” sẽ có thêm mối quan hệ “hasOperation” đến class “Operation” như đã thể hiện trong sơ đồ trên.

- **Task** là một nhiệm vụ hay một bước thực hiện nhiệm vụ. Task có thể không liên hệ đến một object nào hoặc cũng có thể sẽ phải tác động đến một hay nhiều object nào đó trong khi thực hiện. Vì vậy nó sẽ có mối quan hệ hasObject đến class object. Tác giả muốn thêm mối quan hệ này vào nhằm mục đích tạo ra một mạng lưới ngữ nghĩa rộng hơn để khi mà hệ thống sau này có muốn truy vết lại rằng người dùng đã từng tác động đến object nào dựa vào cả câu trả lời thay vì chỉ có thể biết được thông qua phân tích câu hỏi của người dùng. Để dễ dàng lấy cái task ra theo đúng thứ tự, chúng tôi có một qui ước đặt tên cho cái task là cuối mỗi task phải có cụm “\_step” + số thứ tự của step (stt). Ví dụ tôi có 3 task thực hiện process resize\_image thì tên các task sẽ lần lượt là resize\_image\_step1, resize\_image\_step2, resize\_image\_step3, lưu ý rằng nếu bạn không muốn đặt tên các task theo tên process thì vẫn có thể đặt tên khác nhưng phải có cụm “\_step” + stt để phân biệt thứ tự của mỗi bước.

- **Process** là một quy trình hay nói cách khác là một cách để thực hiên câu hỏi how. Nó bao gồm một tập các bước (Task) để thực hiện quy trình đó, đây cũng chính là lý do chúng ta cần mối quan hệ “hasStep” để chỉ rõ Process sẽ bao gồm những Task nào. Một câu hỏi How sẽ có thể có một hoặc nhiều Process để có thể thực hiện yêu cầu. Mỗi Process sẽ có nhiều Task để thực hiện thông qua mối quan hệ “hasStep”.

- **Operation** là phương thức, hành động mà người dùng muốn tác động đến Object. Được dùng cho câu hỏi how.

Ví dụ sử dụng Ontology trên cho các câu hỏi:

What is composite?

-> Object: Composite

What is Recording Tool in CS6?

-> Object: Recording Tool

-> Context - When: CS6, (version)

How to add text into image?

-> Object: image, text

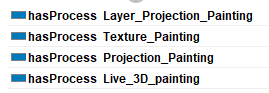
-> Operation: add

How painting 3D?

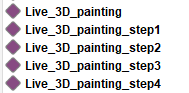
-> Object: 3D

-> Operation: Paint

-> Gồm có 4 process tương ứng với 4 cách vẽ 3D:



-> Mỗi process sẽ gồm nhiều Task hoặc không có Task, ví dụ đối với Process Layer\_Projection\_Painting không có Task, nhưng với Live\_3D\_painting thì có 4 Task, các Task được đặt tên bằng: <tên process>\_step<số thứ tự của Task>, cách đặt tên này giúp chatbot dễ dàng lấy từng bước ra show lên cho user



How to install photoshop CS6 in Mac OS?

-> Object: photoshop

-> Operation: install

-> Context: CS6 (When), Mac OS (Where)

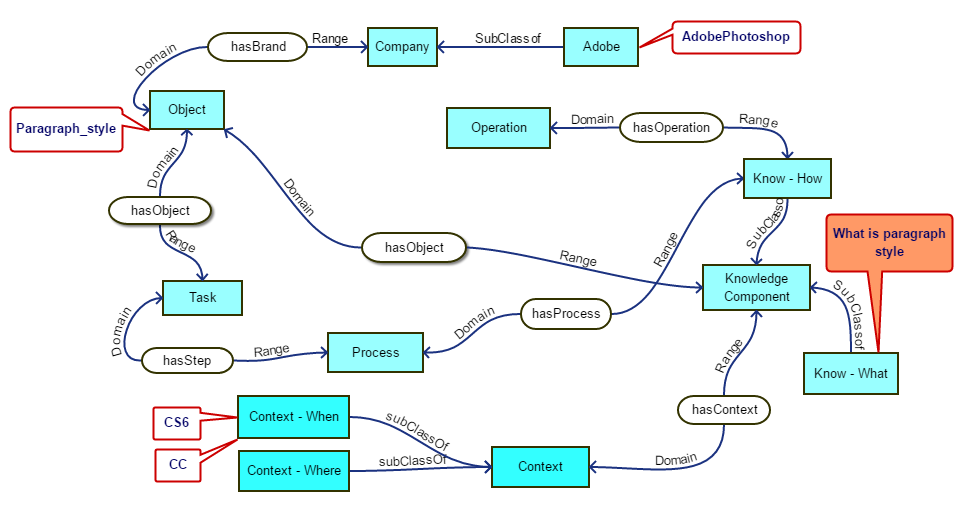
How to dock or undock panel?

-> Object: panel

-> Operation: dock, undock

-> Đây là 1 câu hỏi how đồng thời cũng là một process của thể hiện Manage\_Windows\_and\_Panels và nó có 3 Task.

Dưới đây là ảnh minh họa khi áp các câu hỏi vào mô hình:



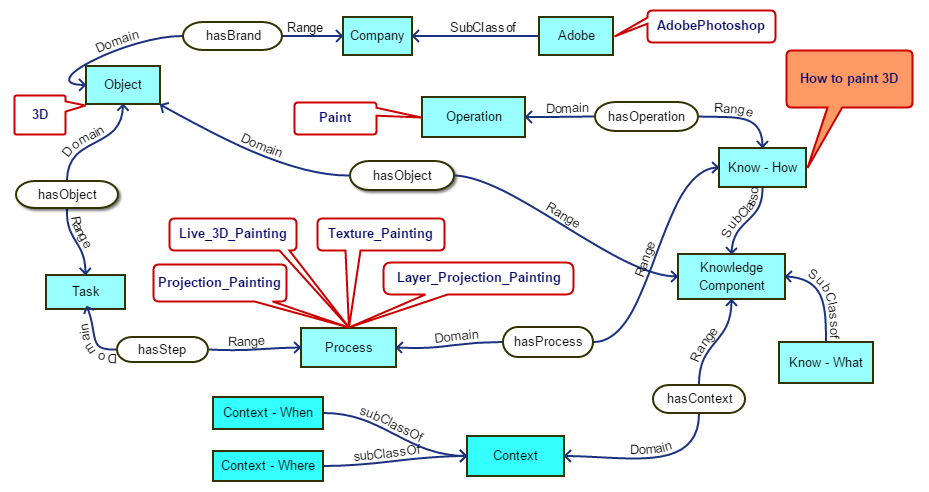
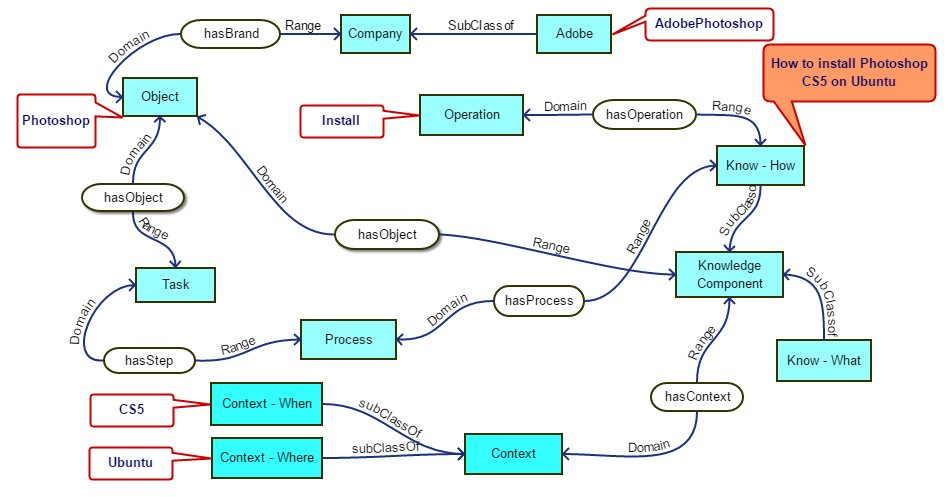
Hình 3. Ví dụ sử dụng ontology đề xuất với câu hỏi “What is paragraph style?”

Hình 3.4 là minh họa cho việc phân tích câu hỏi what is paragraph style và đưa vào ontology. Với câu hỏi này chúng tôi sẽ thêm các thể hiện sau:

* What\_is\_paragraph\_style là một thể hiện của class Know - What.
* CS6 và CC là lần lượt là các thể hiện của class Context – When.
* Paragraph\_style là một thể hiện của class Object.
* AdobePhotoshop là một thể hiện của class Adobe.

Giờ chúng tôi sẽ nói về mối liên hệ của các thể hiện trên. Thể hiện What\_is\_paragraph\_style sẽ có mối quan hệ hasObject đến thể hiện Paragraph\_style và mối quan hệ hasContext đến hai thể hiện CS6 và CC. Thể hiện Paragraph\_style sẽ có mối quan hệ hasBrand đến thể hiện AdobePhotoshop.

Tương tự chúng ta có hai sơ đồ bên dưới là ví dụ cho việc sử dụng ontology với câu hỏi “*How to install Photoshop CS5 on Ubuntu?*” và câu hỏi “*How to paint 3D?*”

Hình 3. Ví dụ sử dụng ontology đề xuất với câu hỏi “How to install Photoshop CS5 on Ubuntu?”

Hình 3. Ví dụ sử dụng ontology đề xuất với câu hỏi “How to paint 3d?”

### 3.3.4 Khả năng mở rộng

Ontology này có thể mở rộng theo thời gian để trả lời thêm cho các câu hỏi phức tạp hơn như: *What is different between background image and background color* chẳng hạn. Đối với những câu hỏi như vậy chúng tôi sẽ mở rộng bằng cách thêm SubClass cho loại câu hỏi vào Intent. Ví dụ với câu hỏi what so sánh như trên sẽ thêm SubClass mới là “Know - whatCompare” – class cho loại câu hỏi what mang nghĩa so sánh. Mô hình cũng có thể mở rộng cho các loại câu hỏi khác ngoài factoid-question như causal-question (Know - Why).

### 3.3.5 Những lợi ích trong mô hình đã đề xuất

- Có thể truy xuất đến các câu hỏi khác liên quan nhằm gợi ý cho người hỏi trong quá trình sử dụng chatbot nếu người dùng hỏi những câu bị thiếu dữ kiện. Ví dụ nếu người dùng hỏi *what is blend mode?* Thì chúng tôi sẽ gợi ý các câu liên quan cho người dùng chọn, đó là *what is blend mode “Hue”? What is blend mode color?* …

- Một câu hỏi có thể bao gồm nhiều context, operation, object.

- Lưu câu trả lời theo step giúp uyển chuyển trong việc tạo response trả lời cũng như khả năng mở rộng tính năng trả lời của chatbot sau này. Ví dụ cho việc mở rộng tính năng như đã nói ở phần trước là chatbot có thể nhận biết người dùng đang ở bước nào của qui trình và gợi ý người dùng thực hiện các bước tiếp theo hay là chỉ đưa ra step tiếp theo sau khi nhận được phản hồi của người dùng…

- Khi người dùng đưa ra một câu hỏi không có trong hệ thống ontology mà chỉ có vài thành phần. Ví dụ: “*How to change color?”* Trong khi hệ thống chỉ có các câu trả lời cho “*How to change background color?”* và “*how to change image color?”* Thì với ontology ta có thể dễ dàng truy xuất ra những dữ liệu có chứa action change color và đưa ra gợi ý cho người dùng.

- Việc phân chia intent thành các loại câu hỏi là cần thiết vì có rất nhiều loại câu hỏi, mà mỗi loại sẽ có các đặc tính khác nhau, việc xử lý câu hỏi dù dùng thuật toán nào thì cũng cần nhận diện được loại câu hỏi trước khi trích xuất đặc trưng câu hỏi. Vậy nên việc lưu theo từng loại câu hỏi là cơ sở để việc truy vấn trở nên chính xác hơn, có thể chỉnh sửa câu trả lời và dự đoán ý định của người dùng một cách thuận tiện hơn. Ngoài ra cũng giúp dễ dàng hơn trong việc nghiên cứu áp dụng lại các hệ thống hỏi đáp được xây dựng đưa theo loại câu hỏi (xem lại chương 2 – phân loại QAS). Hiện tại đã có rất nhiều những công trình nghiên cứu về QAS phân loại theo câu hỏi, các công trình này có thể sử dụng restricted domain hoặc general domain. Nếu kết hợp các thuật toán truy xuất dữ liệu từ general domain trong việc xây dựng một ontology chuẩn sẽ giúp tiết kiệm thời gian trong quá trình xây dựng ontology, cũng như cải thiện hiệu suất của hệ thống rất nhiều.

- Các Task có liên quan đến Object giúp chatbot có thể hiểu được câu trả lời mà nó đề xuất cho người dùng, từ đó giữ lại được nhiều thông tin hơn để phản hồi người dùng trong những câu hỏi tiếp theo, vì nó có thể biết được người dùng đang làm gì, đang tác động đến những đối tượng nào.

## 3.4 Xử lí tin nhắn đầu vào và xây dựng phản hồi với RASA

Phần này sẽ nói về phần xử lí tin nhắn đầu vào và xây dựng phản hồi, thuộc phần còn lại trong sơ đồ ở hình 3.2

### 3.4.1 Giới thiệu chung về Rasa

Hiện nay, có rất nhiều framework cũng như công cụ để xây dựng chatbot như Dialogflow của Google, Bot Framework của Microsoft. Cả 2 framework này đều sử dụng các tính năng hiểu ngôn ngữ tùy chỉnh được huấn luyện sẵn, do đó chúng được xem như một công cụ giúp bạn xây dựng chatbot dễ dàng hơn và nhanh hơn. Kể cả khi chúng ta không có nhật ký trò chuyện nào, chúng ta vẫn có thể sử dụng dữ liệu huấn luyện của mình.

Đôi khi, chúng ta sẽ cảm thấy bị phụ thuộc khi cứ phải sử dụng mã nguồn đóng để xử lý tin nhắn đầu vào của người dùng trên các máy chủ của Google hoặc Microsoft. Chẳng hạn, chúng ta muốn phát triển một chatbot cho doanh nghiệp cá nhân, chatbot sẽ tiếp nhận các thông tin nhạy cảm của người dùng một cách bảo mật. Trường hợp như vậy, chúng ta sẽ cảm thấy thoải mái hơn khi chatbot hoàn toàn được giữ tại doanh nghiệp của mình. Đây là lý do tại sao Rasa được ra đời và mang đến rất nhiều thuận lợi. Rasa là một mã nguồn mở, nó không có bất kì mô hình dựng sẵn nào trên máy chủ mà bạn có thể gọi bằng API, chúng ta phải tự xây dựng để nó trở nên hoàn thiện hơn và tất nhiên sẽ mất nhiều thời gian hơn cho việc xây dựng nó. Tuy nhiên, để có thể kiểm soát hoàn toàn dữ liệu của mình thì đây là một việc hoàn toàn thích đáng để bỏ thời gian, công sức đầu tư.

|  |  |
| --- | --- |
| **DialogFlow** | **RASA** |
| Là công cụ hoàn chỉnh để xây dựng chatbot của google. | Để đạt được như Dialogflow cần xây dựng cả Rasa NLU và Core. Rasa NLU xử lý ý định/thực thể. Trong khi Rasa Core xử lý đối thoại và thực hiện. |
| Có thể phân loại thực thể, ý định. Nó sử dụng những gì gọi là bối cảnh để xử lý đối thoại. Cho phép móc nối các website thực hiện. | Không cung cấp một giao diện người dùng nào cả, tương tác chủ yếu là Json hoặc markdown. Rasa Core yêu cầu phát triển trên python để có thể tùy chỉnh bot. |
| Không có hình thức quản lý người dùng cuối. | Không trực triếp hình thức quản lý người dùng nào. |
| Dữ liệu được lưu trữ trên đám mây và mọi tương tác với API.ai đều yêu cầu liên lạc liên quan đến đám mây. | Không cung cấp dích vụ lưu trữ. Do đó dữ liệu tự lưu trữ và độc quyền dữ liệu. |
| Không thể host tại server của mình. | Có thể host tại server. |
| Có tích hợp với Facebook messenger, Slack, Skype | Không tích hợp |
| Dễ dàng sử dụng, không yêu cầu có kĩ thuật. | Yêu cầu phải có kiến thức về kĩ thuật. |

Table So sánh 2 công cụ xây dựng chatbot là DialogFlow và RASA

Rasa được phát triển từ nhiều nguồn khác nhau, scikit-learn và keras đều là thành phần chính cấu tạo nên Rasa API. Phân loại văn bản (text classification) dựa trên hướng tiếp cận như fastText, các câu được gộp lại thành một word vector, sử dụng pre-train giống như thư viện GloVe. Hiệu suất của Rasa được so sánh với nhiều giải pháp khác nhau (mã nguồn đóng). Các thực thể (Entity) được nhận diện bằng nhiều trường điều kiện khác nhau.

### 3.4.2 Các thành phần chính của Rasa

Rasa là một mô hình kiến trúc tự thiết kế, do đó có thể dễ dàng tích hợp với hệ thống của chúng ta. Rasa core được xem như một hệ thống quản lý các cuộc hội thoại, hướng xây dựng Rasa core khác với một số hệ thống khác. Rasa NLU và Rasa core là 2 thành phần tách biệt và độc lập với nhau hoàn toàn, vì thế ta có thể train 2 model độc lập với nhau.

#### 3.4.2.1 Mô hình (Architecture)

- Trạng thái của cuộc nói chuyện được lưu trữ trong một đối tượng được gọi là “tracker”. Mỗi cuộc hội thoại đều có một tracker lưu trữ và nó là duy nhất. Tracker lưu trữ các slot (entity, object, action…) cũng như nhật kí tất cả lịch sử dẫn đến trạng trạng thái nào đó đã xảy ra trong cuộc trò chuyện. Trạng thái của cuộc trò chuyện được bắt đầu lại bằng việc trả lời hết tất cả các sự kiện.

