ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

****

XỬ LÝ NGÔN NGỮ TỰ NHIÊN

BÁO CÁO BÀI TẬP LỚN

Nhóm 5:

Tìm hiểu Rasa chatbot

Giảng viên: TS. Trần Hồng Việt

Sinh Viên: Đỗ Quang Phong – 15021797

Nguyễn Tuấn Anh – 15020971

Lê Bá Công - 17020022

Mục lục

I. Giới thiệu

1. Giới thiệu về Chatbot

Chatbot là một chương trình máy tính tương tác với người dùng bằng ngôn ngữ tự nhiên dưới giao diện đơn giản, âm thanh hoặc dưới dạng tin nhắn.

Chatbot hay còn được gọi với cái tên khác như Conversational Assistant, Conversational Agent, Dialog Systems là một chương trình phần mềm có khả năng nói chuyện với người sử dụng ngôn ngữ tự nhiên. Để làm được điều đó, chatbot phải hiểu nội dung lời nói của người dùng và cung cấp phản hồi liên quan tương ứng.

Khi nhắc đến Chatbot, dường như nó đã trở thành một danh từ nổi tiếng mà mỗi người đều có những hình dung cụ thể của riêng mình dành cho nó.

Theo một thống kê thực tế từ đầu năm 2018 của Hubspot, số lượng hàng hóa bán ra cho người dùng trên toàn thế giới thông qua chatbot chiếm tới hơn 47% và con số này cho đến nay chắc chắn đã lớn hơn rất nhiều.

Lợi ích và ưu điểm của hệ thống này là khá rõ ràng, cho đến bây giờ thì các hệ thống Chatbot ngày càng trở nên linh động, hoàn hảo hơn , có thể thấy rõ nhất là đối với những chatbot được tạo ra bởi các tập đoàn công nghệ lớn như Google với Google Assistant hay Apple với Siri, Cortana của Microsoft...

2. Phân loại các hệ thống ChatBot

Retrieval-based model: Những hệ thống này không thể tạo ra bất kỳ từ mới, chúng chỉ lấy một số câu phản hồi (câu trả lời) từ một tập có sẵn. Nói cách khác, các câu trả lời được định nghĩa trước trong một kho, do vậy chỉ cần một thuật toán có thể tìm kiếm để chọn ra câu trả lời thích hợp (vd: đang hỏi về sản phẩm hay đang đánh giá sản phẩm).
Generative model: Mô hình sẽ tự tạo ra câu trả lời dựa trên kỹ thuật machine translation, nhưng thay vì chuyển từ ngôn ngữ này sang ngôn ngữ kia, chúng chuyển từ câu thoại này sang câu thoại kia. Để huấn luyện được những mô hình như vậy sẽ khó hơn, cần nhiều dữ liệu hơn.

Long conversations: khó hơn do câu trả lời có thể là câu phức, với nhiều thông tin, mục đích khác nhau.
Short conversations: dễ hơn do câu hỏi là câu đơn.

Open domain: Khó hơn do người dùng tạo cuộc hội thoại ở bất kỳ lĩnh vực nào.
Closed domain: Phổ biến hơn trong doanh nghiệp, khi mà hệ thống nhận một nhiệm vụ cụ thể, ví dụ như bán hàng hay hỏi đáp y tế.

Chatbot là một hệ thống dựa vào các kỹ thuật về xử lý ngôn ngữ tự nhiên, vậy nên giống với hầu hết các bài toán về xử lý ngôn ngữ khác, các hệ thống Chatbot có thể được chia làm 2 loại

1. Hệ thống Chatbot dựa vào các quy luật, thói quen trong ngôn ngữ của người dùng (Rule-based chatbots)
2. Hệ thống Chatbot xây dựng trên một kho dữ liệu hội thoại cho trước (Corpus-based chatbots) - Kho văn bản này có thể được thu thập bằng lượng lớn dữ liệu từ các cuộc nói chuyện của người dùng, sử dụng các phương pháp trích xuất thông tin (Infomation Retrieval) hoặc các phương pháp máy học để tạo ra câu trả lời dựa vào ngữ cảnh trò chuyện với người dùng.

2.1 Rule-Based Chatbots

Hiểu một cách đơn giản, Rule-based chatbot sẽ trả lời người dùng dựa hoàn toàn vào các thói quen sử dụng ngôn ngữ của chúng ta mà không cần xử lý việc ghi nhớ thông tin trước đó. Tuy rằng trong ngôn ngữ nói hàng ngày của chúng ta, mỗi người đều sẽ có những cách diễn đạt, cách sử dụng từ ngữ riêng để tạo ra các câu hội thoại của bản thân, nhưng thói quen sử dụng ngôn ngữ con người có xu hướng lặp lại khá nhiều.Những thói quen này sẽ được lập trình viên khai thác để xây dựng Chatbot với độ phức tạp chương trình tùy vào mong muốn của

người tạo ra. Ví dụ có thể kể đến là nếu trong câu hỏi của người có nhắc tới "thời tiết", "mưa nắng" hay "nhiệt độ" thì khả năng cao người dùng đang muốn hỏi về tình hình thời tiết..

Hệ thống Chatbot thành công và nổi tiếng nhất thuộc loại này là ELIZA Chatbot . ELIZA được tạo ra từ năm 1966, và được coi như một bước tiến quan trọng trong lịch sử của Chatbot nói chung và trí tuệ nhân tạo nói riêng. ELIZA mô phỏng một bác sĩ tâm lý với công việc lắng nghe và hỏi kỹ càng về những câu chuyện của bệnh nhân, giúp cho họ có thể dần dần nói ra những điều mà họ không thể chia sẻ được với ai.

2.2 Corpus-based chatbots

Thay vì việc sử dụng những luật xây dựng bằng tay như rule-based chatbots, hệ thống chatbot dựa trên kho dữ liệu hội thoại trong thực tế của người với để tìm ra được những phản hồi phù hợp cho hoàn cảnh của mình. Những dữ liệu hội thoại này có thể được thu thập trực tiếp trên một số platform trò chuyện hoặc lấy ra từ các lời thoại của nhân vật trong các bộ phim với nhau.

Corpus-based hiện nay thường sẽ có 2 kiểu hoạt động chính: Trích xuất thông tin cần thiết (Information Retrieval) và áp dụng các bài toán Deep learning theo dạng sequence to sequence (tương tự như Dịch tự động)

Information Retrieval based chatbots (IR-based)

Hệ thống nổi tiếng nhất làm theo mô hình này là Simsimi. Ứng dụng từng một thời được rất rất nhiều sự quan tâm của cộng đồng mạng tại Việt Nam. Cơ chế hoạt động của một hệ thống IR-based có bước đầu tiên là tìm trong cơ sở dữ liệu hội thoại, một câu nói có độ tương tự cao nhất với câu nói hiện tại của user. Từ đó có 2 cách để đưa ra câu trả lời:

1. Dùng chính câu nói đó

response = most_similar (q)

response=most_similar(*q*)

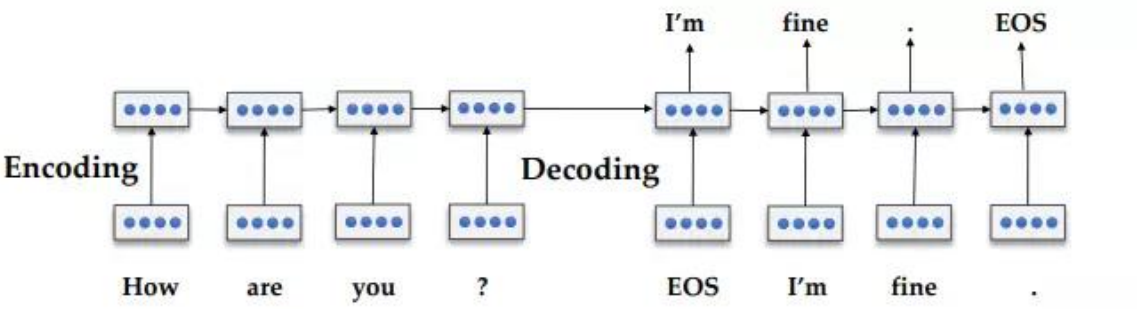
2. Dùng câu trả lời cho câu nói đó

response = process (most_similar (q))

response=process(*most_similar*(*q*))

Trong thực tế, trong mỗi bài toán, chúng ta lại có cách tính độ tương tự câu nói khác nhau, nhưng cách thông thường nhất vẫn là sử dụng thuật toán TF-IDF để chuyển về dạng vector số thực và tính toán độ tương tự dựa vào khoảng cách cosine.

Sequence to sequence chatbots



Sequence to sequence là một bài toán đang được giải quyết một cách khá mạnh mẽ bởi các mạng Deep Learning bây giờ. Với đầu vào là 1 câu, dựa trên tập dữ liệu của chúng ta, chúng ta sẽ có thể sinh ra câu trả lời dựa vào Deep Learning. Bài toán này gần tương tự như bài toán dịch tự động (Auto Translation), chỉ khác ở chỗ, trong trường hợp này của chatbots, ngôn ngữ nguồn và ngôn ngữ đích sẽ cùng là một ngôn ngữ.

Một số từ khóa các bạn có thể tìm hiểu theo hướng giải quyết này đó là: Recurrent Neural Network, LSTM, GRU, TCN và Transformer

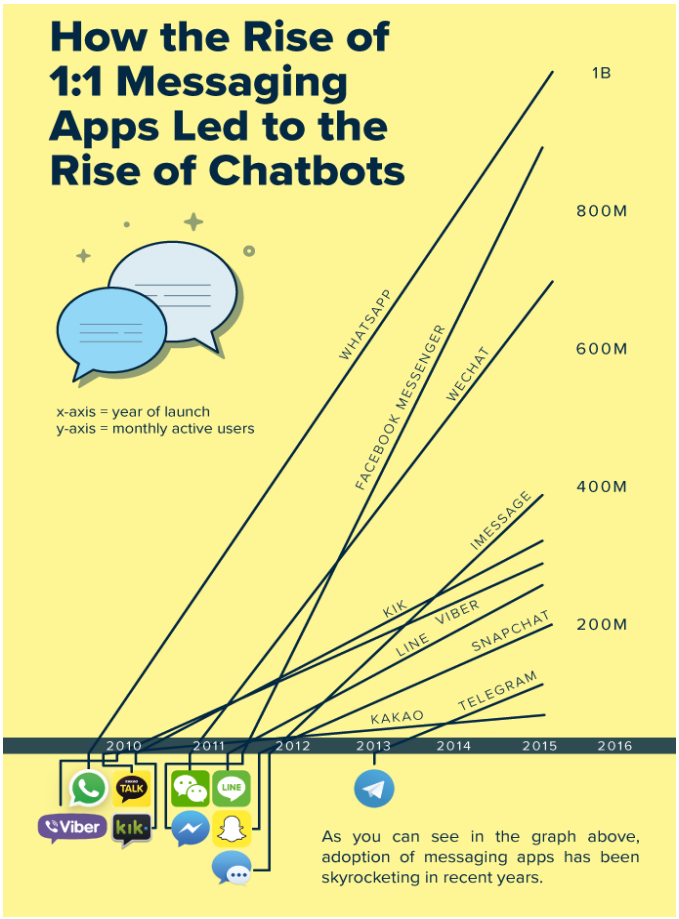
2.3 Frame Based Agents - Chatbot cho một nhiệm vụ cụ thể

Ngoài 2 kiểu Chatbot là Rules-based và Corpus-based mình vừa giới thiệu ở trên, mình cũng muốn nói tới 1 loại Chatbot khác nữa. Đa số các Chatbot mà mọi người tiếp xúc (ngoại trừ 1 số chatbot assistant

của các hãng công nghệ lớn) đều thuộc loại này - Frame Based Agents. Dịch nôm na là Chatbot Đại Lý. Nó mô phỏng nhiệm vụ của một đại lý, sẽ thu thập các yêu cầu của Khách hàng và gửi yêu cầu đó đi khi đã thu thập đầy đủ được thông tin.
Với loại Chatbot này, luồng thực hiện của Chatbot là khá rõ ràng chứ không "mông lung" như các loại trên. Thiếu thông tin nào Chatbot sẽ phải hỏi thêm về thông tin đó, tuy nhiên thách thức đặt ra ở việc lấy thông tin sao cho chính xác. Ví dụ trong trường hợp hỏi các thông tin ngày, nhiều người sẽ có thể kèm luôn cả giờ trong câu trả lời như: "Anh book từ 4-5h 15-9-2019". Vậy làm sao để lấy được ngày và giờ một cách chính xác? Mỗi người sẽ có một cách trả lời thông tin riêng, điều này cần thời gian để chúng ta có thể handle được toàn bộ các tình huống.

3. **Tại sao chatbot trở thành hot trend?**

   Thời đại bùng nổ của tin nhắn

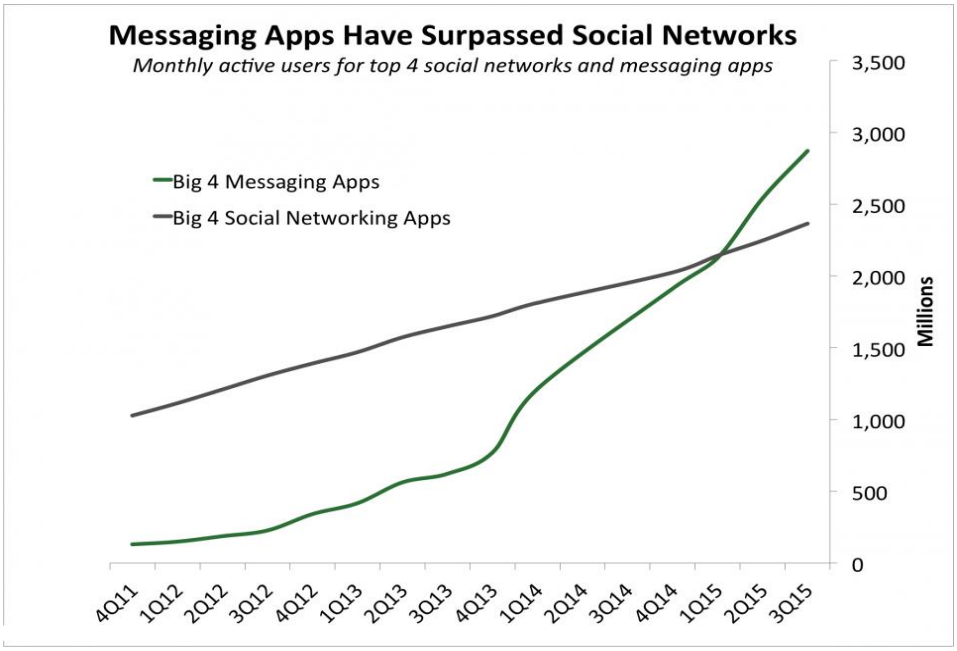


- 28,2 tỷ tin nhắn di động đã được gửi trong năm 2017, gấp đôi so với năm 2012

- 98% tin nhắn sẽ được đọc, với email tỷ lệ là 22%.

- 6 trong top các ứng dụng được cài đặt là ứng dụng nhắn tin.

- Tỷ lệ gỡ bỏ các ứng dụng nhắn tin chỉ bằng một nửa so với các ứng dụng khác.



- Số lượng người dùng hàng tháng của các ứng dụng nhắn tin ngày càng tăng, vượt qua số lượng người sử dụng các ứng dụng social Network

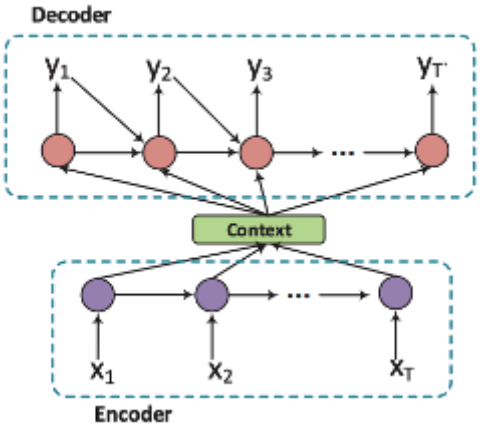
Sự phát triển của NLP và NLU

AI và ML trở thành một trong những công nghệ được đầu tư mạnh nhất trong vài năm trở lại đây giúp cho Natural Language Processing (NLP), Natural Language Understanding (NLU) sử dụng các kỹ thuật, công nghệ mới trở nên đơn giản và chính xác hơn rất nhiều. Chúng ta có thể tạo ra chatbot bằng câu lệnh if-else, tuy nhiên việc tạo ra những dumb-bot như vậy không đem lại nhiều giá trị. NLP bây giờ trở thành yếu tố quyết định độ thông minh của chatbot.