- Mô hình hoạt động của Rasa. Khi nhận được một tin nhắn Rasa sẽ thực hiện lần lượt các bước:

+ Bước 1: khi nhận được một tin nhắn, tin nhắn sẽ được gửi đến “người thông dịch” (Interpreter – Rasa NLU) để rút trích các entity, intent… và các thông tin khác.

+ Bước 2: Tracker sẽ duy trì trạng thái của cuộc trò chuyện. Nó nhận được thông báo là có một tin nhắn mới vừa tới.

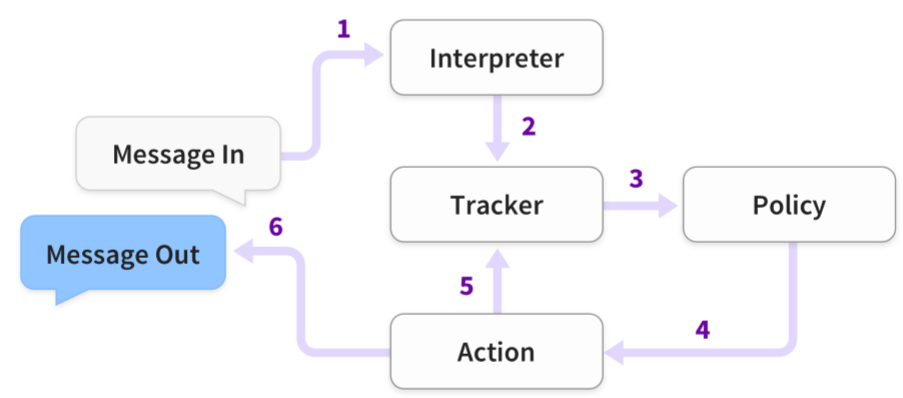
+ Bước 3: Policy nhận được trạng thái hiện tại của Tracker.

+ Bước 4: Policy sẽ quyết định xem action nào tiếp theo sẽ được thực hiện.

+ Bước 5: Action được chọn được ghi lại vào tracker.

+ Bước 6: Action sẽ được gọi và gửi respone về cho user.

+ Bước 7: Nếu predict action không được duyệt thì quay lại bước 3.



Hình 3. Mô hình hoạt động của RASA

#### 3.4.2.2 Hành động (Action)

- Ta sẽ xét vấn đề quản lý cuộc hội thoại như một bài toán phân loại vấn đề. Rasa Core sẽ giải quyết vấn đề này, khi tiếp nhận một ý định từ Rasa NLU, Rasa Core sẽ dự đoán xem action nào sẽ được đưa ra tiếp theo được lấy từ một danh sách hành động cho trước để trả lời cho người dùng. Khi một action được gọi, nó sẽ được lưu vào trong tracker, do đó ta có thể lấy kì thông tin nào trong lịch sử cuộc nói chuyện một cách dễ dàng.

- Action không thể trực tiếp thay đổi tracker, nhưng khi được gọi đến thì có thể trả về một danh sách sự kiện. Tracker sử dụng các sự kiện này để cập nhật lại trạng thái state của mình. Có rất nhiều loại event khác nhau như: SlotSet, AllSlotsReset, …

#### 3.2.2.3 Rasa NLU (Xử lý ngôn ngữ tự nhiên)

- Rasa NLU là một module xử lý ngôn ngữ tự nhiên được kết hợp bởi một số thư viện và máy học để trở thành 1 API hoàn chỉnh. Rasa NLU mang đậm tính tùy biến và dễ sử dụng. Rasa cung cấp một “pipeline” – được xem như là một khung lớn, ta thêm các thành phần xử lý ngôn ngữ tự nhiên vào để có thể hoạt động một cách hợp lý. Rasa NLU được hổ trợ các thư viện nổi tiếng như: spacy\_sklearn, tensorflow\_embedding, mite, …

#### 3.2.2.4 Chính sách (Policy)

- Nhiệm vụ của Policy là chọn hành động tiếp theo để đưa vào tracker. Policy được khởi tạo cùng các đặc trưng, nhằm tạo ra vector biểu diễn trạng thái hiện tại của cuộc đối thoại cung cấp cho tracker.

- Các đặc trưng bao gồm:

+ Hành động cuối cùng là gì?

+ Ý định (intent) và thực thể (entity) mà người dùng đề cập là gì?

+ Những slots nào được xác định?

- Các đặc trưng của slot rất nhiều, trường hợp đơn giản nhất là biểu diễn dưới dạng vector nhị phân. Trường hợp sử dụng slot cho nhiều biến sẽ được biểu diễn one-of-k binary vector.

- Siêu tham số max\_history để lưu lại số lượng các trạng thái trước đã thêm vào. Trạng thái được biểu diễn dưới dạng mảng 2 chiều, có thể được xử lý bằng RNN hoặc là model thông thường. Đối với hầu hết các vấn đề, giá trị của max\_history nhằm trong khoảng 3-6 là hoạt động tốt.

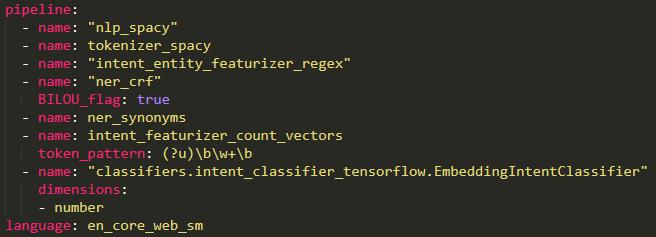
### 3.4.3 Rasa NLU

Bất kì “trợ lý ảo” nào cũng cần phải có 2 yếu tố: hiểu được ý định và đưa ra câu trả lời đúng cho ý định đó. Rasa giải quyết 2 yêu cầu trên với việc sử dụng RasaNLU cho việc hiểu ý định người dùng (natural language understanding) và quản lý cuộc đối thoại với thành phần Rasa Core. Rasa NLU cung cấp cấu hình tùy chỉnh thông qua “pipeline” để xử lý thông điệp người dùng như phân loại ý định, trích xuất thực thể…

#### 3.4.3.1 Pipeline

Pipeline là một khối xây dựng chính của Rasa NLU. Nó xác định các giai đoạn xử lý thông điệp người dùng. Các giai đoạn đó là: tách từ, trích xuất đặc trưng, phân loại ý định, khớp mẫu… Mặc định, Rasa NLU đi kèm một số các thành phần xây dựng sẵn để chúng ta có thể dễ dàng sử dụng. Dưới đây là một ví dụ về cấu hình pipeline:

Hình 3. Ví dụ về pipeline



- Khi đường ống được xác định, mỗi thành phần được gọi lần lượt và tạo ra đầu ra được thêm trực tiếp vào đầu ra mô hình Rasa NLU hoặc được sử dụng làm đầu vào cho các thành phần khác. Điều quan trọng là việc xác định các thành phần như thế nào. Ví dụ nếu bạn cấu hình pipeline của bạn với 3 thành phần: [‘Component1’, ‘Component2’, ‘Component3’] luồng hoạt động sẽ như sau:



Hình 3. Luồng hoạt động của pipeline với 3 thành phần

- Các thành phần đều trả qua 3 giai đoạn chỉnh:

* Khởi tạo (create): khởi tạo các thành phần trước khi huấn luyện
* Huấn luyện (train): thành phần huấn luyện sử dụng bối cảnh và có khả năng là đầu ra của thành phần trước đó.
* Kiên trì (persist): lưu thành phần đào tạo trên đĩa để sử dụng trong tương lai.

- Trước khi thành phần đầu tiên được khởi tạo, một ngữ cảnh được gọi sẽ tạo ra sự tương tác truyền thông tin giữa các thành phần. Ví dụ, một thành phần có thể tính toán các vectơ đặc trưng cho dữ liệu huấn luyện, lưu trữ trong bối cảnh và một thành phần khác có thể truy xuất các vectơ đặc trưng này từ ngữ cảnh và thực hiện phân loại ý định. Khi tất cả các thành phần được tạo, đào tạo và duy trì, siêu dữ liệu mô hình được tạo để mô tả mô hình NLU tổng thể.

- Đặc biệt RasaNLU còn cho phép bạn cấu hình thêm các thành phần bạn tự xây dựng hoặc đã huấn luyện từ trước.

#### 3.4.3.2 Phân loại ý định và rút trích thực thể

##### 3.4.3.2.1 Phân loại ý định

- Rasa sử dụng thuật ngữ ý định (intent) để mô tả phân loại tin nhắn của người dùng. Rasa NLU sẽ phân loại các thông điệp của người dùng thành 1 hay nhiều ý định Có 2 cách sau đây chúng ta có thể lựa chọn:

+ Pretrained Embeddings (Intent\_classifier\_sklearn)

+ Supervised Embeddings (Intent\_classifier\_tensorflow\_embedding)

###### 3.4.3.2.1.1 Pretrained Embeddings (Intent\_classifier\_sklearn)

Classifier này sử dụng thư viện Spacy để tải mô hình ngôn ngữ đã được chọn lọc từ trước, sau đó biểu diễn từng từ trong thông điệp của người dùng dưới dạng “word embedding” - là một vector biểu diễn các từ, tức là mỗi từ sẽ được chuyển thành một vector số. Muốn biết rõ hơn thì có thể tìm hiểu về word2vec [16].

Word embedding là các ngôn ngữ cụ thể đã được train sẵn từ trước. Do đó ta phải chọn một trong những model đó tùy thuộc vào ngôn ngữ mà bạn đang xử lý. Ngoài ra có thể sử dụng các word embedding khác như fastText embeddings của Facebook. Để chuyển đổi các nhúng vào mô hình spaCy tương thích và sau đó liên kết mô hình đã chuyển đổi với ngôn ngữ bạn chọn (ví dụ: en) với liên kết

- Python –m spacy link <converted model> <langague code>.

Rasa NLU lấy trung bình tất cả các “word embedding” trong thông điệp người dùng, sau đó chuyển thành dạng girdsearch [17] để tìm tham số tốt nhất cho support vector classifer [18] – SVC sẽ phân loại các embedding trung bình thành các loại ý định khác nhau. Các girdsearch huấn luyện nhiều loại SVC với các tham số cấu hình khác nhau, sau đó chọn ra cấu hình tốt nhất dựa trên kết quả kiểm tra.

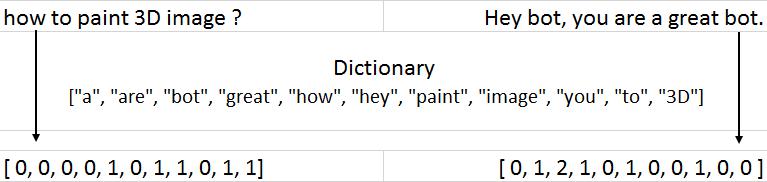
• Tại sao nên sử dụng thành phần này

- Sử dụng các biểu diễn từ (word embedding) đã được huấn luyện trước được sẽ được thừa hưởng những tiến bộ của việc nghiên cứu trong việc huấn luyện các biểu diễn từ mạnh hơn và ý nghĩa hơn. Khi mà các biểu diễn từ này được huấn luyện sẵn, SVM chỉ cần yêu cầu một ít huấn luyện để có thể đưa ra độ chính xác của việc dự đoán các ý định. Điều này làm cho trình phân loại (classifer) được hoàn toàn phù hợp khi bắt đầu các dự án trợ lý ảo AI theo ngữ cảnh. Kể cả khi chúng ta chỉ có một lượng nhỏ dự liệu huấn luyện, ta cũng sẽ nhận được kết quả phân loại chính xác vào trường hợp như vậy. Vì việc huấn luyện không bắt đầu từ đầu nên việc huấn luyện sẽ diễn ra nhanh, và thời gian lặp sẽ ngắn.

- Tuy nhiên một biểu diễn từ tốt thì không có sẵn cho tất cả các ngôn ngữ vì họ hầu như chỉ huấn luyện các bộ dữ liệu (datasets) có sẵn và thường là bộ dữ liệu bằng tiếng anh. Ngoài ra cũng không bao gồm các từ cụ thể của tên miền như là tên viết tắt, tên sản phẩm… Trong trường hợp này, tốt hơn nên huấn luyện các biểu diễn từ của riêng mình với trình phân loại biểu diễn từ có giám sát (supervised embeddings classifier).

###### 3.4.3.2.1.2 Supervised Embeddings: Intent Classifier TensorFlow Embedding

- Trình phân loại ý định intent\_classifier\_tensorflow\_embedding được phát triển bởi Rasa và ý tưởng được kế thừa từ Facebook’s starspace paper. Thay vì sử dụng các biểu diễn từ được huấn luyện từ trước và huấn luyện một trình phân loại trên đó, intent\_classifier\_tensorflow\_embedding huấn luyện các biểu diễn từ đó từ đầu. Nó thường được dùng với thành phần intent\_featurizer\_count\_vectors để tính tần suất các từ có trong bộ dữ liệu huấn luyện xuất hiện trong tin nhắn và cung cấp như là dữ liệu đầu vào cho trình phân loại. Thay vì sử dụng đếm số lượng từ, ta có thể sử dụng mô hình ngôn ngữ (Ngram) bằng cách thay đổi thuộc tính bộ phân tích của thành phần intent\_featurizer\_count\_vectors thành char.

VD:

Hình 3. Ví dụ về supervised Embeddings với câu how to paint 3d image?

- Một vector khác được tạo ra để đánh nhãn cho ý định. Khác biệt với trình phân phân loại huấn luyện sẵn, trình phân loại tensorflow\_embedding hổ trợ nhận dạng nhiều ý định.

VD: hello, tell me how to resize image.

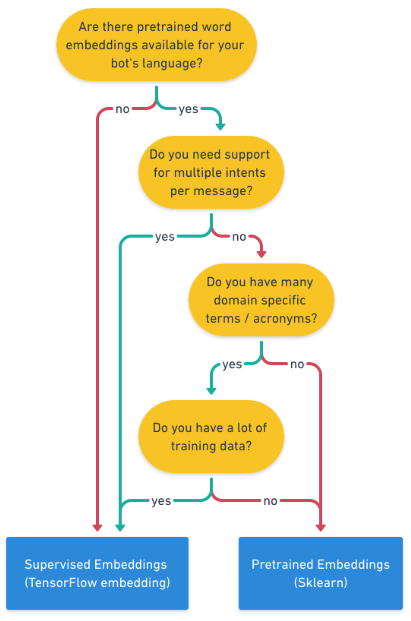
=> Thông điệp của người dùng bao gồm chào hỏi và hỏi về cách sử dụng.

- Trình phân loại này học các biểu diễn từ cho các vector đặc trưng và ý định. Cả 2 cách biểu diễn này đều có cùng số chiều, giúp đo khoảng cách vector giữa các biểu diễn từ đặc trưng và biểu diễn nhãn ý định bằng độ tương đồng cosine. Trong quá trình huấn luyện, sự tương đồng cosin giữa các thông điệp người dùng và nhãn mục đích liên quan được tối đa hóa.

- Intent\_classifier\_tensorflow\_embedding sẽ tối đa hóa sự tương đồng giữa thông điệp người dùng và ý định. Thuật toán này dựa trên StarSpace của Facebook, tuy nhiên có một vài điều chỉnh cho khác đi, thêm vào 2 hidden layers. Đầu ra của thuật toán này là bảng xếp hạng độ tương đồng từ cao đến thấp bao gồm các nhãn không khớp.

• Khi nào nên sử dụng thành phần này:

Bởi vì trình phân loại này huấn luyện dữ liệu từ đầu, do đó dữ liệu cần phải rất nhiều so với các mô hình đã huấn luyện sẵn. Tuy nhiên, khi trình phân loại này huấn luyện mô hình, nó sẽ tập trung (thích nghi) với các tên miền cụ thể có trong các ví dụ - không thiếu biểu diễn từ nào.



Hình 3. Quy tắc cơ bản để chọn trình phân loại ý định

Ngoài ra, nó là một ngôn ngữ độc lập và ta không phụ thuộc vào bất kì biểu diễn từ có kết quả tốt cho ngôn ngữ nhất định. Một tính năng đặc trưng là nó hổ trợ các tin nhắn có nhiều ý định được mô tả như trên. Nói chung điều này làm cho nó một phân loại rất linh hoạt cho các trường hợp sử dụng nâng cao.

##### 3.4.3.2.2 Trích xuất thực thể

- Hiểu được ý định của người dùng là một vấn đề, một vấn đề quan trọng không kém là trích xuất thông tin cần thiết từ thông điệp của người dùng. Đối với bài toán ở đây, chúng tôi chọn Adobe để áp dụng, các thực thể mà chúng tôi quan tâm đến ở đây là các đối tượng ở trong phần mềm Adobe và cụ thể ở đây là Adobe Photoshop. Rasa NLU có nhiều thành phần giúp ta trích xuất thực thể. Ngoài ra nếu ta phát triển được một tool trích xuất thực thể tốt hơn cho bài toán của mình, ta có thể thay thế.

- Là mã nguồn mở, Rasa NLU tập trung đặc biệt vào khả năng tùy biến hoàn toàn. Rasa NLU cung cấp cho chúng ta thành phần nhận dạng thực thể, có thể nhắm tới mục tiêu tùy chỉnh theo yêu cầu của người lập trình.

+ Nhận dạng thực thể với mô hình ngôn ngữ SpaCy: ner\_spacy

+ Nhận dạng thực thể trên quy tắc dùng Facebook’s Duckling: ner\_http\_duckling.

+ Huấn luyện một trình trích xuất cho các thực thể tùy chỉnh: ner\_crf.

###### 3.4.3.2.2.1 Spacy

- Thư viện SpaCy cung cấp các trình trích xuất thực thể được sàn lọc trước. Như là biểu diễn từ, nhưng chỉ một số ngôn ngữ được hổ trợ. Nếu ngôn ngữ bạn đang sử dụng cho bot được SpaCy hổ trợ thì ner\_spacy là một đề xuất cho bạn để nhận dạng các tên thực thể: tên công ty, tên tổ chức, tên người, …

- Đối với bài toán mà dữ liệu của chúng ta ít thì nên sử dụng spacy. Vì sử dụng mô hình pre-trained nên dữ liệu ngôn ngữ sẽ rất phong phú.

VD: What is the image?

What is the picture?

=> Với spacy image và picture sẽ được hiểu như nhau dù không có dữ liệu về picture, và spacy cũng sẽ biết được ý định muốn hỏi về đối tượng image.

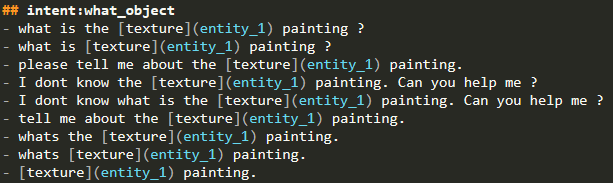
###### 3.4.3.2.2.2 Duckling

- Duckling là một trình trích xuất thực thể dựa trên các luật được phát triển bởi ông lớn Facebook. Giả sử khi bạn muốn trích xuất các dữ liệu kiểu number như số tiền, số tài khoản, khoảng cách,… thì đây là một trình khá phù hợp. Duckling được triển khai trong Haskell và không được các thư viện Python hỗ trợ tốt. Để giao tiếp và sử dụng với Duckling, Rasa NLU sử dụng giao diện REST của Duckling, do đó phải chạy server Duckling khi thêm thành phần ner\_duckling\_http vào trong pipeline. Cách dễ nhất để chạy server Duckling là sử dụ ng docker được rasa cung cấp:

docker run -p 8000:8000 rasa/rasa\_duckling

###### 3.4.3.2.2.3 Ner\_CRF

- Cả ner\_spacy và ner\_duckling đều không yêu cầu bạn chú thích bất kỳ dữ liệu đào tạo nào của bạn, vì chúng đang sử dụng các trình phân loại được xử lý trước (spaCy) hoặc các phương pháp dựa trên quy tắc (Duckling). Thành phần ner\_crf huấn luyện một trường ngẫu nhiên có điều kiện, sau đó được sử dụng để gắn thẻ các thực thể trong thông điệp người dùng. Vì thành phần này được đào tạo từ đầu như một phần của đường ống NLU (NLU pipeline), ta phải tự chú thích dữ liệu đào tạo của mình.



Hình 3. Ví dụ về dữ liệu huấn luyện cho intent what\_object

- Ý tưởng về bài toán này:

Dữ liệu đầu vào của chúng ta là một câu:

x = (x1, x2, …, xn)

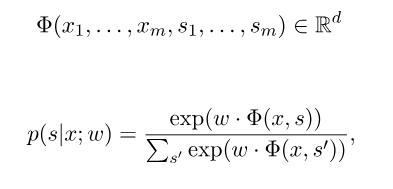
Dữ liệu đầu ra là các thẻ được đặt tên

s = (s1, s2, …, sn)

Mô hình hóa dữ liệu đầu ra dưới dạng xác suất có điều kiện:

P (s1, s2, …, sn |x1, x2, …, xn)

Chúng ta cần xác định một biểu đồ đặc trưng ánh xạ toàn bộ chuỗi đầu vào (x) với toàn bộ chuỗi (s) thành một vector đặc trưng (d chiều). Sau đó có thể sử dụng mô hình nó dưới dạng mô hình log-linear với vector tham số w:



- Mô hình cho CRF rất nhiều bao gồm sklearn\_crfsuite.CRF, mô hình sử dụng mạng neural Bi-LSTM-CRF, … Hiện tại, ner\_crf đang sử dụng là của sklearn và thuật toán Gradient descent sử dụng phương pháp L-BFGS

- Các bước xử lý:

+ Tiền xử lý dữ liệu: tách câu thành những từ đơn lẻ, sau đó tôi chuyển tất cả từ đơn này về dạng nguyên mẫu của nó. Đặc biệt ở đây ta đang đánh nhãn cho cả câu, không phải chỉ riêng bất kì từ nào, vì mỗi đặc trưng đều dựa vào các trực trưng phía trước và sau từ đó, nên ta không được loại bỏ stop word nào.

+ Lọc dữ liệu huấn luyện: lọc tất cả dữ liệu huấn luyện không có tham số trích xuất hoặc mẫu chỉ định NERCRF.

+ Tạo dữ liệu huấn luyện có đánh nhãn sẵn các thực thể (entity) theo một cách nhất quán. Sau đó, ta sẽ chuyển entity trong từ điển của mình thành một danh sách dữ liệu dạng (start\_index, end\_index, entity).

VD: how to paint 3D image => [(7,11,’paint’), (12,19,’3D image’)]

+ Đánh nhãn BILOU: (Begin, Intermediate, Last, Other, Unigram) là một định dạng gắn thẻ văn bản cho phép trích xuất thực thể. Ở bước này, các tokens ở trên sẽ được đánh nhãn vào. Có các nhãn đặc trưng như:

+ Nếu entity là một từ, trước và sau không phải là entity kèm theo thì nhãn là U-.  
 + Nếu entity đó bao gồm 3 từ thì sẽ được đánh nhãn lần lượt là B-, I-, L-

VD: Tokens :: [how to install photoshop]

=> BILOU::[O, O, U-action, U-entity]

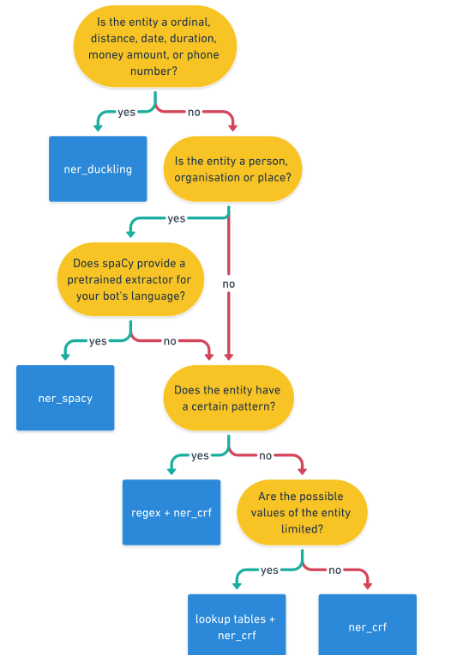
Token :: [what is the background layer]

=> BILOU :: [O, O, O, B-entity, L-entity]

+ Chuyển về định dạng format của CRF: [(token, POS\_tag, entitiy, pattern)]  
 VD: [(“how”, None, ‘O’, None), (“to”, None, ‘O’, None), (“install”, None, ‘U-action’,None), (“photoshop”, None, ‘U-entity’, None) ].

+ Sau đó, ner\_crf sẽ tự sinh ra các đặc trưng bằng các cấu hình có sẵn của sklearn hổ trợ và đó là x\_train, y\_train là nhãn BILOU ở trên. Sử dụng thuật toán **lbfgs** để huấn luyện.

- Ngoài ra, còn có hổ trợ bảng tra cứu (lookup table) cho trường hợp dữ liệu thực thể của bạn có giới hạn. Dưới đây là hình tham khảo về việc lựa chọn thành phần ner nào sẽ có kết quả tốt:

- Ngoài ra, nếu mô hình Rasa NLU của bạn sử dụng là tensorflow\_embedding, chúng ta còn có thể cấu hình các siêu tham số để tìm ra cấu hình tốt nhất.

Hình 3. Quy tắc cơ bản của việc chọn thành phần ner

Ví dụ: language: en

Pipeline:

- Name: "intent\_featurizer\_count\_vectors"

- Name: "intent\_classifier\_tensorflow\_embedding"

epochs: {epochs}

- Trong suốt quá trình tìm kiếm tham số chúng ta sẽ cố gắng tìm ra số lượng epochs tối thiểu. Bạn có thể khởi tạo số phạm vi của tham số mà bạn muốn đánh giá mô hình NLU của bạn.

Ví dụ:

From hyperopt import hp

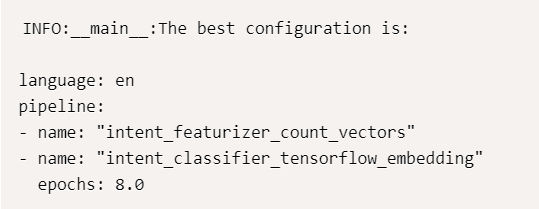
search\_space = {

'epochs': hp.uniform(“epochs”, 1, 10)

}

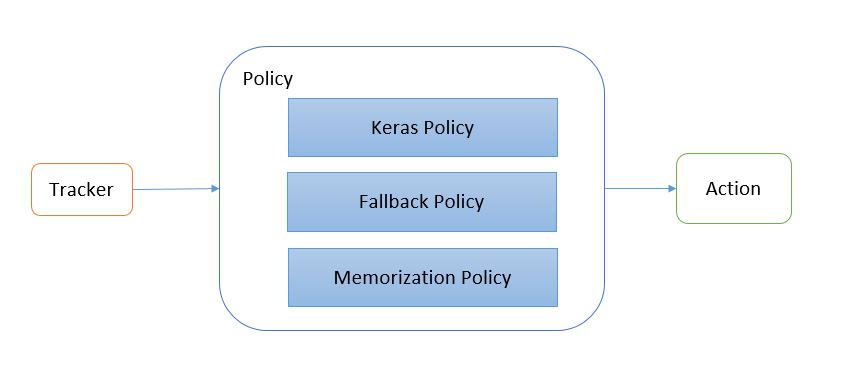
- Với không gian tìm kiếm từ 1-10, mô hình sẽ được huấn luyện với các tham số epochs khác nhau. Ngoài ra còn có các siêu tham số khác có thể kham khảo tại [19]

Rasa NLU cung cấp một tool cho chúng ta đánh giá pipeline phù hợp nhất đối với dữ liệu hiện tại. Sau khi chạy xong tool đó, output cho ra một cấu hình pipeline tốt nhất mà RasaNLU tính toán được.

 Ví dụ: output sẽ tìm ra số epochs là 8.0 sẽ tốt nhất đối với dữ liệu hiện tại.

Hình 3. output sau khi dùng tool đánh giá pipeline của Rasa NLU

### 3.2.4 Rasa Core

- Rasa Core là nơi để quản lý cuộc hội thoại, dự đoán và quyết định xem hành động nào tiếp theo sẽ được gọi. Và mỗi khi một hành động được gọi sẽ được lưu vào tracker, các entity sẽ được lưu vào slots. Các acion tiếp theo diễn ra được quyết định bởi các Policy trong đó Keras Policy là thành phần quan trọng nhất.

Hình 3. 15 Mô hình Rasa core

Hình 3. Các thành phần của RASA Core

- Fallback Policy: thành phần này giúp tự động trả lời 1 action mặc định khi độ tin cậy của việc dự đoán Rasa NLU < một ngưỡng cho trước, hoặc khi độ tin cậy của next\_action < một ngưỡng cho trước, thì Fallback Policy sẽ được gọi và trả respone mặc đinh cho user.

Ví dụ: user: “hey, stupid bot!”: độ tin cậy dự đoán bởi Rasa Nlu < 0,4

bot: “Sorry, I don’t understand that…”

=> Fallback policy sẽ tự động gọi đến action\_fallback trả về câu:” Sorry, I don’t understand that…”

- Keras Policy: Keras Policy sử dụng Neural Network để dự đoán phải hồi của người dùng. Neural Network sử dụng ở đây là LSTM có thể tùy chỉnh hidden layer (thêm, bớt, các hàm activation, loss, …) do người dùng quyết định. Input của Keras Policy là các stories có chứa các hành động liên tiếp mà chúng ta sinh ra nhờ tool hoặc từ dữ liệu thật. Output của Keras Policy có 2 loại:

+ Nếu ta sử dụng **MaxHistoryTrackerFeaturizer** với tham số max\_history = n: “Tracker featurizer” sẽ lấy tất cả các lịch sử trong tracker, chia nhỏ thành từng max\_history nhỏ, sau đó tạo dữ liệu huấn luyện cho LSTM. Đối với tham số này, chiều dài output sẽ bằng 1 vì nó sử dụng tất cả lịch sử trong tracker để dự đoán hành động tiếp theo dựa vào phân lớp cuối cùng.

+ Nếu ta sử dụng tham số **FullDialogueTrackerFeaturizer**: “Tracker featurizer” sẽ lấy tất cả lịch sử trong tracker, tạo thành dữ liệu cuộc đối thoại hoàn chỉnh cho LSTM. Trường hợp này, chiều dài output sẽ bằng 2, tham số này sử dụng hành động trước (previous\_action) để dự đoán cho hành động tiếp theo (next\_action).

# CHƯƠNG 4: CÀI ĐẶT ỨNG DỤNG

*Chương này sẽ trình bày về quá trình xây dựng và cài đặt ứng dụng, từ phần xây dựng cơ sở tri thức đến phần xử lí ngôn ngữ tự nhiên, xây dựng phản hồi và phần giao diện người dùng. Sau đó chúng tôi sẽ trình bày về quá trình deploy, publish ứng dụng để mọi người có thể cùng xem và sử dụng.*

## 4.1 Quá trình chuẩn bị

### 4.1.1 Về dữ liệu ontology

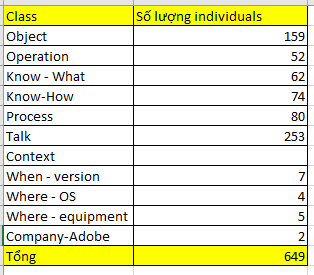
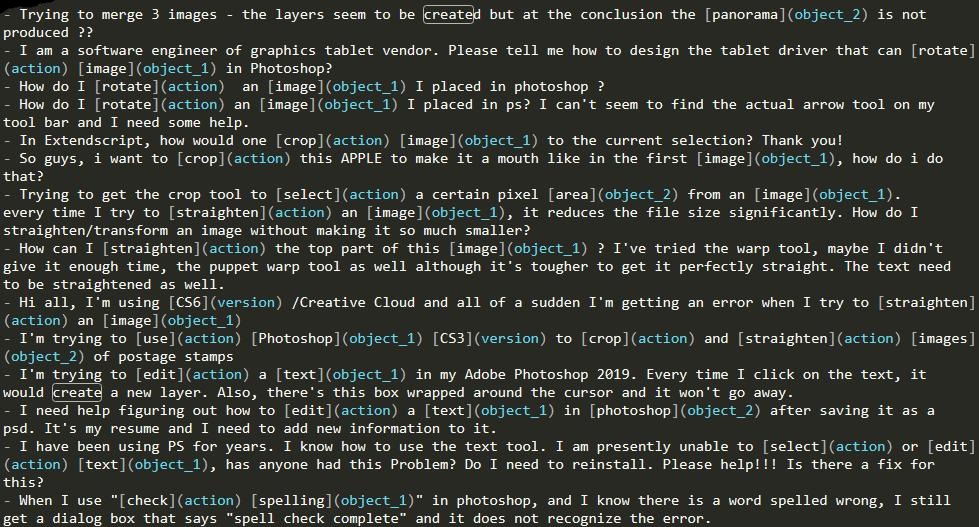
 Dữ liệu dùng cho ontology chủ yếu được lấy trên website hướng dẫn của Adobe Photoshop [12], dựa trên tài liệu thống kê của thầy Thắng, chúng tôi dễ dàng nhận biết đâu là câu hỏi how, đâu là câu hỏi what, và chúng gồm những object hay operation nào. Sau đó chúng tôi truy cập vào website hướng dẫn để lấy câu trả lời, đọc hiểu xem đâu là response, process, talk, image, video và thêm vào ontology. Dưới đây là bảng thống kê lượng dữ liệu đã được thêm vào:

Table 3: Thống kê số lượng dữ liệu ontology đã chuẩn bị

### 4.1.2 Về dữ liệu training xử lí câu hỏi

Dữ liệu training được trích xuất từ forum Adobe Photoshop và các nguồn khác, đó là các câu hỏi của người dùng, được lọc lại để sử dụng làm dữ liệu huấn luyện.

Tổng số câu có được là 1437 câu train lấy từ forum của Adobe và 195 câu test được lấy từ một số nguồn khác như stackoverflow… Sau đó tiến hành đánh nhãn các câu hỏi đã trích được từ forum – được thể hiện dưới dạng json hoặc .md như hình bên dưới:

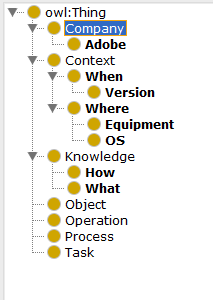
Hình 4. 1 Hình ảnh của dữ liệu được đánh nhãn

## 4.2 Quá trình cài đặt chi tiết

### 4.2.1 Cài đặt ontology

Công cụ sử dụng: Protégé – định dạng file .owl

Với mô hình ontology đã đề xuất ở chương 3.3, chúng tôi tiến hành cài đặt như sau:

* Classes:

Hình 4. 2 Cấu trúc các class trong ontology

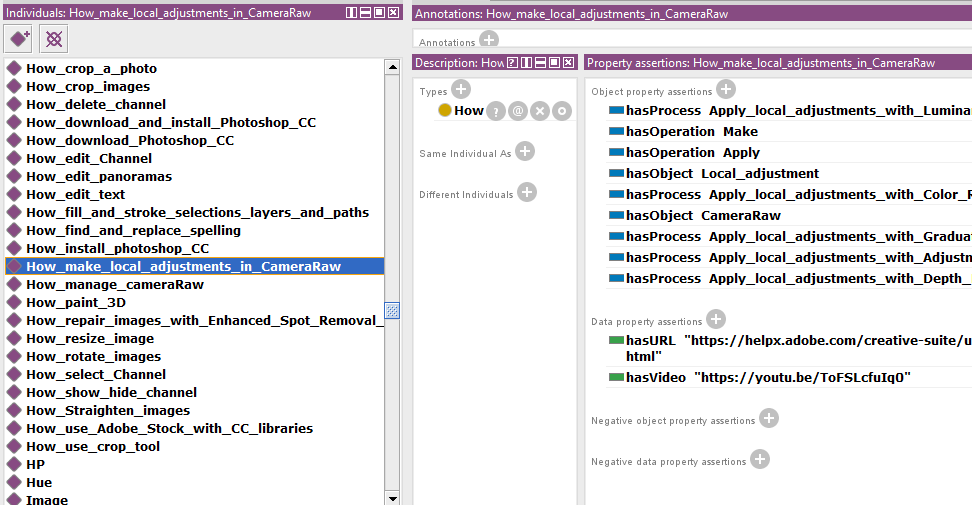
* Inviduals: các invidual có types là “What”, “How”, “Process”, “Task” thì có thể có các thuộc tính (data property) sau:

- Response: Phản hồi bằng văn bản dùng để chatbot trả lời cho người dùng.