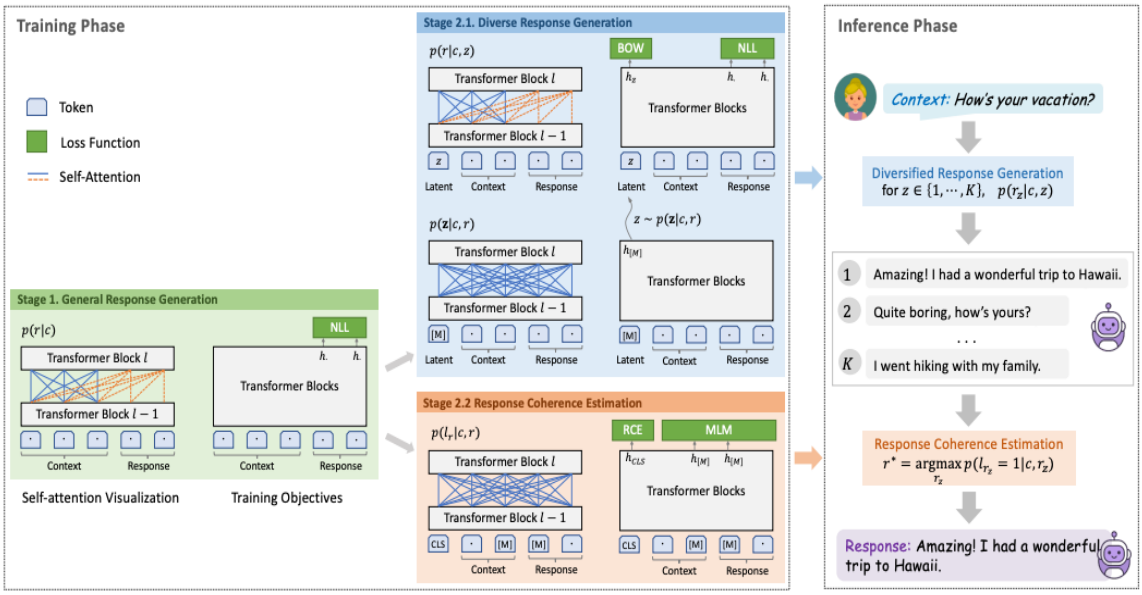
Các mô hình mới, cải tiến

Seq2seq: Hầu hết các mô hình chatbot hiện nay đều sử dụng phương pháp mạng nơ-ron hồi quy (RNN) do thông tin phía trước được giữ lại và truyền đi theo chuỗi. Hai biến thể của mạng hồi quy được dùng thường xuyên là LSTM và GRU. Về tổng thể, các mô hình cho chatbot có thể chia làm hai phần: encoder và decoder. Với phần encoder, thông tin được biểu diễn dưới dạng vector rồi cho đi qua mạng, đầu ra của mạng encoder là một vector thể hiện bối cảnh của cuộc hội thoại.

Vector bối cảnh này được đưa tiếp đến mạng decoder để sinh ra câu đích.



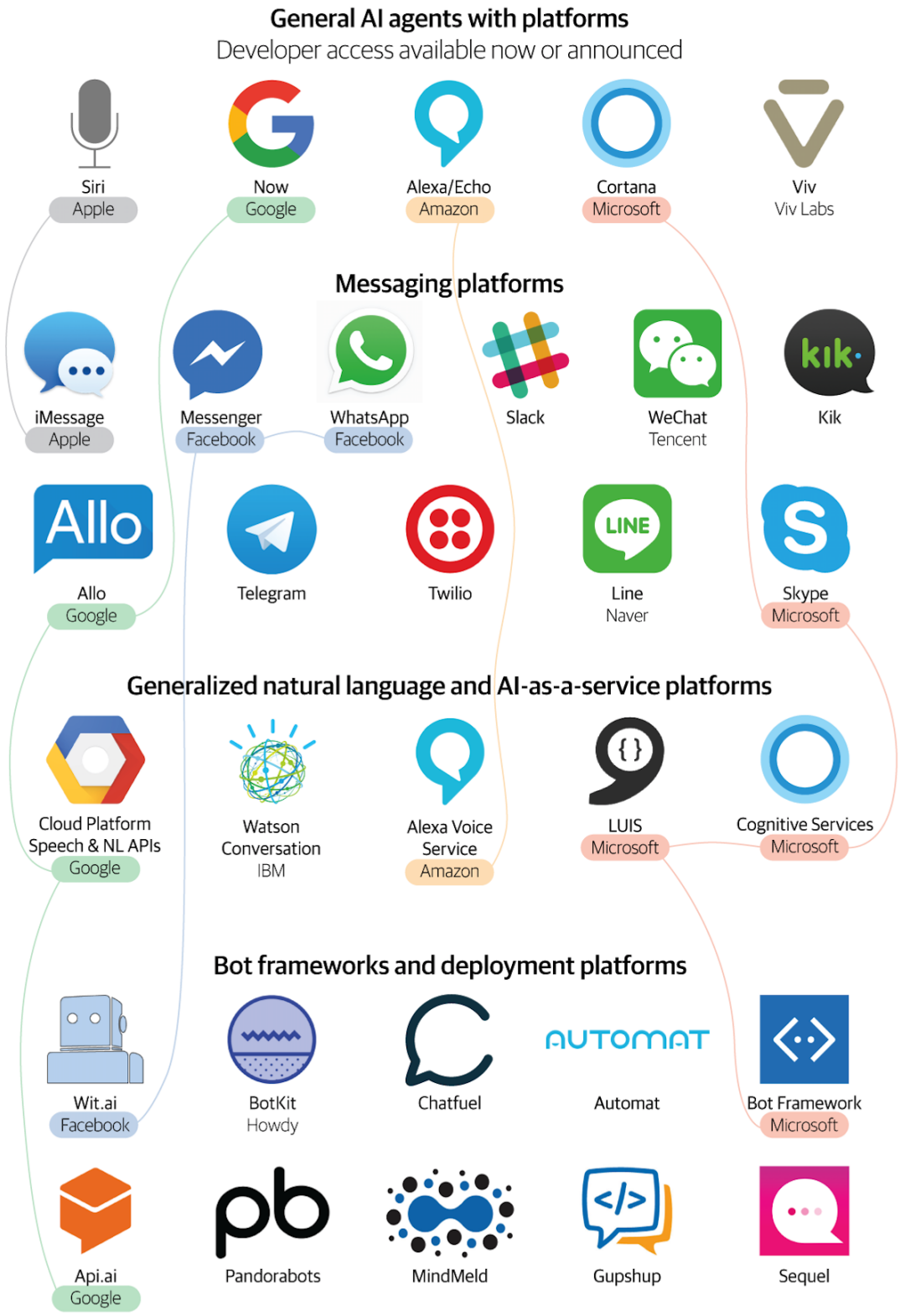
PLATO-2: Đây là một phương pháp xây dựng open-domain chatbot. Với hai bước chính trong quá trình học: sinh thô và sinh mịn. Dưới đây là biểu đồ quá trình huấn luyện và suy luận của mô hình PLATO-2 [1].



Hệ sinh thái, platform

Tất cả các top messenger đều có platform của riêng mình phục vụ cho việc xây dựng chatbot, gần như chúng giống nhau nên việc code một lần sử dụng cho nhiều platform là điều có thể.

Hệ sinh thái, các projects mã nguồn mở để xây dựng chatbot ngày càng hoàn thiện và đầy đủ, các bạn có thể sử dụng miễn phí (hầu hết) trên môi trường develop



Các ngôn ngữ cũng có những thư viện riêng dành cho chatbot, các platform có công cụ phát triển (sdk) đầy đủ cho các service của mình

4. Các thử thách thường gặp

Incorporating Context

Để tạo ra câu trả lời có ý nghĩa, hệ thống cần kết hợp cả ngôn ngữ học và ngữ cảnh vật lý (*linguistic context* and *physical context),* Trong các cuộc hội thoại dài con người sẽ lưu lại những gì đã nói và những thông tin đã trao đổi. Đây là một ví dụ về ngôn ngữ học. Hầu hết các cách tiếp cận thường thấy là chuyển cuộc hội thoại thành một vector, nhưng đối với một cuộc thoại dài là một thử thách.

Coherent Personality

Khi tạo ra các câu trả lời, máy cần phải trả lời một cách thống nhất với các câu đầu vào giống nhau. Ví dụ, bạn muốn lấy cùng một câu trả lời cho "How old are you?" và "What is your age?". Điều này nghe thì đơn giản, nhưng kết hợp với hiểu biết cố định hay cá nhân và trong model là một vấn đề cần nghiên cứu nhiều. Rất nhiều hệ thống học tạo các câu trả lời đúng về ngữ nghĩa, nhưng chúng không được train từ cùng một nguồn thống nhất.

Dữ liệu huấn luyện

Bộ dữ liệu huấn luyện cho chatbot thường là các corpus chưa cuộc hội thoại giữa các nhân vật trong phim. Ví dụ như bộ *Cornel Movie-Dialog Corpus* hoặc *OpenSubtitles Corpus.* Hoặc từ các cuộc hội thoại cá nhân (*PERSONA-CHAT, Blended skill Talk, FinChat*), từ trao đổi trên diễn đàn (Reddit)

Đánh giá model

Có nhiều cách để đánh giá cuộc hội thoại của máy bởi các thang đo hoặc không, nó có hoàn thành nhiệm vụ, ví dụ giải quyết vấn đề hỗ trợ khách hàng, trong một cuộc hội thoại. Việc đánh giá cuộc hội thoại là đắt đỏ bởi vì cần ý kiến đánh giá của con người. Thỉnh thoảng không có một mục đích tốt được định nghĩa trước như trong trường hợp với open-domain models. Các thang đo thông thường như BLEU cái mà được sử dụng trong machine translation và được dựa trên text matching là không phù hợp bởi vì độ hợp lý của câu trả lời có thể chứa các từ hay cụm từ khác nhau.