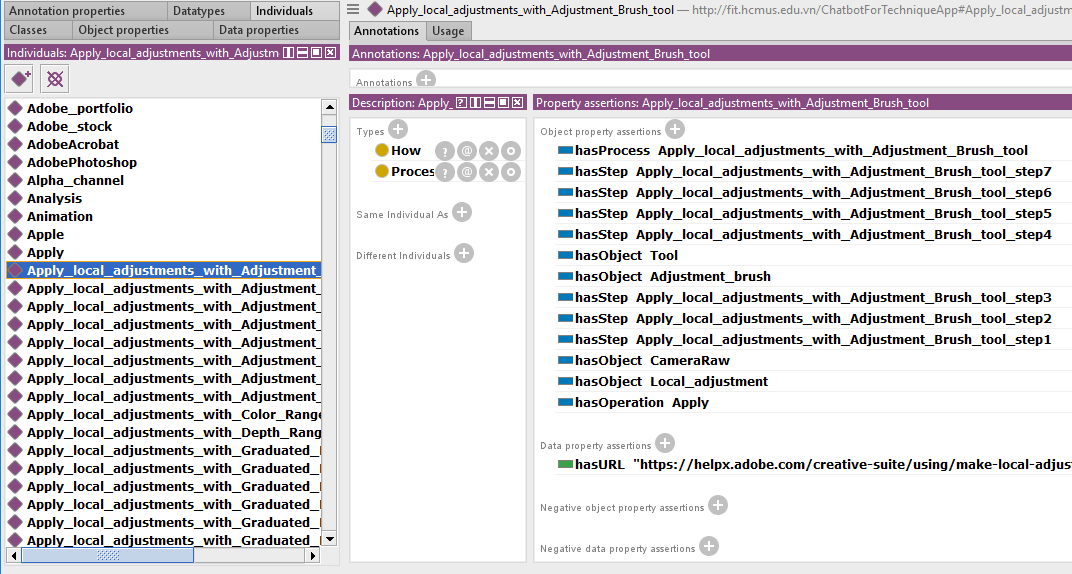
- hasURL: Đường dẫn đến trang tham khảo cho câu hỏi đó. (Task thì không cần vì URL giống với process)

- hasVideo: Đường dẫn đến Video hướng dẫn.

- hasImage: Hình ảnh bổ sung cho câu trả lời, được lưu dưới dạng image base64.



Hình 4. Ành chụp dữ liệu ontology chuẩn bị cho câu hỏi “how mack local adjustments in camera raw”.



Hình 4. Ảnh chụp dữ liệu ontology chuẩn bị cho process “Apply local adjustments with adjustment brush tool” là một process của câu hỏi how trong hình 4.6

Một số ràng buộc nhằm đảm bảo độ chính xác khi truy vấn ontology:

* Một invidual phải có ít nhất 1 types là class.
* Một invidual phải có data property Response hoặc hasVideo thì chatbot mới có thể phản hồi lại nếu invidual này được truy vấn.
* Invidual là Task thì sẽ được đặt tên bằng Invidual Process + “\_step” + STT của Task.
* Invidual mà thuộc class “How” thì bắt buộc phải có ít nhất một mối quan hệ “hasProcess” đến process. Process này có thể là chính câu hỏi how (đối với câu how chỉ có một cách xử lí – vậy thì câu how này cũng phải thuộc class Process)
* Tên các invidual phải được in hoa chữ cái đầu. Tên các object, operation gồm 2 kí tự trở lên gộp lại thì phải in hoa chữ cái đầu và được ghép lại với nhau bằng dấu gạch dưới “\_”.
* Các invidual là how, what hoặc process thì phải có thuộc tính hasURL là đường dẫn đến tài liệu tham khảo. (Luật này chúng tôi áp dụng cho dữ liệu của adobe photoshop, ghi nguồn nhằm mục đích đảm bảo tính chính xác của dữ liệu nhưng với các phần mềm khác nếu không có đường dẫn đến dữ liệu thì cũng không có vấn đề gì, nếu người phát triển sản phẩm có thể tự đảm bảo tính chính xác của dữ liệu).

### 4.2.2 Cài đặt RASA

#### 4.2.2.1 Cài đặt Rasa NLU

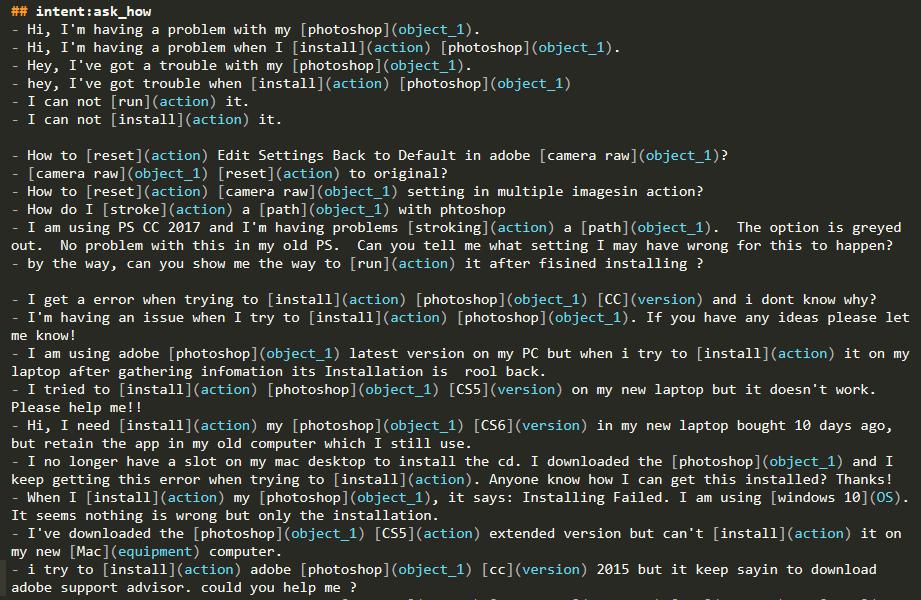
 - Trước hết ta phải chọn các thành phần sao cho phù hợp với bài toán hiện để cấu hình pipeline. Ở đây, ta đang sử dụng một miền riêng biệt là sản phẩm của Adobe cụ thể là Adobe Photoshop. Do đó, pipeline được tôi chọn như sau:

Hình 4. 5 Đoạn code cấu hình pipeline

Hình 4. 5 Đoạn code cấu hình pipeline

Đầu tiên nlp spacy sẽ tách từ thành các từ đơn (tokenizer), sau đó spacy\_tokenizer\_lemma sẽ có nhiệm vụ chuyển các từ này về nguyên mẫu của nó (VD: edits => edit, images => image).

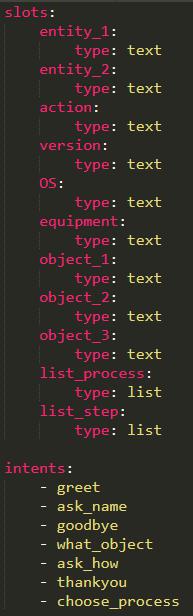
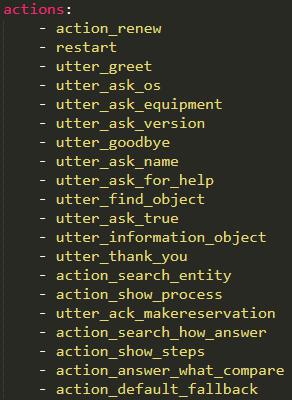
* Thành phần intent\_entity\_featurezier\_regex giúp tăng độ tin cậy cho việc dự đoán ý định. Ner\_crf để rút trích các thực thể. Intent\_featurizer\_count\_vector để chuyển câu đã chuyển dạng ban đầu nhận từ spacy\_tokenizer\_lemma thành vector làm data train cho classifer tensorflow.

Dữ liệu tôi lấy từ các nguồn forum adobe khoảng 1500 câu, đánh nhãn sẵn để làm dữ liệu huấn luyện, bao gồm 7 intents. Nếu một object nào đó chưa đánh nhãn thì sẽ không nhận ra. Hình minh họa:

Hình 4. 6 Dữ liệu train đã đánh nhãn cho ý định ask\_how

Hình 4. 6 Dữ liệu train đã đánh nhãn cho ý định ask\_how

* Khai báo tất cả các slot, action, intents, template vào trong một file gọi là adobe\_domain.yml. Không thể sử dụng các slot, action nếu không khai báo trong file này. Hình ảnh file domail.yml



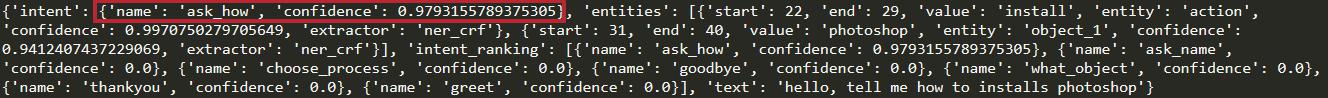
Hình 4. 8 Các actions hiện có

Hình 4. 7 Các slots và intents

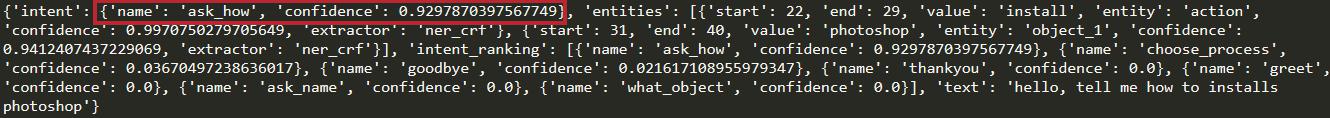


Hình 4. 9 Một số mẫu hội thoại định nghĩa sẵn

Hình 4. 8 Một số mẫu hội thoại định nghĩa sẵn

* Sau khi cấu hình và chuẩn bị dữ liệu hoàn chỉnh, ta sẽ tiến hành huấn luyện cho Rasa NLU, Rasa đã cung cấp đầy đủ các hàm, ta chỉ cần gọi và cung cấp đường dẫn để tiến hành huấn luyện. Sau khi huấn luyện xong chúng ta sẽ tiến hành kiểm tra vài mẫu, ở đây chúng ta sẽ kiểm tra nếu có thành phần regex features và không có.
* VD: hello, tell me how to installs photoshop
* Độ tin cậy (confidence) khi có sử dụng regex features: 0.9739
* Độ tin cậy (confidence) khi không sử dụng regex features: 0.9297

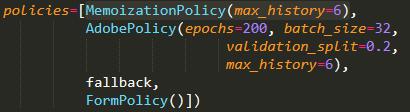
Hình 4. 9 Ảnh chụp độ tin cậy của câu hỏi hello, tell me how to installs photoshop khi sử dụng regex features.



Hình 4. 10 Ảnh chụp độ tin cậy của câu hỏi hello, tell me how to installs photoshop khi không sử dụng regex features.

#### 4.2.2.2 Cài đặt Rasa Core:

Rasa core hiện tại đang sử dụng gồm 3 thành phần policy chính: Keras Policy, Memoization Policy, Fallback Policy.

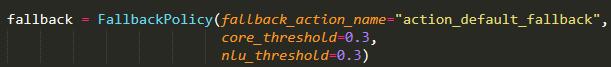


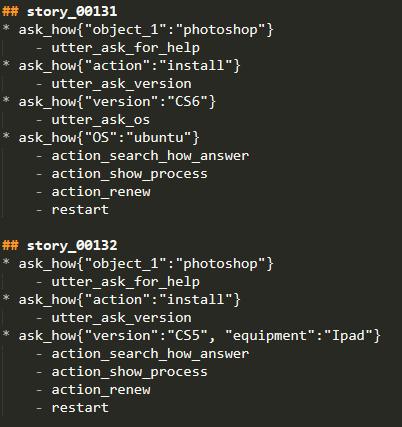
Hình 4. 11 Đoạn code sử dụng MemoizationPolicy

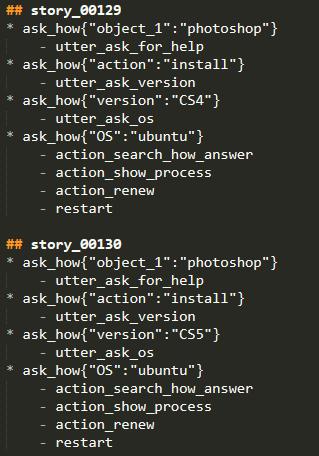
- Keras Policy: cấu hình đang sử dụng cấu hình mặc định của Rasa. Giải thích về keras policy đã đề cập ở trên. Lớp activation cuối để đưa giá trị trả ra (output) về trong khoảng [0, 1] để tiện cho việc dự đoán. Hàm mất mát ở đây là categorical\_crossentropy vì đơn giản ta có nhiều action nên phân lớp sẽ dùng hàm loss này. Các tham số sử dụng max\_history = 6 -> sử dụng output cell cuối cùng của LSTM để dự đoán next\_action. Số epochs = 200, batch\_size = 32, validation\_split =0.2 tức chia dữ liệu huấn luyện thành 2 phần trong đó 80% train và 20% test.



Hình 4. 13 Đoạn code sử dụng keras policy và fallback policy

- Fallback Policy: cấu hình fallback sẽ được gọi action\_default\_fallback khi Rasa NLU dự đoán ý định với độ tin cậy < 0,3 hoặc là rasa\_core dự đoán hành động tiếp theo với độ tin cậy < 0.3.

- Dữ liệu huấn luyện là các stories được sinh ra nhờ công cụ tương tác với chat bot hoặc tự tạo nếu như ta có dữ liệu thật của cuộc đối thoại, được định dạng như sau:

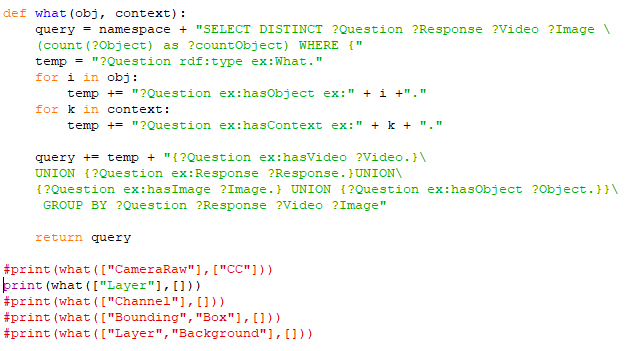
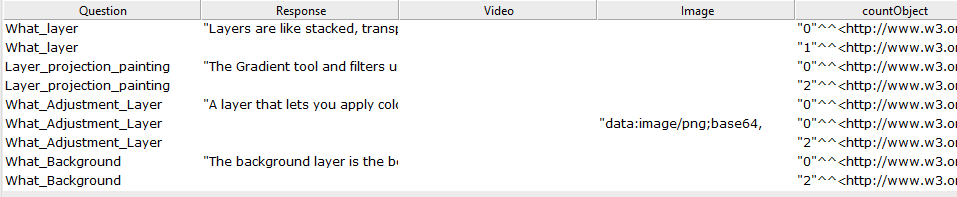


Hình 4. 14 Data stories

### 4.2.3. Kết nối ontology với RASA sử dụng thư viện Owlready2

Owlready2 là thư viện hỗ trợ cho ngôn ngữ Python nhằm kết nối, khởi tạo, truy vấn dữ liệu từ ontology, thư viện này chỉ hỗ trợ với định dạng file được viết bằng ngôn ngữ owl. Chúng tôi sử dụng flask server của python kết hợp với thư viện Owlready2 [20] để tạo một API server truy vấn ontology

Hàm tạo câu truy vấn cho câu hỏi what:



Hình 4. 16 Ảnh chụp kết quả trả về từ ontology cho câu hỏi what is layer.

Hình 4. 15 Hàm chung để truy vấn cho câu hỏi What với đầu vào là danh sách các Object và danh sách các Context dạng mảng các chuỗi string

Kết quả trả về cho hàm what([“Layers”], []) -> Object là layers, không có context.

-> Có nhiều instance được trả ra vì các inviduals trên đề là câu hỏi What và có chứa class object “layer”.

Ý nghĩa các cột:

- Question: Tên các invidual – hay 1 câu hỏi what có thể là ý mà người dùng muốn hỏi.

- Response: Phản hồi dạng văn bản ứng với question.

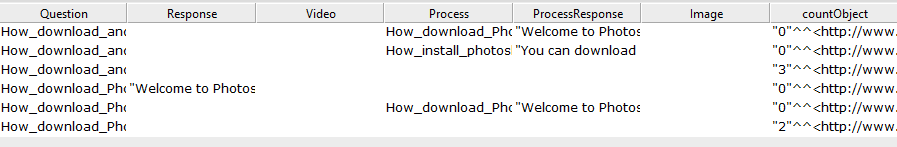
- Video: Link video của question.

- Image: Hình ảnh phản hồi.

- CountObject: Số lượng object và operation ứng với Question (bằng 0 là dữ liệu giả, chỉ tính những số lớn hơn không). Dùng để dự đoán đâu là câu hỏi của người dùng (thường là câu hỏi có số lượng countObject thấp nhất).

Hàm tạo câu truy vấn cho câu hỏi how:

Hình 4. 17 Hàm chung để truy vấn cho câu hỏi How với đầu vào là danh sách các Object, danh sách các Context và danh sách các operation dạng mảng các chuỗi string



Hình 4. 18 Ảnh chụp kết quả trả về từ ontology cho câu hỏi How to download photoshop CC

Kết quả trả về từ hàm query how(["Download"], ["Photoshop"], ["CC"])

-> Object là “Photoshop”, operation là “Download”, context là “CC”.