Việc đánh giá một hệ thống chatbot có đủ ‘thông minh’ hay không có thể được thực hiện bởi người dùng. Các công ty có thể thuê người

dùng đánh giá hệ thống chatbot của họ theo thang điểm, để từ đó cải thiện thêm. Nếu muốn đánh giá tự động mô hình trên trên các metrics cụ thể, ta có thể chọn độ chính xác Bleu hoặc Perplexity.

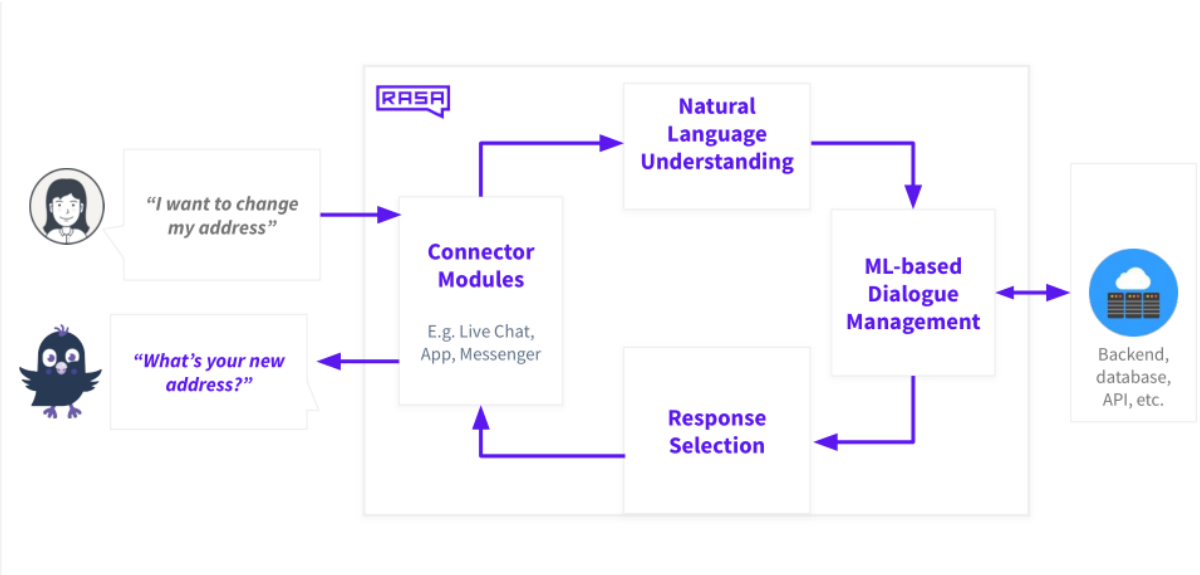
Phán đoán cảm xúc của người dùng

Một chatbot vẫn hoàn thành tốt nhiệm vụ của nó mà không cần quan tâm đến cảm xúc của người dùng. Tuy nhiên việc áp dụng phân tích cảm xúc người dùng giúp chatbot có thể giao tiếp một cách mềm dẻo hơn, tạo cảm giác thân thiện hơn với người sử dụng

II. Rasa Chatbot

1. Tổng quan về Rasa

Rasa là một open-source Machine Learning Framework áp dụng cho việc tự động giao tiếp bằng văn bản hoặc giọng nói (chatbot).



Mô hình cách thức Rasa hoạt động

Nếu các bạn là người bắt đầu muốn nghiên cứu về chatbot, hay chỉ đơn giản là nảy ra một ý tưởng xây dựng một chú "bot" thú vị có thể "chat", hoặc cập nhật tin tức, hoặc làm một tác vụ gì đó phức tạp thì Rasa chính là một cách tiếp cận với nhiều ưu điểm:

- Rasa thực sự dễ tiếp cận cho người mới bắt đầu, hầu hết công việc của người sử dụng là tập trung xây dựng nội dung kịch bản,

khả năng của chatbot chứ không cần thiết phải quan tâm đến công nghệ xây dựng.

- Rasa hoạt động khá tốt và mạnh mẽ, đặc biệt trong vấn đề xác định ý định người dùng (intent) và đối tượng được nhắc đến trong câu (entity) dù dữ liệu bạn thu thập và cung cấp cho rasa là vô cùng ít
- Mã nguồn của Rasa là mã nguồn mở, do đó Rasa giúp bạn biết chính xác được bạn đang làm gì với chatbot của mình, thậm chí có thể custom chatbot theo ý thích của bản thân,

Cấu trúc cơ bản của 1 project Rasa:

actions.py	code for your custom actions
config.yml	configuration of your NLU and dialogue models
credentials.yml	details for connecting to other services
data/**nlu.md**	your NLU training data
data/**stories.md**	your stories
domain.yml	your assistant's domain
endpoints.yml	details for connecting to channels like fb messenger

- **actions.py**: Nơi bạn sẽ code tất cả mọi hành động tùy chỉnh mà bạn muốn bot làm
- **config.yml**: Nơi bạn cấu hình các thông tin liên quan tới mô hình NLU và Core, cách mà chúng hoạt động.
- **credentials.yml**: Thông tin chi tiết về cách kết nối chatbot với các dịch vụ như Facebook, Slack, Telegram,...
- **data/nlu.md**: Dữ liệu huấn luyện cho NLU, bao gồm các câu được gán nhãn intent và entities theo định dạng cho trước.
- **data/stories.md**: Dữ liệu huấn luyện cho Rasa core, là các kịch bản mà bạn muốn bot làm theo.
- **domain.yml**: Đây coi như phần khai báo tất cả mọi thứ mà chatbot của bạn sử dụng, bao gồm các intent, entities, actions,...
- **endpoints.yml**: Các endpoints mà bạn muốn chatbot trả ra
- **models/:** Nơi lưu trữ các model bạn đã huấn luyện

2. Rasa NLU

Rasa NLU được thiết kế với mục đích là phân tích những thông tin có trong tin nhắn mà con người gửi đến cho chatbot. Các thông tin bao gồm ý định của người dùng (intent) và các đối tượng, thực thế được nhắc đến cần trích xuất (Entities).

Ví dụ như trong câu:

> Tôi muốn thuê nhà nghỉ ở khu vực Hà Nội

Thì thông tin mà NLU module trả về sẽ là:

```
{
  "intent": "thue_nha",
  "entities": {
    "house_type" : "nhà nghỉ",
    "location" : "Hà Nội"
  }
}
```

Dữ liệu training cho NLU model bạn có thể định nghĩa theo định dạng json hoặc markdown, ở đây chúng ta sẽ sử dụng markdown và cụ tể là file nlu.yml. Dữ liệu huấn luyện sẽ có dạng như sau

```
## intent:greet
- hi
- hello
- xin chào
- chào em

## intent:goodbye
- bye
- tạm biệt em
- tạm biệt

## intent:ask_bot
- Cho anh mua [đôi giầy](product_name)
- Anh muốn mua [đôi tất](product_name)
- Anh muốn mua [đôi vớ](product_name:đôi tất)

## intent:feedback
- Anh nhận được hàng rồi, sản phẩm chất lượng tốt
```

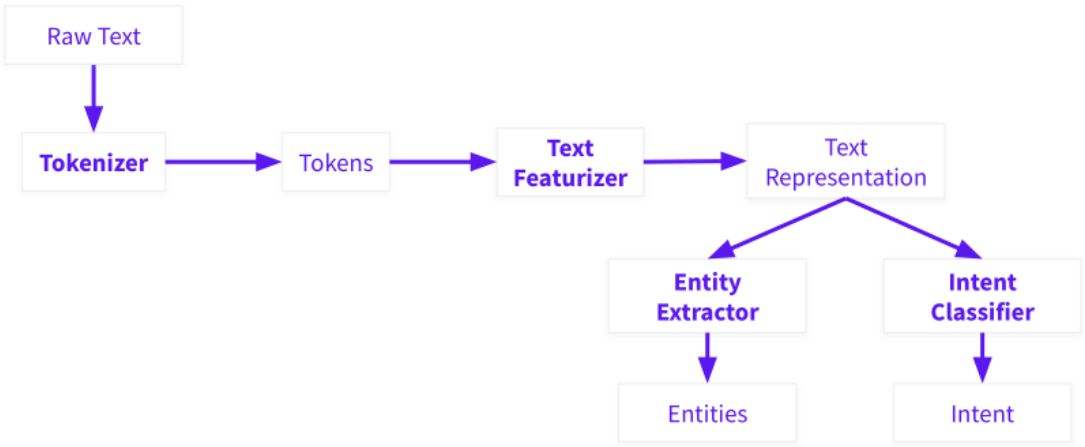
Ở đây ta nghĩa ra 4 intent ứng với mỗi ý định của người dùng mà bot có thể hỗ trợ và 1 entity mà mình gọi là product_name. Trong trường hợp các sản phẩm được gọi với nhiều tên, mình sử dụng cú pháp Entity Synonyms như bạn thấy ở trên để ánh xạ từ *đôi vớ* thành *đôi tất*.