Ý nghĩa các cột:

- Question: Tên các invidual – hay 1 câu hỏi how có thể là ý mà người dùng muốn hỏi.

- Response: Phản hồi của Question.

- Video: Link video của Question.

- Process: Tên các Process mà Question đó có.

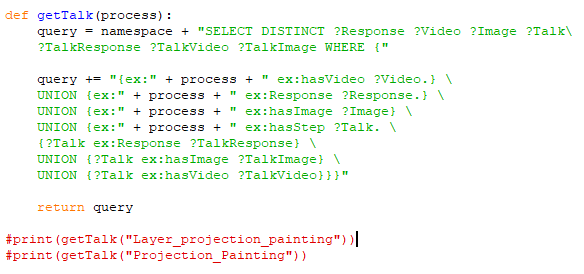
- ProcessResponse: Phản hồi của process.

- Image: Hình ảnh phản hồi.

- CountObject: Số lượng object và operation ứng với Question (bằng 0 là dữ liệu giả, chỉ tính những số lớn hơn không). Dùng để dự đoán đâu là câu hỏi của người dùng (thường là câu hỏi có số lượng countObject thấp nhất).

=> Hàm trên mới chỉ trả về các invidual của câu hỏi how có thể có cho các entity nhận diện được từ câu hỏi của người dùng, chứ chưa lấy các Task của process (nếu có)

Hàm lấy step từ process:



Hình 4. 19 Hàm chung lấy các Task của Process dựa theo tên của process



Hình 4. 20 Ảnh chụp kết quả trả về từ ontology cho process Projection Panting

Kết quả trả về từ hàm query getTalk("Projection\_Painting")

->Process có tên là Projection\_Painting

Ý nghĩa các cột:

- Response: Phản hồi của Process.

- Video: Link video của Process.

- Image: Hình ảnh phản hồi của Process.

- Task: Tên các Task thuộc Process.

- TaskResponse: Phản hồi của Task.

- TaskVideo: Link Video của Task.

- TaskImage: Hình ảnh phản hồi ứng với Task.

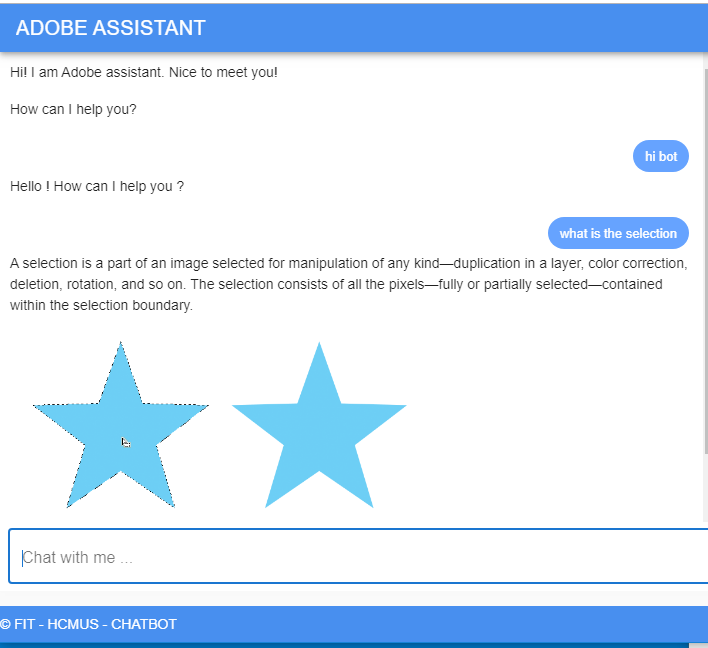
### 4.2.4 Cài đặt giao diện người dùng

Để xây dựng phần giao diện cho chatbot, chúng tôi sử dụng VueJS là ngôn ngữ code giao diện cùng thư viên axios để xây dựng ứng dụng web chat Restful service nhằm gọi API từ phần back-end phía trên lấy kết quả phản hồi cho người dùng.

Theo Wikipedia [21], “*Vue.js, gọi tắt là Vue, là một framework linh động dùng để xây dựng giao diện người dùng. Khác với các framework nguyên khối, Vue được thiết kế từ đầu theo hướng cho phép và khuyến khích việc phát triển ứng dụng theo các bước.*”. Trên trang chủ VueJS [22], Vue được định nghĩa như sau: *“VueJS - Gọi tắt là Vue (phát âm là /vjuː/, giống như****view****trong tiếng Anh), Vue.js là một****framework linh động****(nguyên bản tiếng Anh: progressive – tiệm tiến) dùng để xây dựng giao diện người dùng (user interfaces). Khác với các framework nguyên khối (monolithic), Vue được thiết kế từ đầu theo hướng cho phép và khuyến khích việc phát triển ứng dụng theo từng bước. Khi phát triển lớp giao diện (view layer), người dùng chỉ cần dùng thư viện lõi (core library) của Vue, vốn rất dễ học và tích hợp với các thư viện hoặc dự án có sẵn. Cùng lúc đó, nếu kết hợp với những kĩ thuật hiện đại như [SFC (single file components)](https://vi.vuejs.org/v2/guide/single-file-components.html) và [các thư viện hỗ trợ](https://github.com/vuejs/awesome-vue" \l "components--libraries" \t "_blank), Vue cũng đáp ứng được dễ dàng nhu cầu xây dựng những ứng dụng một trang (SPA - Single-Page Applications) với độ phức tạp cao hơn nhiều.*”

Với việc áp dụng những kĩ thuật hiên đại, Vue và React đang là hai ngôn ngữ lập trình rất được ưa chuộng hiện nay, do đó chúng tôi muốn thông qua đồ án để tìm hiểu về nền tảng này. Vue và React có những tính năng khá giống nhau, nhưng vì Vue phù hợp hơn với việc học tập và dễ dàng để tiếp cận hơn đối với người mới nên chúng tôi chọn sử dụng Vue cho phần giao diện của chatbot.

Sau đây là ảnh chụp màn hình một đoạn hội thoại được hỏi trên giao diện chatbot:

Mặc định khi người dùng truy cập vào trang web thì chat bot sẽ chào hỏi như hình trên. Khi người dùng hỏi what is selection, câu trả lời gồm có Response dạng text và image sẽ được hiển thị lên giao diện.

Hình 4. 21 Ảnh chụp màn hình giao diện chào hỏi và trả lời câu hòi của người dùng có trả về hình ảnh minh họa.

Response trả về khi gọi API từ back end với câu hỏi What is selection là:

*[*

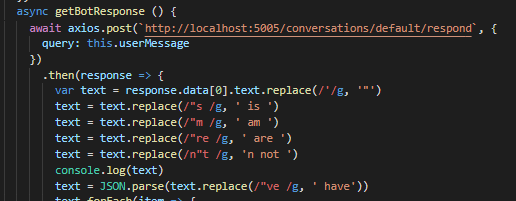
*{*

*"recipient\_id": "default",*

*"text": "[{'respone': 'A selection is a part of an image selected for manipulation of any kind—duplication in a layer, color correction, deletion, rotation, and so on. The selection consists of all the pixels—fully or partially selected—contained within the selection boundary.'}, {'image': 'data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAroAAAGcCAYAAAA24AOvAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAANUTSURBVHhe7L3fk2VXcSY6/5Oe+kERinAEEX7gCb1YEY6wwg+jIMIyAmRsZDDIeLCwsYaZ0dhxucLDaHxtBR6PsI2Fzchj2eJeLI+Hbv1sAWpAUgsJNUhq1D+qu7r3XZk7v9xf5l77VHV3VdepU/lFZKzML7/MfX6stc6qU6fq/JuhUCgUCoVCoVDYQNRBt1AoFAqFQqGwkaiDbqFQKBQKhUJh…..’}]”*

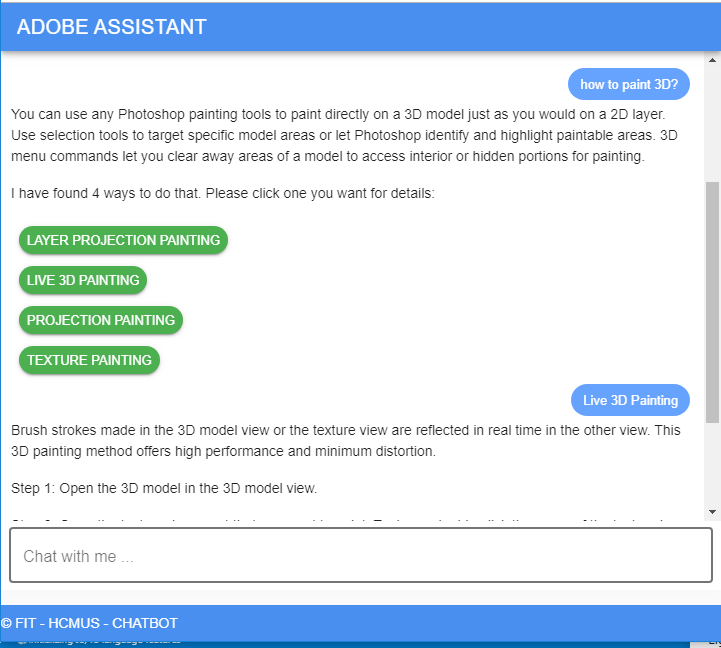
*}*

*]*

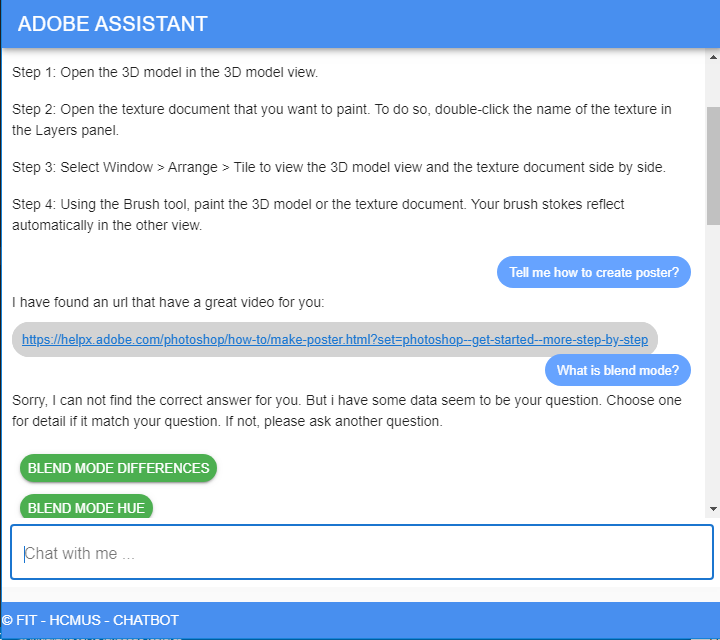


Hình 4. 22 Một phần đoạn code gọi API từ server backend

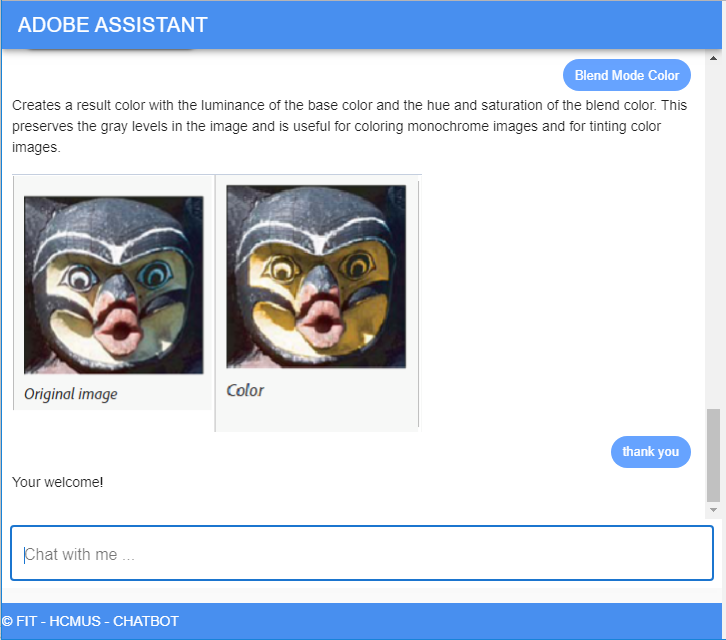
Chúng tôi sử dụng RASA làm backend luôn mà không thêm một server xử lí khác nên không thể trả về một format JSON qua API được, chúng tôi tạo một chuỗi data có dạng json gán vào biến text như trên. Vì vậy sau khi lấy response về từ API chúng tôi tiến hành xử lý chuỗi này và dùng lệnh JSON.parse() để chuyển đổi kiểu string sang dạng object JSON. Sau đó tiến hành lấy phân tích trong text có những kiểu biến nào mà xử lí và hiển thị lên giao diện. Ví dụ trên ta có ‘response’ và ‘image’ sẽ được hiển thị lên giao diện. Những biến khác có thể là ‘video’, ‘process’, ‘talk’, … tùy vào loại biến sẽ có cách hiển thị và lời dẫn khác nhau. Dưới đây là một số hình ảnh minh họa:



Hình 4. 23 Ảnh chụp màn hình giao diện chat khi người dùng hỏi một câu hỏi how gồm nhiều cách xử lý khác nhau.



Hình 4. 24 Ảnh chụp màn hình giao diện chatbot trả lời cách xử lí gồm 4 bước, trả lời câu hỏi có video hướng dẫn và trả lời câu hỏi thiếu dữ liệu.



Hình 4. 125 Ảnh chụp giao diện chatbot đưa ra câu trả lời có hình ảnh sau khi người dùng xác nhận lại câu hỏi thiếu dữ kiện ở hình 4.18 và chào kết thúc đoạn hội thoại

## 4.3 Phát hành, công bố sản phẩm

Để bất kì ai cũng có thể truy cập vào sử dụng ứng dụng cũng như xem và sử dụng lại ontology, chúng tôi cần phải công bố lên địa chỉ public. Phần này sẽ nói về cách chúng tôi công bố toàn bộ ứng dụng của mình, bao gồm công bố ontology, và công bố sản phẩm chatbot lên web.

### 4.3.1 Công bố ontology

#### 4.3.1.1 Công bố ontology để mọi người có thể vào xem

Sau khi tham khảo [23], chúng tôi rút ra được tiêu chí để publish ontology đó là có thể mở được bằng cả trang web (dưới dạng ontology document - tài liệu ontology mà con người có thể hiểu được) và bằng cả các công cụ đọc ontology (protégé, ...). Để làm được điều này chúng ta phải cấu hình cho web hosting của mình sao cho khi client yêu cầu trả về application/rdf+xml thì server phải trả về file ontology, ngược lại thì trả về ontology document.

Một số công cụ dùng để tạo ontology doccument khá tốt là LODE, Parrot, OWLdoc, neologism, Ontospec, hoặc nếu dùng protégé thì phần mềm này cũng tích hợp sẵn một công cụ mở rộng giúp tạo doccument từ file owl khá đẹp (đây cũng chính là cách mà chúng tôi dùng để tạo ontology doccument)

Ban đầu, chúng tôi thử dùng firebase để host ontology của mình, vì đây là một web hosting miễn phí và khá dễ sử dụng, tuy nhiên sau khi host xong thì để cấu hình cho trang web có thể nhận cả khi mở bằng web lẫn công cụ đọc ontology thì cần chỉnh sửa file .htaccess (đây là loại file dùng để cấu hình đường dẫn cho hầu hết các web hosting) giống như hướng dẫn của tổ chức W3C [24] thì phát hiện ra firebase không dùng .htaccess làm file cấu hình mà dùng firebase.json để cấu hình. Chúng tôi đã cố tìm hiểu cách sử dụng file cấu hình của firebase nhưng trong giới hạn tìm hiểu của mình, chúng tôi thấy rằng firebase không hỗ trợ cấu hình được yêu cầu trả về trong phần header của client (để trả về ontology nếu trong header có accept là application/rdf+xml).

Sau khi tham khảo một số web hosting miễn phí khác trên taimienphi.vn [25], thì chúng tôi chọn AwardSpace làm web hosting cho ontology của mình. Theo [25]: “*AwardSpace xuất hiện trên thị trường được khoảng 10 năm và được Tbwhs.com đánh giá khá tốt. Một số tính năng trong gói Web hosting AwardSpace.com bao gồm:*

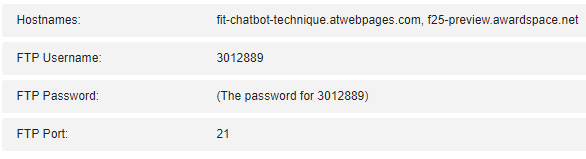
*- Dung lượng lưu trữ: 1 GB.*

*- Lưu lượng truy cập hàng tháng: 5 GB.*

*- Hỗ trợ Domain: 1.*

*- Tài khoản Email: Có.*

*Ngoài ra cung cấp các gói miễn phí, AwardSpace còn cung cấp các gói trả tiền, các gói chia sẻ, VPS và các gói reseller. Mức giá của AwardSpace cũng rẻ hơn so với các nhà cung cấp dịch vụ Web Hosting khác.”*



Hình 4. 136 Thông tin truy cập FPT

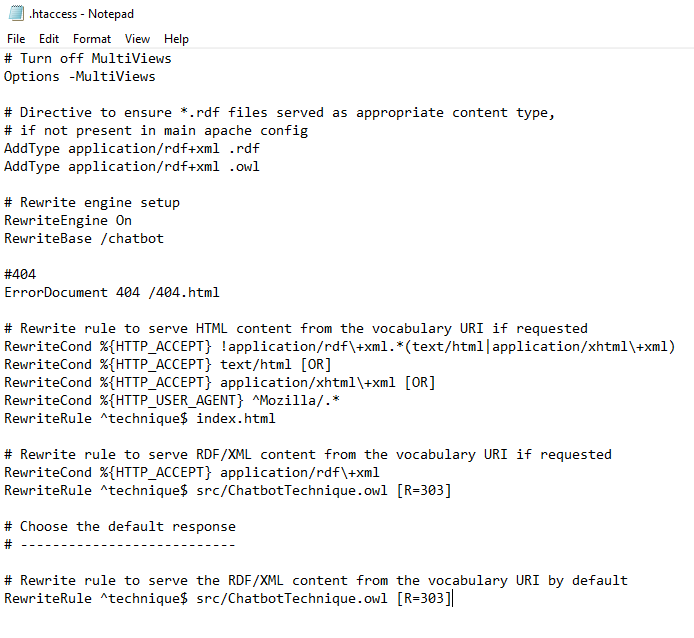
Thông tin FTP đăng kí publish ontology



Hình 4. 147 Cấu trúc thư mục hosting (mở bằng FileZilla)

Nói sơ về cấu trúc thư mục:

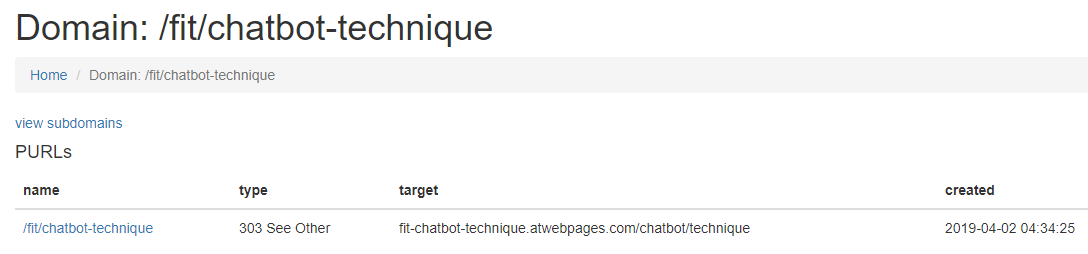
* Index.html là trang hiển thị chính của ontology document.
* 404.html là trang hiển thị thông báo lỗi
* Thư mục src chứa các trang html dùng cho index.html để hiển thị ontology doccument và cả file ontology – ChatbotTechnique.owl



Hình 4. 158 Ảnh chụp cấu hình chi tiết file .htaccess

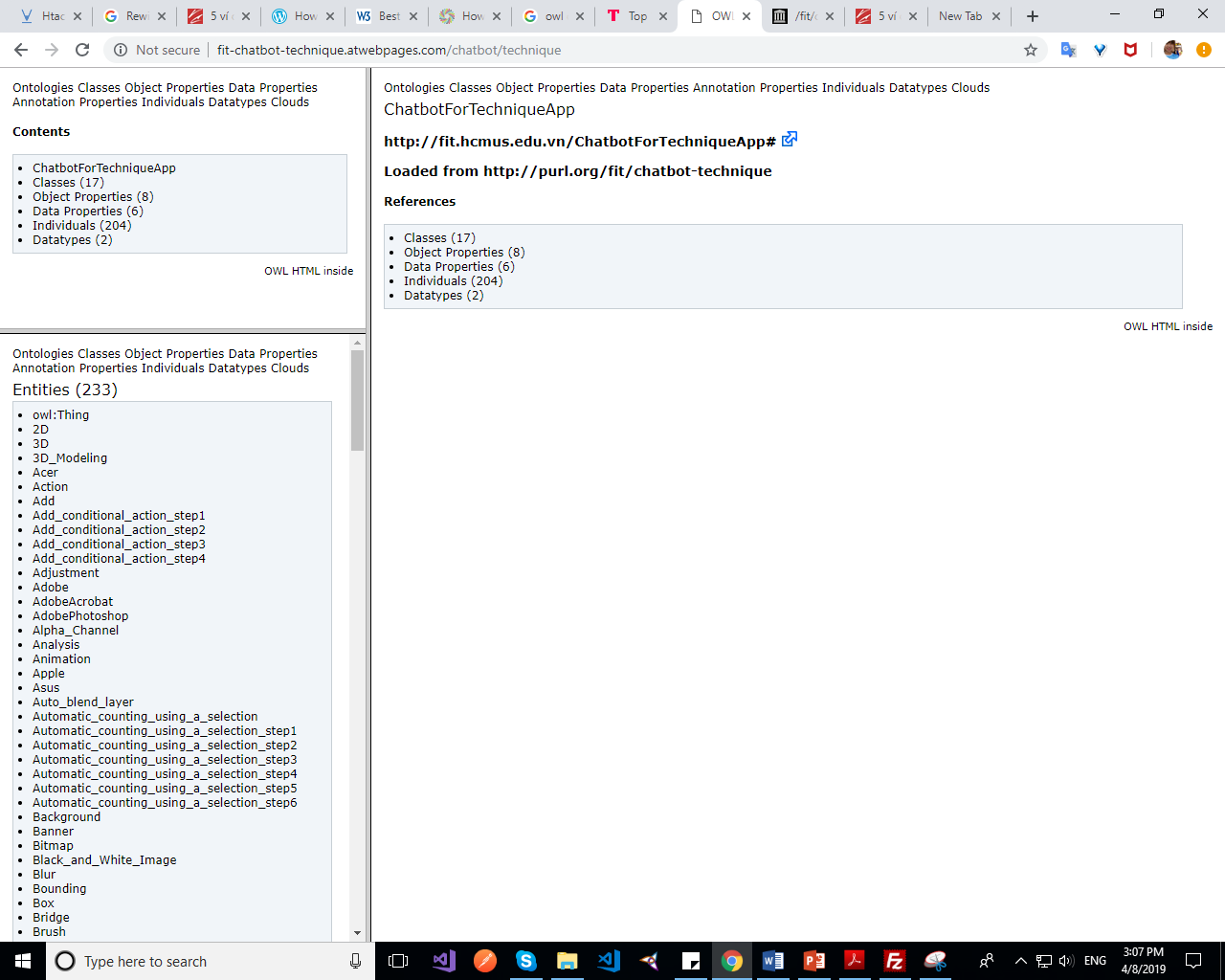
Bước cuối cùng để hoàn thiện việc publish này giúp ontology có thể chuyển hướng đúng thì chúng ta phải sử dụng dịch vụ của PURL [26] để tạo một domain mà có thể hỗ trợ điều hướng như tiêu chí publish ontology đã nêu trên. Với PURL, chúng tôi tạo một domain có URL <http://purl.org/fit/chatbot-technique> có type là 303 See orther, target là đường dẫn đến nơi lưu trữ ontology đã tạo ở các bước trên, ở trường hợp của chúng tôi là

fit-chatbot-technique.atwebpages.com/chatbot/technique (do cấu hình như vậy trong file .htaccess, để hiểu hơn về cách cấu hình file .htaccess chúng tôi tìm thấy một trang blog có 5 ví dụ khá dễ hiểu ở tài liệu [27]).

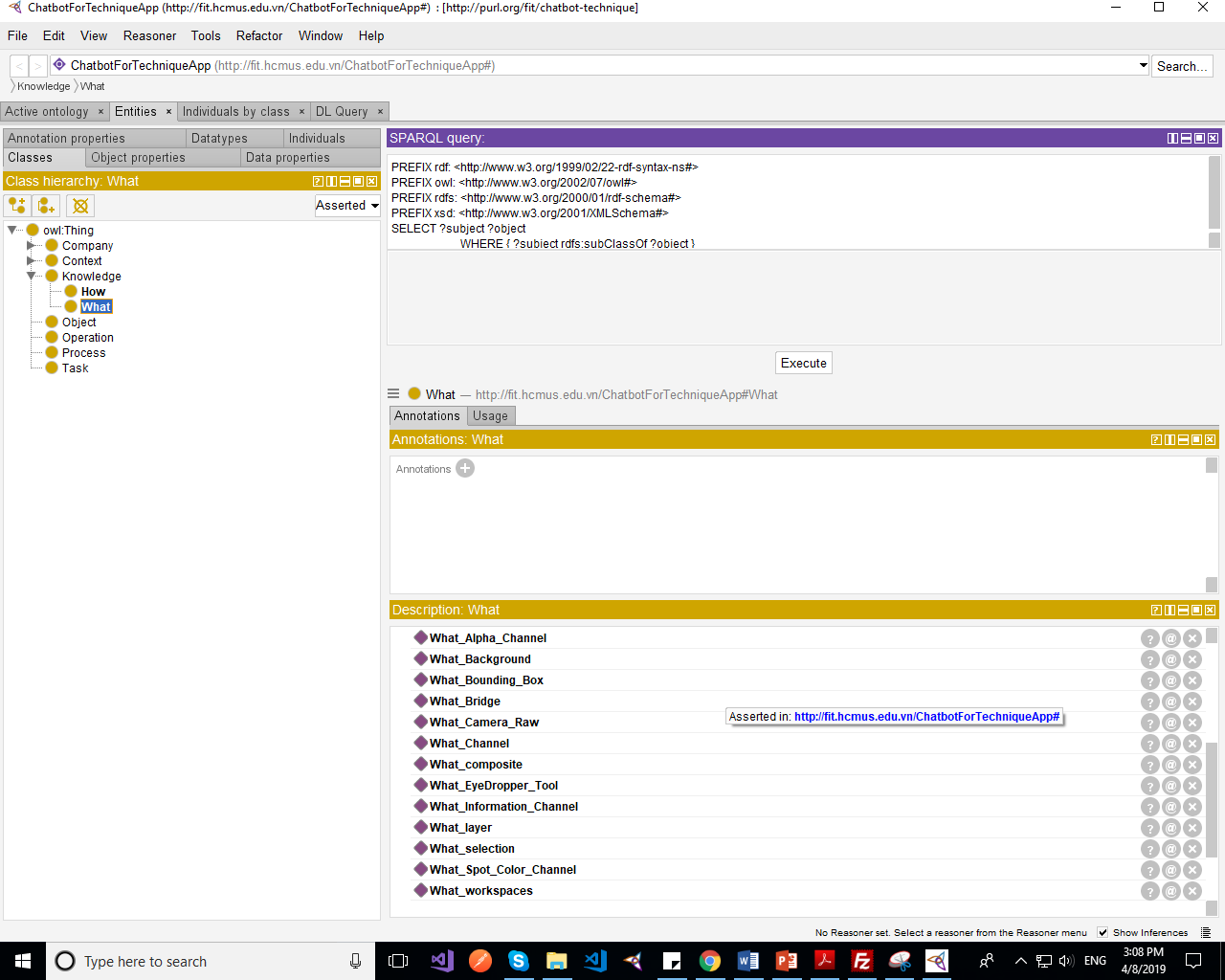


Hình 4. 169 Thông tin domain đăng kí trên PURL

Như vậy là đã hoàn thành xong việc publish ontology, giờ đây mọi người có thể truy cập ontology của chúng tôi qua đường link <http://purl.org/fit/chatbot-technique>



Hình 4. 30 Hình ảnh khi mở đường link trên web

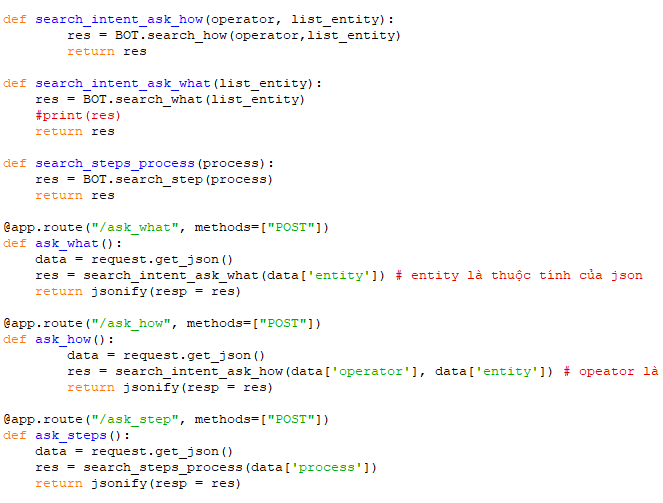


Hình 4. 31 Hình ảnh khi mở đường link bằng công cụ protégé

Ngoài ra, chúng tôi cũng upload file ontology dạng document trên server của trường dưới địa chỉ <https://services.fit.hcmus.edu.vn:251/>

#### 4.3.1.2 Công bố ontology để mọi người có thể truy xuất và chỉnh sửa dữ liệu

Để việc update dữ liệu tốt hơn chúng tôi thực hiện khởi chạy ontology dưới dạng một api server sử dụng ngôn ngữ python, sử dụng thư viện owlready2. Người dùng thông qua API publish có thể dễ dàng gửi request đến server nhằm truy xuất và chỉnh sửa dữ liệu. Tuy nhiên việc truy xuất và chỉnh sửa này phải bó buộc trong những API mà người lập trình cho phép.

Ở trên là đoạn code cho thấy các API hiện chúng tôi đã lập trình để truy cập database đó là ask\_what, ask\_how, ask\_step (đểu là phương thức POST). Ví dụ với API ask\_what, API này nhằm mục đích tìm kiếm các thể hiện của know-what mà có object người dùng đưa vào (đoạn code query database đã viết ở chương 4.2.2.3 – kết nối ontology sử dụng thư viên owlready2)

Hình 4. 32 Một số API truy cập ontology hiện có

Chúng tôi sử dụng server do khoa cấp để publish phần API ontology này. Địa chỉ truy cập API là: https://services.fit.hcmus.edu.vn:274/

### 4.3.2 Phát hành ứng dụng

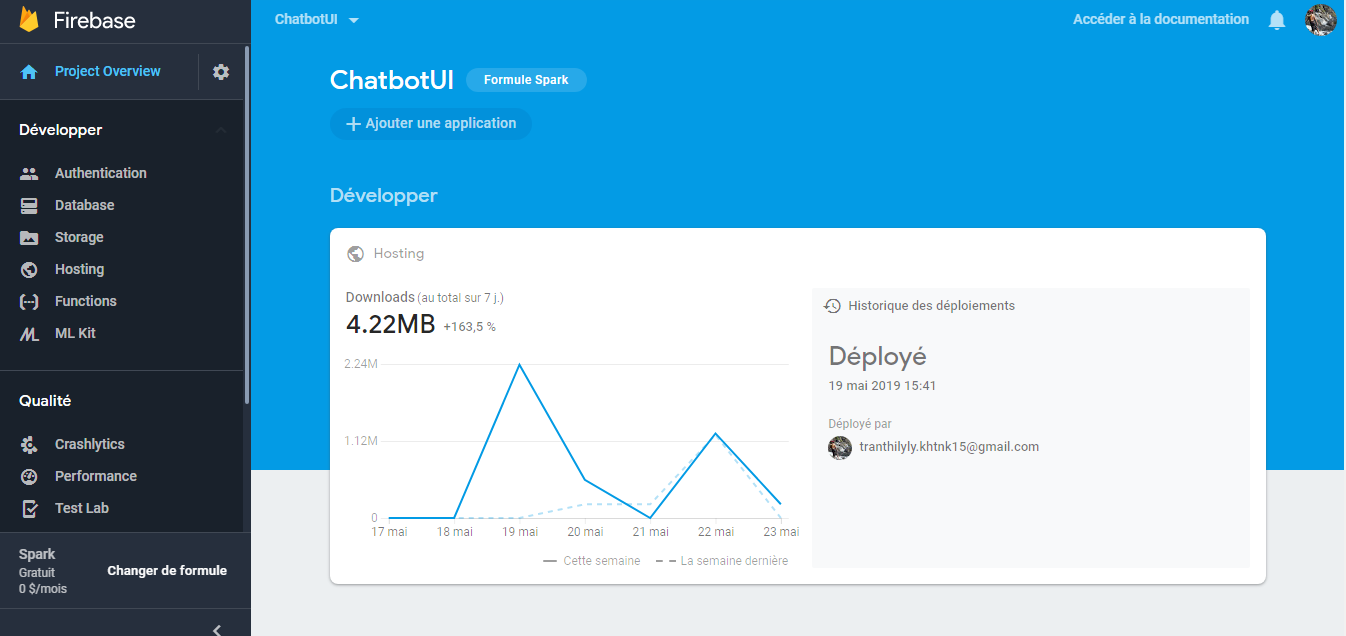
#### 1. API backend

Chúng tôi sử dụng server do khoa cấp để publish phần API backend sử dụng RASA này. Địa chỉ truy cập API là: https://services.fit.hcmus.edu.vn:272/chat

#### 2. Giao diện người dùng

Với mục đích tìm hiểu các công cụ hosting hiện có trên internet, chúng tôi chọn firebase của google là một trong các free web hosting được sử dụng nhiều nhất hiện nay để publish phần giao diện của mình. Sau khi sử dụng firebase để publish ứng dụng vueJS của mình thì chúng tôi thật sự rất thích công cụ này bởi tính dễ sử dụng của nó. Để deploy một website trên firebase rất đơn giản, chỉ cần đăng kí một tài khoản firebase, tạo một domain, tải firebase CLI về máy chứa project cần hosting, sau đó chỉ với 3 dòng lệnh đơn giản là có thể truy cập URL để vào giao diện sử dụng được rồi.

* Firebase login – đăng nhập vào tài khoản firebase cho máy local.
* Firebase init – khởi tạo thư mục firebase hosting.
* Firebase deploy – deploy trang web lên domain đã tạo.



Hình 4. 33 Ảnh chụp trang firebase quản lý project ChatbotUI mà nhóm đã đăng kí và deploy

Link giao diện chatbot được host trên firebase: <https://chatbotui-187b1.firebaseapp.com>, tuy nhiên firebase sẽ đổi tên miền từ firebaseapp.com sang web.app vào 28/5, vậy nên sau thời gian này địa chỉ truy cập vào chatbot chính thức sẽ là: <https://chatbotui-187b1.web.app>

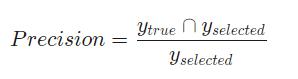
# CHƯƠNG 5: THỰC NGHIỆM VÀ ĐÁNH GIÁ SẢN PHẨM CHATBOT

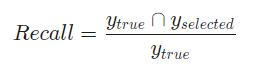
*Phần này sẽ nói về kết quả thực nghiệm, đánh giá cuối cùng của sản phẩm. Để đánh giá ứng dụng, chúng tôi có tham khảo một số bài báo về đánh giá chatbot hay QAS và tổng hợp ra một số tiêu chí phù hợp với ứng dụng hiện tại từ* [28] *và* [29]*. Chúng tôi đánh giá sản phẩm dựa trên ba tiêu chí chính đó là đánh giá thuật toán (rasa NLU), chỉ số chất lượng hội thoại (dialogue quality metric) và mức độ hài lòng của người dùng (users' satisfaction).*

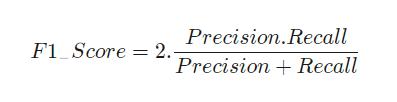
## 5.1 Đánh giá thuật toán Rasa NLU

* Để đánh giá mô hình rasa NLU ta lần lượt đánh giá các thành phần: trích xuất thực thể (entity extraction), phân loại ý định (intent classification).
* Để đánh giá toàn diện chatbot, ta đánh giá thêm các yếu tố: thời gian phản hồi chatbot, độ hài lòng của người dùng.