Tiếp theo, chúng ta xem qua các cấu hình của model NLU trong file config.yml. Chi tiết thì các bạn chỉ có thể đọc ở Doc của Rasa là chi tiết và đầy đủ nhất.

```yaml
language: vi
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturiz
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 100
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100
```

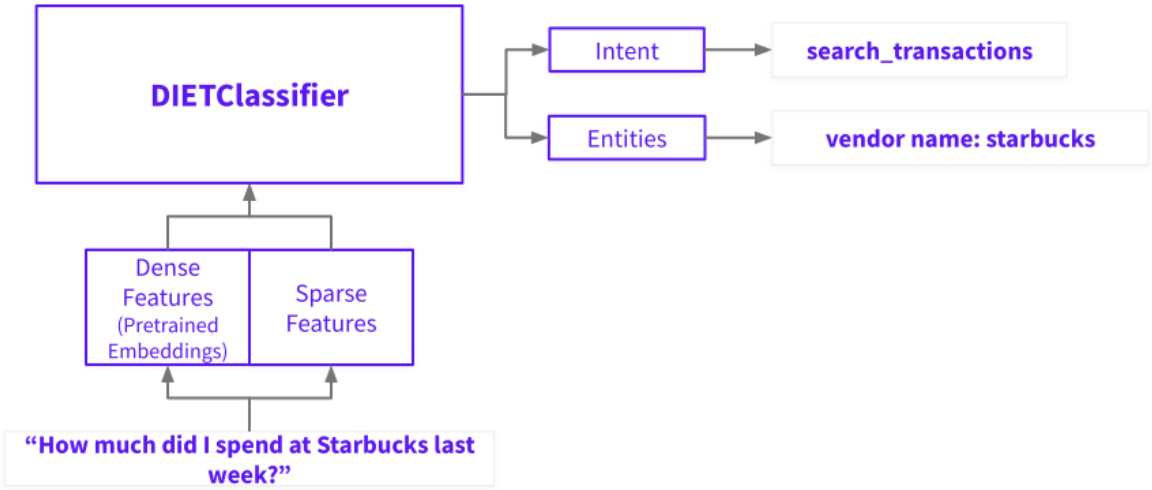
Cấu hình mặc định với pipeline NLU là ***supervised_embeddings***, với pipeline này thì Rasa có thể huấn luyện với bất kỳ ngôn ngữ nào trên đời vì chính xác là bạn đang training lại mọi thứ từ đầu, chỉ phụ thuộc vào dữ liệu huấn luyện của bạn.

Ngoài ra cũng có các pipeline mẫu khác cho bạn lựa chọn như ***pretrained_embeddings_spacy, pretrained_embeddings_convert, MITIE*** hay bạn cũng có thể tùy chỉnh 1 pipeline bất kỳ nào theo ý bạn

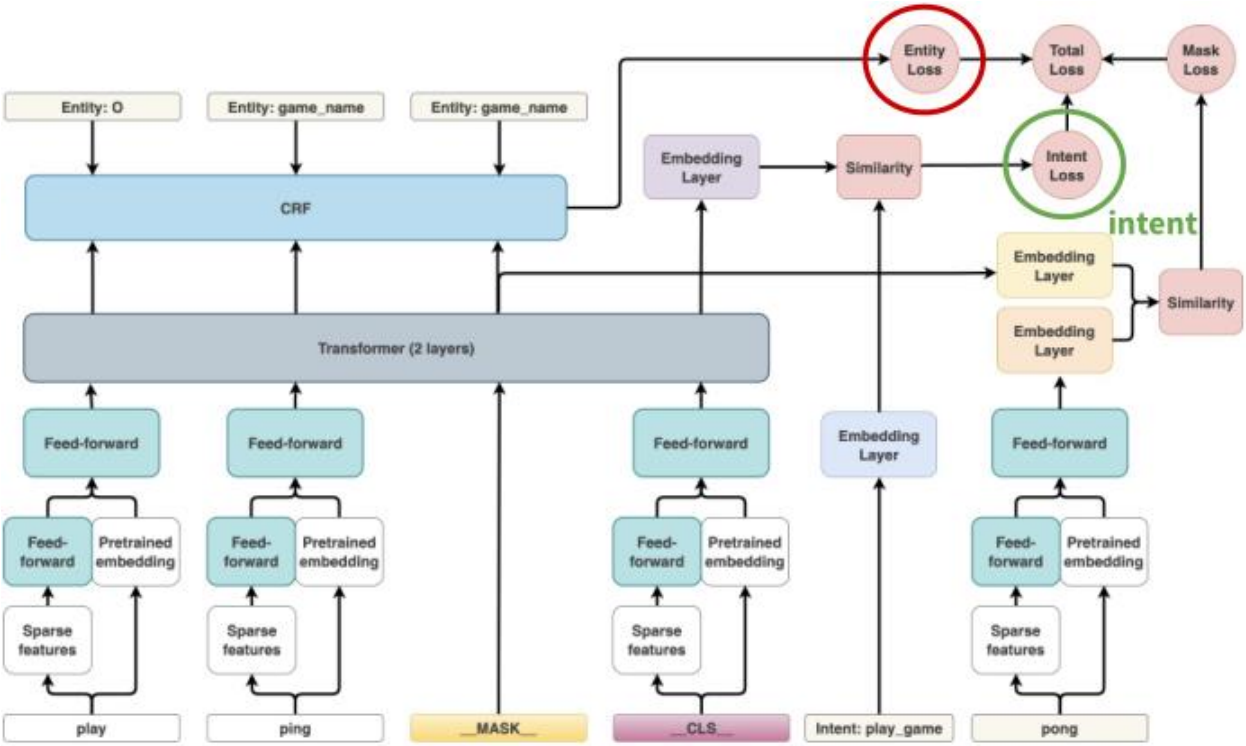


Pipeline định nghĩa một quy trình hoàn chỉnh từ lựa chọn ***Tokenizer***, ***Featurizer***, ***Extractor*** đến ***Classiffer***.

Sau khi ***Tokenizer*** và ***Featurizer***, Rasa sử dụng model ***DIETClassifier*** (Dual Intent Entity Transformer) để xác định intent cũng như trích xuất được entities

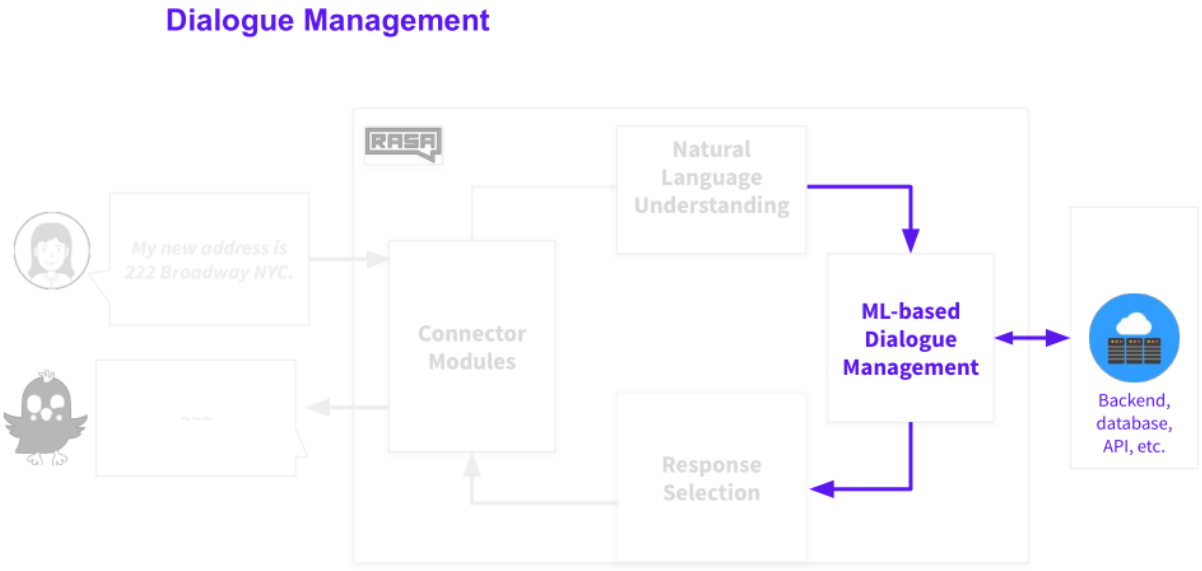


Kiến trúc tổng quát của DIET



Kiến trúc chi tiết của DIET

3. Rasa Core



Sau khi đã phân tích được các thông tin cần thiết có trong tin nhắn của người dùng gửi tới chatbot, việc tiếp theo là dự đoán hành động tiếp theo mà chatbot cần làm để phản hồi lại người dùng. Cái này bao có thể là phản hồi lại tin nhắn hoặc truy vấn database hay bất cứ hành động nào ta có thể định nghĩa. Và Rasa Core được sinh ra để làm nhiệm vụ dự đoán này.