Đánh giá mô hình rasa NLU dựa trên 3 độ đo: F1-score, Precision, Recall. Trong đó:  
+ Precision: tỉ lệ số điểm **true positive** trong số những điểm **được phân loại là positive**



  
+ Recall: tỉ lệ số điểm **true positive** trong số những điểm **thực sự là positive**

+ F1 là một **trung bình điều hòa (harmonic mean)**của các tiêu chí *Precision* và *Recall*

1/ Đánh giá intent classification (intent\_classifer\_ tensorflow\_embedding):

* Số lượng tập train và test cho từng loại ý định:

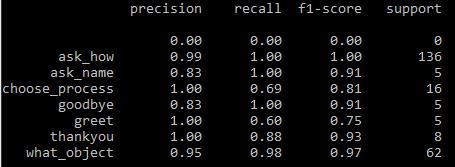
|  |  |  |  |
| --- | --- | --- | --- |
| STT | Ý định | Data train | Data test |
| 1 | greet | 13 | 5 |
| 2 | what\_object | 683 | 62 |
| 3 | Ask\_how | 728 | 136 |
| 4 | Choose\_process | 26 | 16 |
| 5 | Thankyou | 24 | 8 |
| 6 | Ask\_name | 6 | 5 |
| 7 | goodbye | 8 | 5 |
| Tổng | | 1488 | 237 |



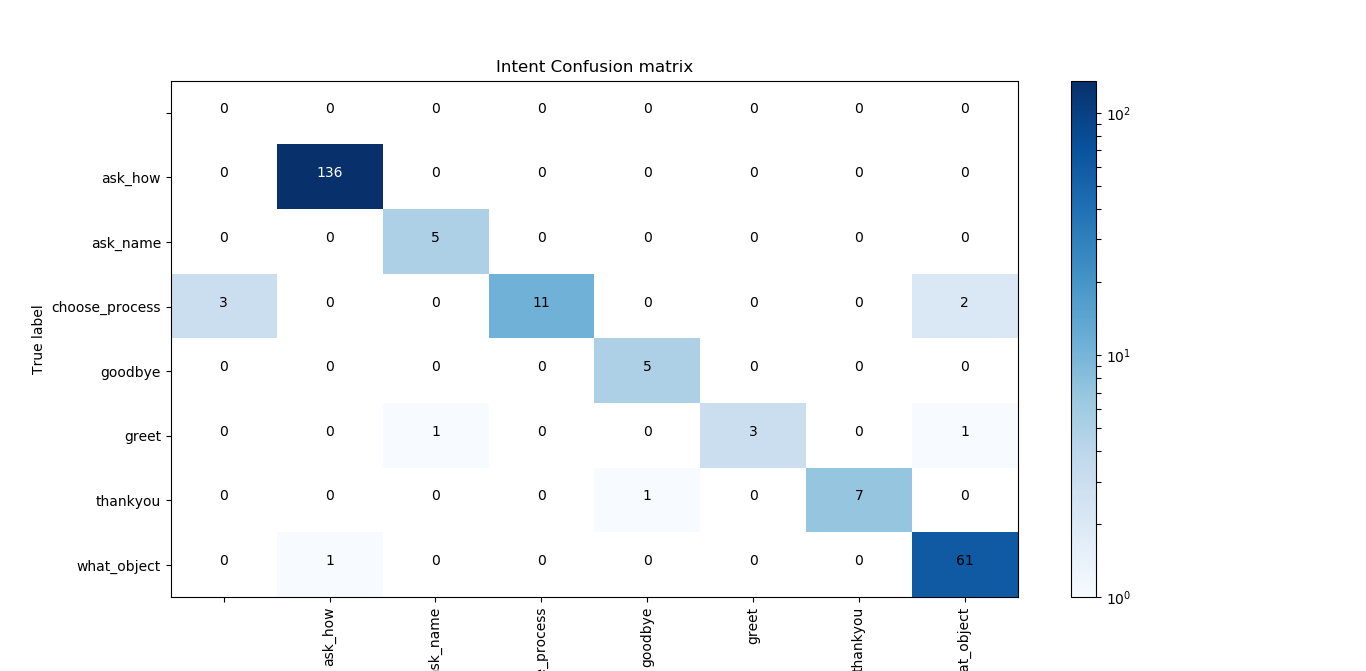
Hình 4. 34 Kết quả đánh giá trình phân loại ý định: Tensorflow classifier

Kết quả:

* Precision: tỉ lệ phân loại ý định chính xác trên tổng số kết quả: 97,65%
* Recall: tỉ lệ phân loại ý định là chính xác đúng với thực tế: 96,2%
* F1-Score: là kết quả trung bình của Precision, Recall 96,57%.
* Giải thích chi tiết:



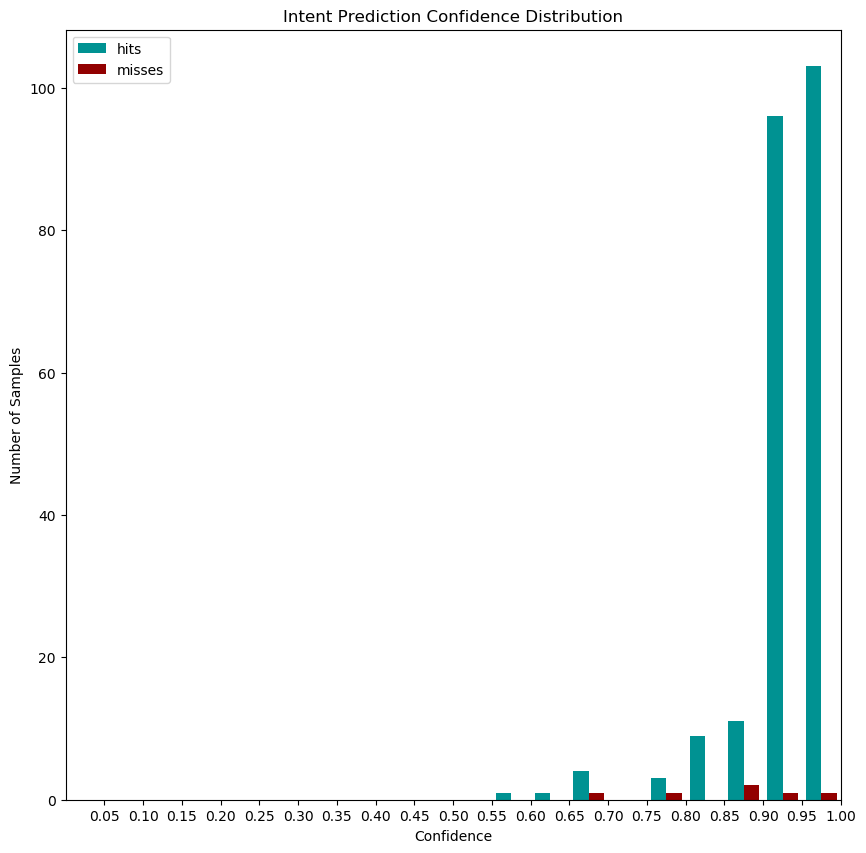
Hình 4. 35 Ma trận kết quả dự đoán với từng ý định người dùng



*Hình đánh giá chi tiết từng intent.*

* Dựa vào ma trận dự đoán kết quả (Hình 4.35) ta có thể thấy:  
  + Chọn intent what\_object:  
   - precision(what\_object): số dự đoán là intent what\_object trên tổng số mẫu dự đoán là what\_object:  
   => precision(what\_object) = 61/65 = 0.95 (vì có 1 phân lớp goodbye, 2 phân lớp choose\_process sai nằm ở cột ask\_object nên tổng số mẫu của what\_object là 65)  
   - recall(what\_object): tổng số mẫu đúng là đúng trong số các mẫu test.  
   => recall(what\_object) = 61/62 = 0.98 (ta có 62 mẫu test, trong đó có 61 mẫu chính xác).  
   - F1-score = 2\*0.98\*0.95/ (0.98+0.95) = 0.97

Hình 4. 36 Biểu đồ phân phối kết quả độ chính xác dự đoán đúng /sai tensorflow\_embedding



2/ Đánh giá entity extraction (ner\_crf):

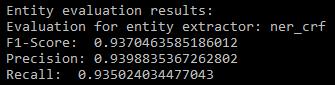
* Bảng số lượng thực thể (entity).

|  |  |  |
| --- | --- | --- |
| STT | Thực thể | Số lượng |
| 1 | Entity | 159 |
| 2 | action | 52 |
| 4 | Version, equipment, OS | 16 |

* Bảng data train và test:

|  |  |  |
| --- | --- | --- |
|  | Data train | Data test |
| Số lượng | 1437 | 195 |

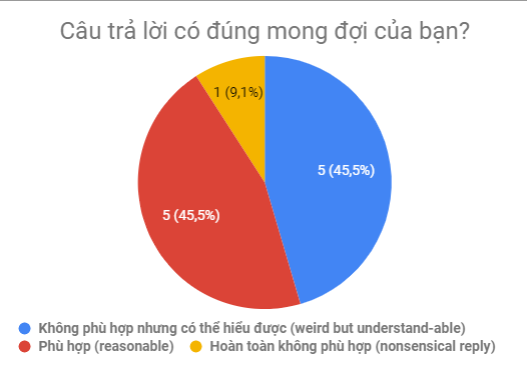
* Kết quả đánh giá mô hình:



Kết quả đánh giá hình trên:

* Precision: ta có 93,9% trích xuất thực thể chính xác trong số tổng số kết quả trích xuất được
* Recall: tỉ lệ thực thể trích xuất là đúng và đúng với thực tế là 93,5%
* F-score: là kết quả trung bình của Precision, Recall (93.7%).

## 5.2 Chỉ số chất lượng hội thoại (dialogue quality metric)

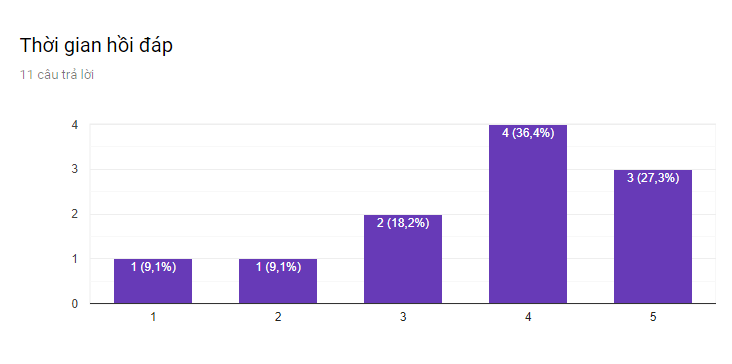
 Trong phần thực nghiệm này chúng tôi nhờ 11 người dùng thử chatbot, trong đó có 4 người chưa từng dùng phần mềm adobe photoshop, 4 người đã từng dùng nhưng thời gian dưới một năm và 3 người có kinh nghiệm sử dụng adobe photoshop trên một năm. Trước khi dùng thử chatbot chúng tôi có cho họ biết về miền kiến thức đã chuẩn bị (danh sách các câu hỏi what how chỉ gồm object, operation và context để người dùng có thể thoải mái đặt câu hỏi theo cách của họ), chúng tôi cho phép họ hỏi bất cứ câu hỏi nào tuy nhiên có nói trước là chỉ những câu đã chuẩn bị mới có đáp án chính xác. Sau khi dùng thử và đánh giá thì sau đây là kết quả đánh giá:

Hình 4. 37 Biểu đồ khảo sát đánh giá của người dùng về chất lượng hội thoại (dialogue quality metric)

## 5.3 Mức độ hài lòng của người dùng (users' satisfaction)

Cũng với 11 người trên, chúng tôi hỏi người dùng năm vấn đề để đánh giá mức độ hài lòng, các câu được đánh giá theo thang điểm từ 1 đến 5 cho độ hài lòng tăng dần, cụ thể về thời gian hồi đáp, độ phù hợp của câu trả lời (câu trả lời có đúng với yêu cầu tìm kiếm?), độ dễ sử dụng (có khó để bạn tìm ra câu trả lời phù hợp khi sử dụng chatbot?), giao diện, sự hài lòng (bạn có cảm thấy thoải mái khi sử dụng chatbot?). Sau đây chúng tôi sẽ trình bày kết quả khảo sát của từng phần và nêu lý do cũng như cách khắc phục.

### 5.3.1 Thời gian hồi đáp



Hình 4. 38 Biểu đồ thống kê đánh giá của người dùng về thời gian hồi đáp trên thang điểm 5

Thời gian phản hồi của hệ thống hiện tại:

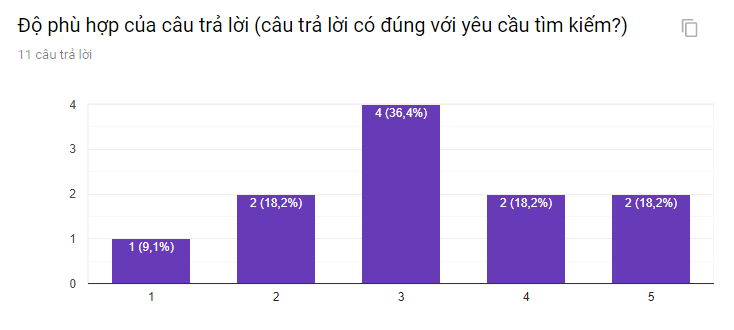
- Server đang sử dụng: Ram 4GB  
- Thời gian phản hồi một số trường hợp:



Hình 4. 39 Thời gian phản hồi một số trường hợp với hệ thống hiện tại khi thực hiện chạy trực tiếp trên server.

Về thời gian phản hồi, chúng tôi thấy phần đánh giá khá tốt, tuy nhiên vẫn có 1 người cho 1 điểm, có lẽ là do phần giao diện người dùng chưa thật sự làm cho người dùng có cảm giác là hệ thống chạy nhanh (sẽ nói rõ hơn ở phần sau). Kết quả khi chạy trực tếp trên server theo từng mẫu hội thoại thì thời gian trung bình mỗi câu hỏi server phản hồi chưa đến một giây. Điều này tùy thuộc vào mỗi câu hỏi, các yêu tố ảnh hưởng sẽ gồm dữ liệu lưu trữ database (ví dụ khi truy vấn câu trả lời có hình ảnh thì thời gian sẽ chậm hơn), trong quá trình xử lí câu hỏi nếu câu hỏi thuộc những trường hợp hệ thống chỉ cần phản hồi lại theo mẫu câu có sẵn thì quá trình này sẽ nhanh hơn, ….

### 5.3.2 Độ phù hợp của câu trả lời



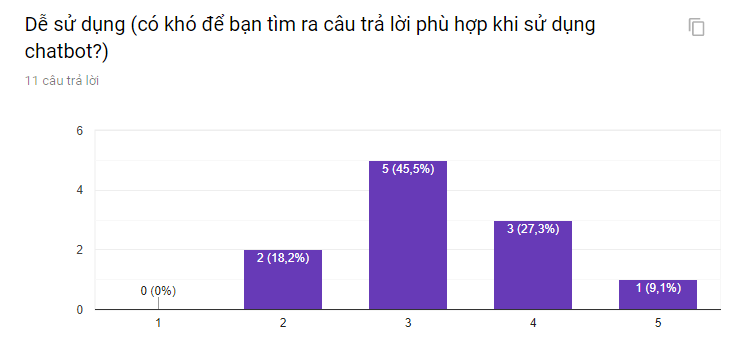
Hình 4. 40 Biểu đồ thống kê đánh giá của người dùng về mức độ phù hợp của câu trả lời trên thang điểm 5

Nhận xét: Biểu đồ cho thấy người dùng đánh giá câu trả lời chỉ ở mức độ trung bình. Một số phản hồi từ người dùng:

|  |  |
| --- | --- |
| Phản hồi người dùng | Giải thích |
| - Đôi khi Adobot sẽ hỏi lại về context như hệ điều hành (operating system), phiên bản photoshop đang sử dụng (version) hay máy tính bạn đang sử dụng (equipment). | - Vì phần thiết kế kịch bản (stories) hiện tại cần yêu cầu tất cả thông tin, sau đó khi kịch bản kết thúc, tất cả mọi context, entity được đệ đến sẽ không còn được ghi nhớ nữa. Vì đôi khi user có thể đang sử dụng window 10 mà hỏi về cách cài đặt trên linux hay ubuntu chẳng hạn. Có thể thay đổi theo ý muốn của mình. |
| - Nhiều câu trả lời không đúng mong đợi  - Cần trả lời được nhiều câu hỏi hơn, câu trả lời cần đúng trọng tâm hơn | - Về phần phân loại ý định và rút trích thực thể thì tỉ lệ khá cao, tin chắc là phần này không phải vấn đề gây ra các sai sót đó. Thay vào đó, việc quản lý query bằng các thực thể rút trích được thì cần điều chỉnh lại, hiện tại nhóm chưa giải quyết được một số câu khó như: “I just know how to use photoshop, but I don’t know how to edit the large image”. Thực thể rút trích được là đầy đủ nhưng ý định ở đây là “edit large image” là chính, vấn đề này nhóm đã có cách giải quyết và sẽ cải tiến sau. |
| - Câu trả lời dư, chưa chính xác. Cần cải thiện thêm | - Trường hợp này đôi khi xảy ra với các entity không được học từ trước.  VD: drop photo in CS5 ubuntu.  - Entity [drop], [photo] không được học từ trước do đó ner\_crf chỉ nhận ra được [CS5], [ubuntu] để rút trích truy vấn. Dĩ nhiên kết quả trả về sẽ vô cùng nhiều mà không liên quan đến câu hỏi của người dùng. Nhóm sẽ khắc phục vấn đề này. |

### 5.3.3 Độ dễ sử dụng:

Mục tiêu của tiêu chí này, chúng tôi muốn biết là người dùng có cảm thấy dễ để tìm ra câu trả lời khi sử dụng chatbot hay không.