Rasa Core là nơi thực hiện quản lý luồng hội thoại. Dựa vào các intent, entity đã được detect ra ở phần NLU, Rasa Core tiến hành lấy các kết quả này làm đầu vào, rồi quyết định message đầu ra.

Đầu tiên vẫn cần phải cấu hình cho Core trong file *config.yml*

```
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 100
  - name: MappingPolicy
  - name: FallbackPolicy
    nlu_threshold: 0.3
    core_threshold: 0.3
    fallback_action_name: "act_unknown"
```

Chúng ta lần lượt khai báo các *Policy* cần thiết. Ở đây chúng ta cần một số policies như

- **MemoizationPolicy** (quyết định message đầu ra dựa vào thông tin của những đoạn hội thoại trước đó)
- **TEDPolicy** (The Transformer Embedding Dialogue là 1 kiến trúc model để dự đoán action tiếp theo của con bot là gì),
- **MappingPolicy**(quyết định message dựa vào dữ liệu đã mapping)
- Và trong trường hợp, việc tính xác suất đầu ra không thể vượt được ngưỡng mà **FallbackPolicy** đề ra, message trả ra sẽ là một act_unknown kiểu như: "Xin lỗi anh chị ạ, em không hiểu được nội dung anh chị nói ạ"

Tiếp theo, chúng ra cần khai báo các thông tin cần thiết trong file *domain.yml*

```
session_config:
  session_expiration_time: 60.0
  carry_over_slots_to_new_session: true
intents:
- greet
- goodbye
- thankyou
- praise
- decry
- ask_for_lunch
- ask_ability
responses:
  utter_greet:
  - text: "Em chào anh(chị) ạ. \nEm là chatbot được thiết kế để giúp anh chị quyết\
      \ định 'trưa nay ăn gì?' ạ"
  utter_goodbye:
  - text: Hẹn gặp lại anh chị ạ ^^
    image: https://i.imgur.com/nGF1K8f.jpg
  utter_happy:
  - text: Hì hì, anh chị khen quá lời rồi ạ
  utter_sorry:
  - text: Em xin lỗi vì em chưa đủ thông minh ạ =(((
  utter_noworries:
  - text: Em luôn sẵn lòng giúp đỡ anh(chị) bất cứ lúc nào ạ ^^
  utter_show_ability:
  - text: Em có thể trò chuyện với anh(chị), thi thoảng có thể đề xuất anh(chị) nên
      ăn gì trưa nay ạ
  utter_fallback:
  - text: Em xin lỗi, em chưa hiểu ý muốn của anh(chị) ạ. Anh chị có thể nói lại được
      không ạ
actions:
- utter_greet
- utter_happy
- utter_goodbye
- utter_sorry
- utter_noworries
- action_recommend
- utter_show_ability
- utter_fallback
```

Ở đây:

- *intent* là các thông tin đã nếu trong file *nlu*, (có thể có cả entity),
- *action* là phần liệt kê các hành động, message đầu ra mà chúng ra định nghĩa.
- *response* là phần chúng ta định nghĩa các message dạng text, hoặc hình ảnh, ... (các response này thường có dạng utter_{})
- Với các action cần thao tác với database, chúng ta định nghĩa trong file *actions.py*
- Cuối cùng là session_config, là phần cấu hình cho một session như thời gian để restart lại một session, có mang slot từ session cũ sang session mới hay không

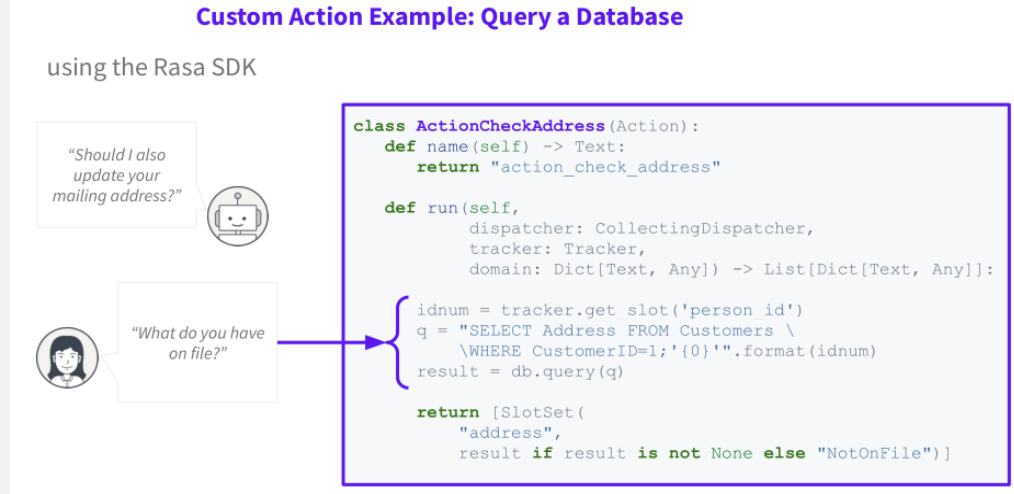
  Sau khi khai báo trong *domain*, chúng ta xây dựng các kịch bản cần thiết cho việc trò chuyện của "bot" trong file *story.yml*

```
## greet
* greet
  - utter_greet

## goodbye
* goodbye
  - utter_goodbye

## thankyou
* thankyou
  - utter_noworries

## ask ability
* ask_ability
  - utter_show_ability

## praise
* praise
  - utter_happy

## decry
* decry
  - utter_sorry

## ask_for_lunch
* ask_for_lunch
  - action_recommend
```

  Việc dự tính trước các luồng hội thoại và xây dựng sẵn một kịch bản sẽ giúp con bot của chúng ta xử lí một cách trơn tru hơn và có vẻ là thông minh hơn. Vậy nên, hãy cố gắng nghĩ đủ sâu các trường hợp mà người dùng có thể hỏi bot, việc này thật sự cần thiết nếu muốn bot đưa ra câu trả lời như mong đợi, không bị hỏi một đằng, trả lời một nẻo.

4. Custom action

Về cơ bản, một chatbot sẽ luôn cần có một database để lưu trữ thông tin: đó có thể là ngân hàng câu hỏi, thông tin về lĩnh vực bot được hỏi, các thao tác với database, ...do đó, chúng ra cũng cần có những xử lý riêng theo từng tác vụ. Rasa hỗ trợ điều đó trong file *actions.py*



Với mỗi action cụ thể, chúng ta xây dựng riêng một class. Class này có đặc điểm sau: chỉ bao gồm 2 method là name() và run()

- name() sẽ trả về tên của action, cái mà chúng ta khai báo trong file domain và file stories
- run() là nơi chúng ta thỏa sức sáng tạo, làm điều ta muốn (cụ thể là code python cho cho cái action này hoạt động thôi).

Ngoài Action, chúng ta còn có các khái niệm khác về Slot, Form Action, Tracker, ... những công cụ hữu ích cho việc custom cho Rasa.

Sau khi đã có file actions.py, muốn action hoạt động được, đừng quên file *endpoints.yml*

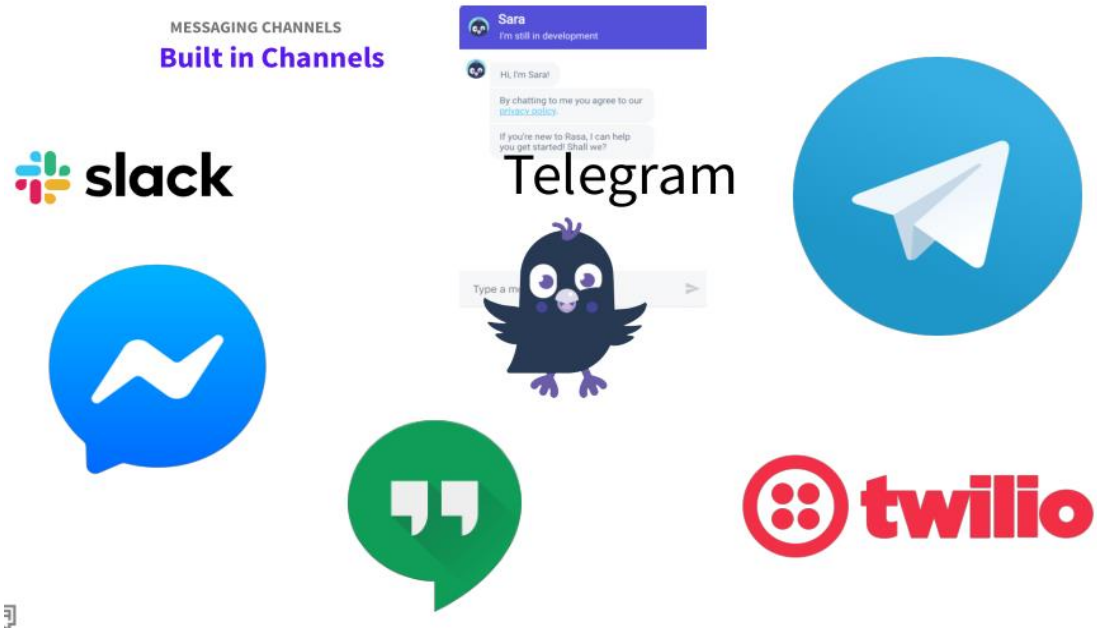
```
action_endpoint:
 url: "http://localhost:5055/webhook"
```