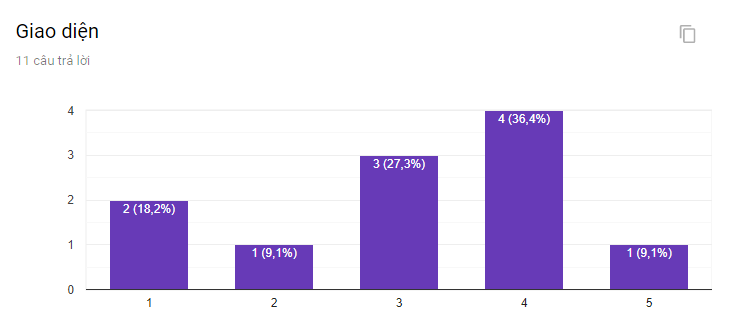
Hình 4. 41 Biểu đồ thống kê đánh giá của người dùng về độ dễ sử dụng của chatbot trên thang điểm 5

Theo chúng tôi đây sẽ là phần được đánh giá khá cao vì chắc chắn việc trò chuyện để tìm ra câu trả lời sẽ dễ dàng hơn tự mình mày mò nhiều. Tuy nhiên kết quả chỉ ở mức trung bình, một số nhận xét liên quan là:

* “*Cứ hỏi lại version và operation system hoài khiến tự nhiên mình đang hỏi nó lại phải đi trả lời cho nó.”*
* *“Cần cải thiện phần nhận biết câu hỏi”*

Về nhận xét hệ thống cứ hỏi lại version và operation system hoài, chúng tôi khá phân vân bởi vì có thể rơi vào trường hợp nếu chỉ hỏi version và operation system một lần, những lần sau không hỏi nữa thì liệu có thật sự chính xác, nhỡ người dùng muốn sử dụng hai version trên hai thiết bị khác nhau giữa hai lần hỏi. Tuy nhiên chúng tôi cũng cảm thấy đây là một phần thiếu sót lớn của hệ thống khi chưa thể xử lí tốt vấn đề này. Do khả năng còn hạn chế, chúng tôi cũng chưa thể xử lí đa luồng khi có nhiều người dùng cùng truy cập vào hệ thống, vì thế đây vẫn còn là phần mà chúng tôi cần phải cố gắng để cải thiện ứng dụng cho tốt hơn.

### 5.3.4 Giao diện người dùng



Hình 4. 42 Biểu đồ thống kê đánh giá của người dùng về độ dễ sử dụng của chatbot trên thang điểm 5

Về giao diên, điểm đánh giá khá thấp, có đến 2/11 người cho điểm tệ nhất là 1. Một số nhận xét, đề nghị cải thiện của người dùng về giao diện như sau:

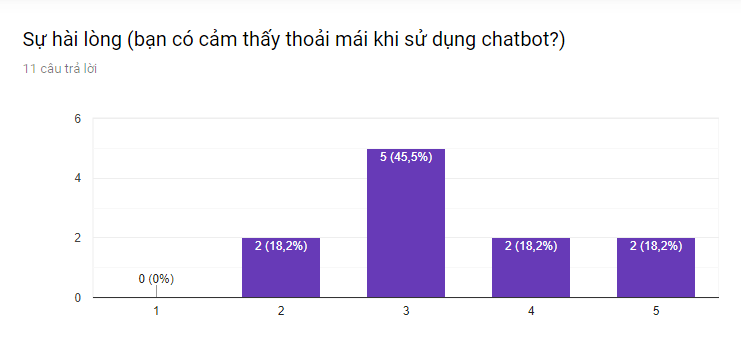
* *“Sau khi nhắn xong thì clear message ở dòng nhập câu hỏi. Nhấp vào url quay lại bị refresh trang.”*
* *“Giao diện hơi chuối. Nt xong mà nó không có status đang chờ...”*
* *“Nhấn enter thì textbos nên empty, chatbot khì làm khung chat bên dưới thôi, cái phần bên cạnh có thể dể html giới thiệu cái gì đó”*
* *“This app will improve UI friendly with user, otherwise please make sure remove the response after user reply question”* – Tạm dịch: ứng dụng nên cải thiệu giao diện người dùng thân thiện hơn, mặc khác hãy chắc chắn xóa phần phản hồi sau khi người dùng trả lời câu hỏi.

Phần giao diện nhận được khá nhiều nhận xét, đánh giá, có vẻ như là vì người dùng là những người biết sử dụng photoshop, một số còn là lập trình viên front-end nên phần giao diên được đánh giá rất chính xác và có tâm. Chúng tôi công nhận phần giao diện chưa thật sự đẹp và thân thiện với người dùng do khả năng còn hạn chế, chưa có nhiều kinh nghiệm trong phần lập trình và thiết kế giao diện thân thiện với người dùng. Sau khi tổng hợp nhận xét thì chúng tôi nhận thấy cần cải thiện những điểm sau:

* Tạo biểu tượng như chatbot đang gõ phím (typing spinner) sau khi người dùng nhập tin nhắn xong và chờ phản hồi từ chatbot.
* Xóa tin nhắn sau khi người dùng nhập xong và không cho người dùng nhập tiếp cho đến khi nhận được phản hồi từ chatbot.
* Mở tab mới trên trình duyệt khi người dùng nhấp vào đường dẫn mà chatbot phản hồi cho người dùng.
* Đối với người dùng sử dụng máy tính để truy cập trang web thì nên thu nhỏ màn hình chat lại cho đẹp.

Tất cả những vấn đề trên đều có thể giải quyết bằng vueJS, tuy nhiên cần phải thiết kế lại khá nhiều phần trong code. Vì trước đây để cho tiện, chúng tôi chỉ thực hiện xử lí tất cả trên một trang, phần gọi api cũng phải tách ra thành 3 giai đoạn khởi đầu (start), thành công (success), lỗi (fairlure) sau đó dùng những trạng thái này để xuất ra typing spinner, cũng như xóa tin nhắn của người dùng. Cũng cần phải tìm hiểu về cách mở tab mới trên trình duyệt hay sử dụng các phương thức của vuejs để xử lí trang web khác nhau khi người dùng truy cập chatbot bằng điện thoại hay máy tính. Vì lí do thời gian hạn chế nên chúng tôi chưa thể cập nhật ngay được, sau này chúng tôi sẽ nâng cấp để phần giao diện thân thiện hơn với người dùng.

### 5.3.5 Sự hài lòng của người dùng sau khi sử dụng chatbot



Hình 4. 43 Biểu đồ thống kê đánh giá của người dùng về sự hài lòng thoải mái khi nói chuyện cùng chatbot trên thang điểm 5.

Tiêu chí này chúng tôi xem như là một phần tổng kết cảm nghĩ của người dùng sau khi sử dụng chatbot, kết quả có thể tạm coi ở mức trung bình khá. Cuối cùng bên cạnh những nhận xét không tốt mà đã nêu ở phần trên thì cũng có một số nhận xét khá tích cực như:

* *“Khá ổn”*
* *“Trả lời tương đối nhanh và kết quả tạm ổn.”*
* *“Làm tốt rồi, sử dụng cũng khá vui.”*

Tuy kết quả đánh giá ở mức khá nhưng chúng tôi cho rằng đây cũng là một bước khởi đầu khá thành công với nhóm. Chương tiếp theo sẽ tổng kết và đánh giá những gì mà nhóm đã học được sau khi thực hiện đề tài, cũng như những khó khăn gặp phải.

# CHƯƠNG 6: TỔNG KẾT, ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN

*Chương này sẽ tập trung trình bày về những kết quả mà chúng tôi đúc kết được qua quá trình thực hiện luận văn, đồng thời đưa ra những đánh giá, nhận xét về những gì nhóm đã làm được trong suốt thời gian qua.*

## 6.1 Kết quả đạt được

Sau thời gian nghiên cứu và thực hiện đề tài, chúng tôi đã gặp không ít khó khăn. Từ việc chuyển đề tài nghiên cứu từ việc tập trung câu hỏi why sang tập trung vào câu hỏi what và how, đến việc không có đủ dữ liệu để làm phải thực hiện thủ công rất nhiều. Với kiến thức nền của chúng tôi là hệ thống thông tin, về quản trị database, chúng tôi phải tiếp xúc với rất nhiều những kiến thức mới mà chúng tôi chưa từng nghĩ là nó có tồn tại như là semantic web hay QAS. Tuy nhiên chính những khó khăn đó đã giúp chúng tôi tiến bộ hơn rất nhiều: chúng tôi được mở rộng tầm nhìn để biết ngoài kia có rất nhiều những người đang nghiên cứu những đề tài to lớn giúp xã hội ngày càng tiến bộ, biết được cách để xây dựng một ứng dụng, từ bước coding đến bước deploy sản phẩm, ngoài ra việc thực hiện đề tài còn giúp chúng tôi học được nhiều những kĩ năng quan trọng khác như: trình bày báo cáo, nghiên cứu tài liệu, các bài báo khoa học, kĩ năng trình bày ý tưởng và làm việc nhóm. Sau đây chúng tôi xin trình bày rõ hơn về những kết quả đã đạt được về lý thuyết và thực hành:

### 6.1.1 Về lý thuyết

* Hiểu được khái niệm và phân loại một hệ thống hỏi đáp (QAS) cũng như biết được ưu khuyết điểm của từng loại.
* Hiểu được khái niệm chatbot, cách vận hành của một chatbot và sự liên quan giữa chatbot với QAS.
* Tổng hợp và so sánh được các nền tảng, công cụ xây dựng chatbot và ưu nhược điểm của từng loại.
* Hiểu được khái niệm về semantic web và tầm quan trọng của semantic web đối với tương lai của internet.
* Hiểu được về web API và làm thế nào để xây dựng một web API hoàn chình.
* Hiểu được lý thuyết về xử lí ngôn ngữ tự nhiên, các thuật toán intent classification và NER.

### 6.1.2 Về thực hành

* Khả năng phân tích và hiểu được các ứng dụng QAS hiện nay từ đó áp dụng vào đề tài.
* Khả năng xây dựng một trang web API service hoàn chỉnh gồm server API ontology, server backend API và giao diện frontend cho người dùng dễ dàng thao tác với chatbot.
* Biết cách thiết kế, xây dựng một ontology. Biết cách thêm dữ liệu và truy vấn ontology với ngôn ngữ SPARQL.
* Biết cách sử dụng RASA như một công cụ để áp dụng các thuật toán xử lí ngôn ngữ tự nhiên nhằm bóc tách ý định người dùng và xây dựng phản hồi phù hợp.
* Biết cách deploy, publish một ứng dụng để bất kì ai cũng có thể trò chuyện với chatbot một cách dễ dàng. Biết cách sử dụng các công cụ hosting trên mạng và server của riêng mình (do bộ môn cấp cho một account truy cập).

## 6.2 Hạn chế

Ngoài những nhược điểm mà bất kì một ứng dụng chatbot hay QAS thuộc miền kiến thức hạn chế và sử dụng database gặp phải thì ứng dụng còn có những hạn chế do bản thân nhóm tác giả sau đây:

* Do thời gian có hạn và không có nhiều kiến thức trong lĩnh vực xử lí ngôn ngữ tự nhiên nên việc chuẩn bị dữ liệu còn thực hiện thủ công chứ chưa thể bóc tách dữ liệu từ nguồn có sẵn và tự động thêm vào database.
* Vì thực hiện thủ công nên dữ liệu chuẩn bị chưa đầy đủ, chỉ ở mức đủ để demo giải pháp, ý tưởng.
* Phần giao diện làm chưa đủ tốt do chưa có kinh nghiệm trong việc thiết kế giao diện, cũng như chưa thật sự tập trung nhiều thời gian cho phần này.

## 6.3 Hướng phát triển

Ứng dụng của đề tài mới chỉ là bước sơ khởi, bắt đầu. Để chatbot có thể thông minh hơn trong tương lại thì cần phải có rất nhiều những nghiên cứu khác nhằm bổ trợ như:

* Nghiên cứu xử lí tài liệu thô thành dữ liệu có cấu trúc để thêm vào dữ liệu ontology nhằm tiết kiệm thời gian xây dựng KBS cũng như khiến chatbot có một nguồn KBS dồi dào hơn.
* Cần xây dựng chatbot có thể hiểu được không chỉ là câu hỏi what, how mà còn các câu hỏi phức tạp khác như causal question hay hypothetical question.
* Bên cạnh đó, sẽ thêm thành phần đánh giá độ phải hồi của người dùng là tích cực hay tiêu cực để tính chính xác cho câu trả lời trong database.
* Nghiên cứu giúp chatbot có khả năng tự đánh giá phản hồi của người dùng và tự học, tức là xử dụng chính những đoạn hội thoại cũ làm dữ liệu train khiến chatbot ngày càng thông minh hơn.
* Nghiên cứu phát triển giúp chatbot có thể nhận diện hình ảnh lỗi do người dùng đưa vào và đưa ra phản hồi thích hợp. Áp dụng các kĩ thuật xử lý ảnh của deeplearning vào để xử lý bài toán này.
* Xa hơn nữa, có thể nghiên cứu về việc chatbot không chỉ biết trả lời câu hỏi người dùng mà còn có thể lấy thông tin của người dùng, theo dõi người dùng đang thực hiện công việc gì trên ứng dụng kĩ thuật được hỏi từ đó đưa ra giải pháp hữu ích nhất.

# DANH MỤC THAM KHẢO

|  |  |
| --- | --- |
| [1] | I. V. G.-D. A. G. C. A. S. C. S. C. A. a. B. Y. Serban, “Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus,” 2016. |
| [2] | “Information retrieval,” [Trực tuyến]. Available: https://en.wikipedia.org/wiki/Information\_retrieval. |
| [3] | “Natural language processing,” [Trực tuyến]. Available: https://en.wikipedia.org/wiki/Natural\_language\_processing. |
| [4] | H. Heidenreich, “Towards Datascience,” [Trực tuyến]. Available: https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8. |
| [5] | K. S. C. N. M. Ajitkumar M. Pundge, Question Answering System, Approaches and Techniques: A Review. |
| [6] | S. K. J. Amit Mishra, A survey on question answering systems with classification. |
| [7] | “Chatbot,” [Trực tuyến]. Available: https://en.wikipedia.org/wiki/Chatbot. |
| [8] | M. Sanjeevi, “Chapter 11: ChatBots to Question & Answer systems.” [Trực tuyến]. Available: https://medium.com/deep-math-machine-learning-ai/chapter-11-chatbots-to-question-answer-systems-e06c648ac22a. |
| [9] | A. Mohite, “What are the differences between a question answering system and a chatbot?” [Trực tuyến]. Available: https://www.quora.com/What-are-the-differences-between-a-question-answering-system-and-a-chatbot. |
| [10] | T. L. D. W. C. T. B. D. S. K. M. C. V. S. K. Andrei Dulceanu, PhotoshopQuiA: A Corpus of Non-Factoid Questions and Answers for Why-Question Answering. |
| [11] | T. L. D. W. C. T. B. D. S. K. M. C. V. S. K. Andrei Dulceanu, “PhotoshopQuiA: A Corpus of Non-Factoid Questions and Answers for Why-Question Answering,” Adobe Research, California, United States. |
| [12] | “AdobePhotoshop Guide,” [Trực tuyến]. Available: https://helpx.adobe.com/photoshop/. |
| [13] | “Mạng ngữ nghĩa,” [Trực tuyến]. Available: https://vi.wikipedia.org/wiki/M%E1%BA%A1ng\_ng%E1%BB%AF\_ngh%C4%A9a. |
| [14] | “http://ontology-study.blogspot.com/2011/09/ontology-la-gi.html,” [Trực tuyến]. |
| [15] | “github” [Trực tuyến]. Available: https://github.com/hyperopt/hyperopt/wiki/FMin#21-parameter-expressions. |
| [16] | PaulBrown, “blog,” [Trực tuyến]. Available: http://www.paulbrownmagic.com/blog/flask\_foaf.html. |
| [17] | “VueJS,” [Trực tuyến]. Available: https://vi.wikipedia.org/wiki/Vue.js. |
| [18] | Guide. [Trực tuyến]. Available: https://vi.vuejs.org/v2/guide/. |
| [19] | D. Garijo, “How to (properly) publish a vocabulary or ontology in the web,” [Trực tuyến]. Available: https://figshare.com/articles/How\_to\_properly\_publish\_a\_vocabulary\_or\_ontology\_in\_the\_web/881824. |
| [20] | “Best Practice Recipes for Publishing RDF Vocabularies,” [Trực tuyến]. Available: https://www.w3.org/TR/swbp-vocab-pub |
| [21] | “Top 14 Web Hosting miễn phí tốt nhất, không chứa quảng cáo 2018,” [Trực tuyến]. Available: http://thuthuat.taimienphi.vn/top-14-web-hosting-mien-phi-tot-nhat-2018-33470n.aspx. |
| [22] | [Trực tuyến]. Available: https://archive.org/services/purl/. |
| [23] | “5 ví dụ về Rewrite URL dùng htaccess thông dụng,” [Trực tuyến]. Available: https://vietcoding.com/5-vi-du-ve-rewrite-url-dung-htaccess-thong-dung/. |
| [24] | “Các thuật ngữ trong xử lí ngôn ngữ tự nhiên,” [Trực tuyến]. Available: https://techtalk.vn/cac-thuat-ngu-trong-xu-ly-ngon-ngu-tu-nhien.html. |
| [25] | A. S. m. Drift, The 2018 State of Chatbots Report. |