5. Kết nối Rasa với các nền tảng nhắn tin khác

Bên cạnh việc chat trên Rasa X, Rasa có thể hỗ trợ kết nối đến rất nhiều nền tảng khác nhau. Mở file credentials.yml :

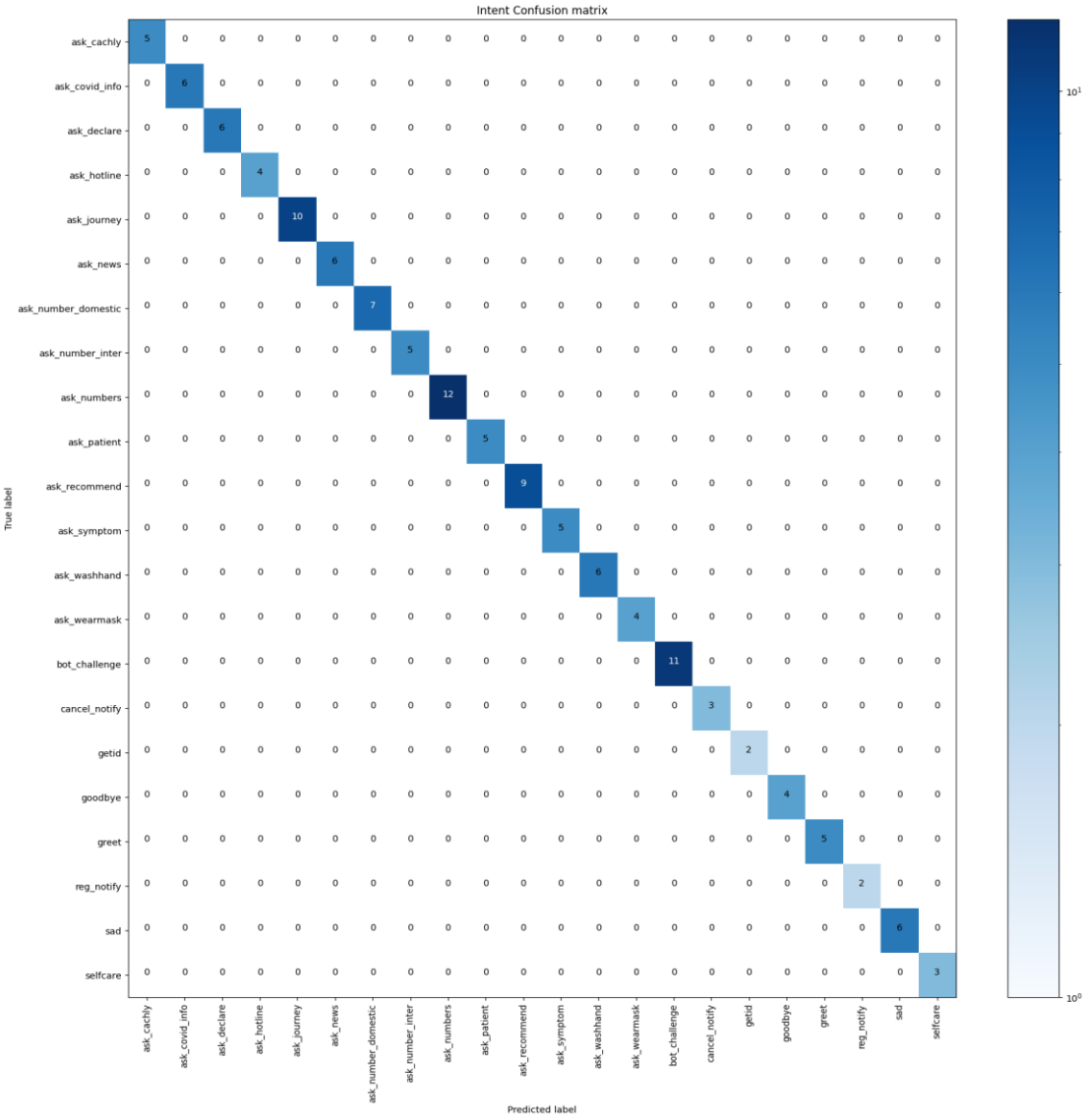
```
rest:
#  # you don't need to provide anything here - this channel doesn't
#  # require any credentials


#facebook:
#  verify: "<verify>"
#  secret: "<your secret>"
#  page-access-token: "<your page access token>"

#slack:
#  slack_token: "<your slack token>"
#  slack_channel: "<the slack channel>"

#socketio:
#  user_message_evt: <event name for user message>
#  bot_message_evt: <event name for but messages>
#  session_persistence: <true/false>

#mattermost:
#  url: "https://<mattermost instance>/api/v4"
#  team: "<mattermost team>"
#  user: "<bot username>"
#  pw: "<bot token>"
#  webhook_url: "<callback URL>"

rasa:
  url: "http://localhost:5002/api"
```

Các bạn có thể thấy một vài kết nối của Rasa đến facebook, slack, socketio,.... Công việc của chúng ta chỉ cần uncomment phần code đó, sau đó điền thông tin về chatbot của mình vào. Với những nền tảng chưa được hỗ trợ mặc định, chúng ta cũng có thể custom nó theo ý của bản thân

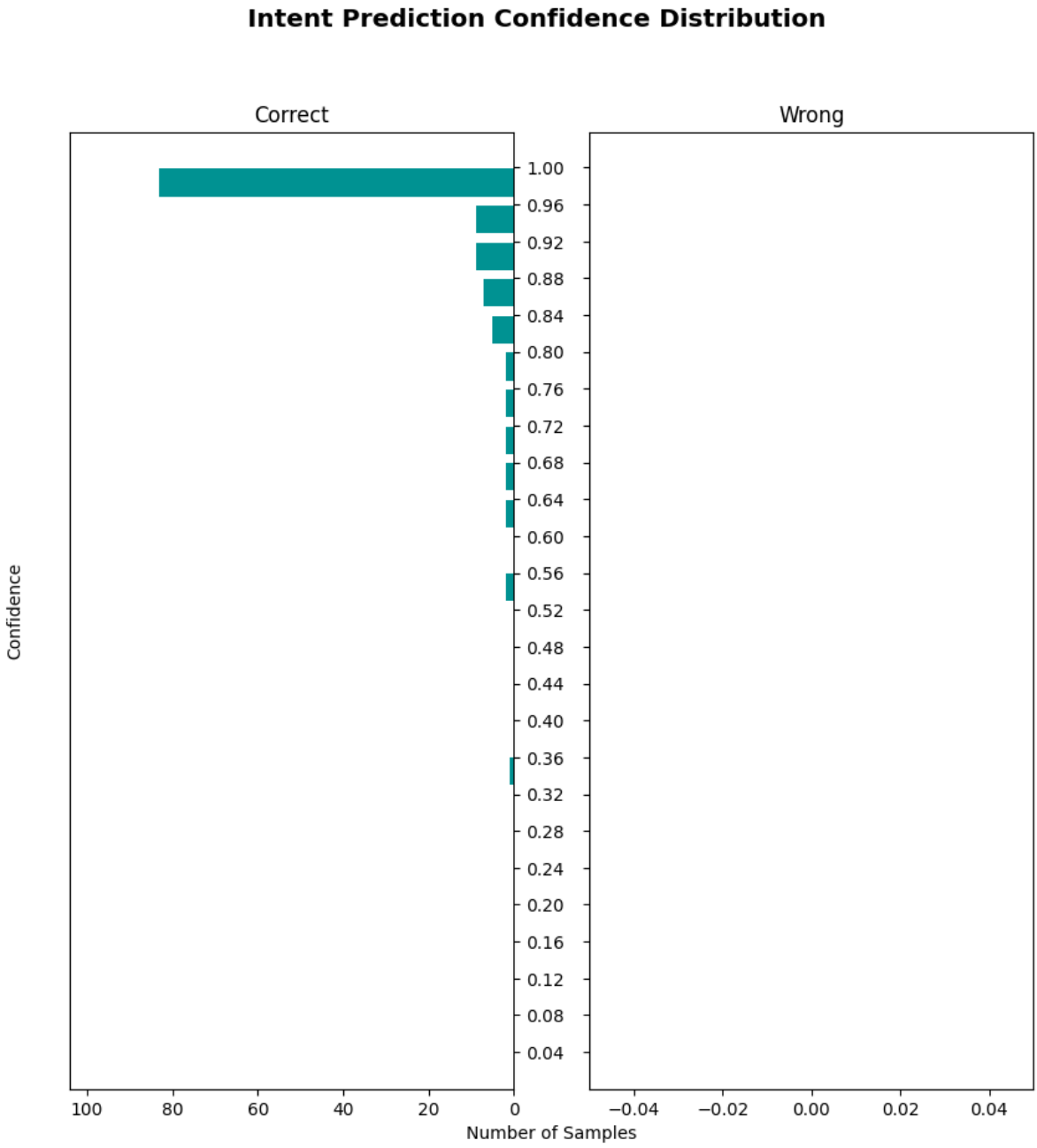


III. Đánh giá mô hình và cải tiến

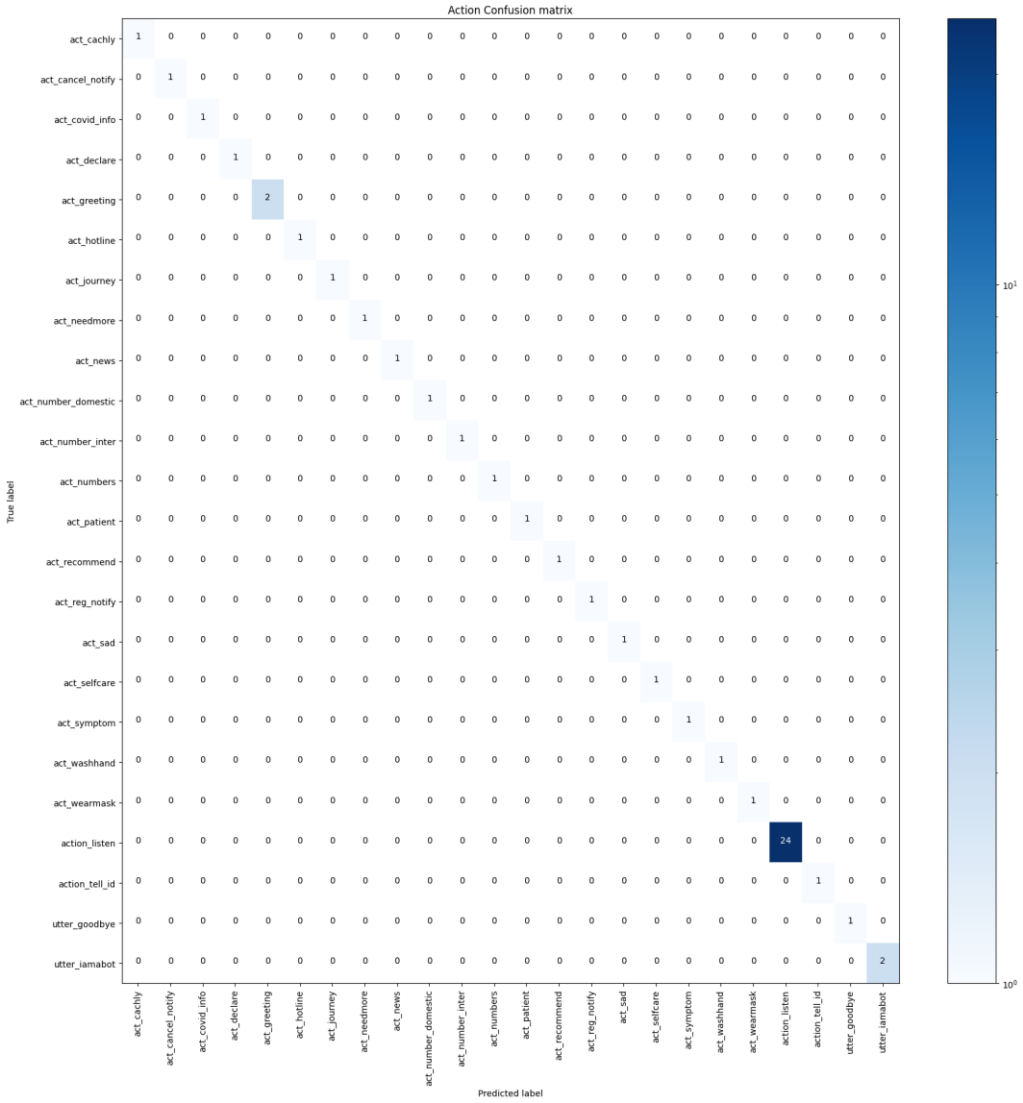
1. **Đánh giá mô hình**
   Đối với các bài toán xử lý những tác vụ cụ thể, kết quả của mô hình là tương đối tốt



Intent Confusion Matrix



Intent histogram



Story confusion matrix

Đối với Kiến trúc DIET cũng cho kết quả outperform SotA ngay cả khi không dùng pretrained embedding

Previous state of the art: intent: 87.55 entities: 84.74

sparse	dense	mask loss	Intent	Entities
✓	✗	✗	87.10±0.75	83.88±0.98
✓	✗	✓	88.19±0.84	85.12±0.85
✗	GloVe	✗	89.20±0.90	84.34±1.03
✓	GloVe	✗	89.38±0.71	84.89±0.91
✗	GloVe	✓	88.78±0.70	85.06±0.84
✓	GloVe	✓	89.13±0.77	86.04±1.09
✗	BERT	✗	87.44±0.92	84.20±0.91
✓	BERT	✗	88.46±0.88	85.26±1.01
✗	BERT	✓	86.92±1.09	83.96±1.33
✓	BERT	✓	87.45±0.67	84.64±1.31
✗	ConveRT	✗	89.76±0.98	86.06±1.38
✓	ConveRT	✗	89.89±0.43	**87.38±0.64**
✗	ConveRT	✓	**90.15±0.68**	85.76±0.80
✓	ConveRT	✓	89.47±0.74	86.04±1.29

2. Cải tiến

Để cho chatbot của chúng ta trở nên mềm mại, nắm bắt được tâm lý người dùng, chúng ta sẽ dùng thêm Module phân tích cảm xúc của người dùng (Sentiment Analysis)
Ở đây sử dụng package Underthesea

```
Anh thích sản phẩm này lắm
{
  "intent": {
    "name": "feedback",
    "confidence": 0.6913111209869385
  },
  "entities": [
    {
      "value": "positive",
      "confidence": 0.5,
      "entity": "sentiment",
      "extractor": "sentiment_extractor"
    }
  ],
```

Từ một câu đầu vào của người dùng, Rasa NLU trả về phần **entities** có

thông tin về sentiment analysis với tên là sentiment, giá trị là positive và confidence fake là 0.5 do Module này không trả về giá trị này.
Nhờ có những giá trị này mà ta có thể phán đoán được cảm xúc người dùng, từ đó đưa ra actions phù hợp với tùy tình huống cụ thể.

IV. Kết luận

Trên đây là những nội dung mà nhóm chúng em tìm hiểu về Chatbot nói chung và ứng dụng Rasa nói riêng, cùng với đó là đề cập tới Module phân tích cảm xúc của người dùng, từ đó cải tiến giúp cho chatbot trở nên mềm dẻo, thân thiện hơn.

Tham khảo

[1] SiqiBao, PLATO-2: Towards Building an Open-Domain Chatbot via Curriculum Learning, Published in May 2021, arXiv:2006.16779v4

[2] Palvlo, Intelligent Dialogue System Based on Deep Learning Technology, 2018

[3] M. Huang, Challenges in Building Intelligent Open-domain Dialog Systems, 2019, arXiv:1905.05709

https://rasa.com/docs/

https://viblo.asia/p/tim-hieu-va-xay-dung-cac-he-thong-chatbot-trong-thuc-the-XL6lA8D4Zek

https://viblo.asia/p/tat-ca-nhung-gi-ban-can-biet-ve-chatbot-Az45bnNg5xY

https://viblo.asia/p/phan-1-deep-learning-cho-chatbot-gioi-thieu-gDVK2QEr5Lj

https://viblo.asia/p/hieu-rasa-qua-quy-trinh-xay-dung-mot-chatbot-giup-ban-tra-loi-cau-hoi-hom-nay-an-gi-WAyK8P4pKxX

https://viblo.asia/p/rasa-chatbot-tang-kha-nang-chatbot-voi-custom-component-va-custom-tokenizationtieng-viet-tieng-nhat-Qbq5QN4mKD8#_xay-dung-module-phan-tich-feedback-nguoi-dungsentiment-analysis-6

https://www.youtube.com/watch?v=wWNMST6t1TA&list=PL75e0qA87dlG-za8eLI6t0_Pbxafk-cxb&index=1