

Content

[Abstract 4](#_Toc49423716)

[1. Introduction 5](#_Toc49423717)

[1.1 Background 5](#_Toc49423718)

[1.2 Motivation 5](#_Toc49423719)

[1.3 Chatbot 6](#_Toc49423720)

[1.4 NLU 6](#_Toc49423721)

[1.5 Rasa 7](#_Toc49423722)

[2. Literature Review 9](#_Toc49423723)

[2.1 Chatbot 9](#_Toc49423724)

[2.1.1 Social chatbot 9](#_Toc49423725)

[2.1.2 Task-oriented chatbot 12](#_Toc49423726)

[2.2 NLU 14](#_Toc49423727)

[2.3 Rasa 15](#_Toc49423728)

[2.3.1 Rasa Architecture 15](#_Toc49423729)

[2.3.2 Compare NLU framework 16](#_Toc49423730)

[2.4 Summary 18](#_Toc49423731)

[3. Methodology 19](#_Toc49423732)

[3.1 Pipeline Components 19](#_Toc49423733)

[3.1.1 Pipeline in Rasa 19](#_Toc49423734)

[3.2 Policies 23](#_Toc49423735)

[3.2.1 Policies in Rasa 23](#_Toc49423736)

[3.3 Stories 25](#_Toc49423737)

[3.3.1 What is story? 25](#_Toc49423738)

[3.3.2 Interactive Learning 25](#_Toc49423739)

[3.3.3 Conclusion 28](#_Toc49423740)

[3.4 Exercise Dataset 28](#_Toc49423741)

[3.4.1 What is Web Crawler 28](#_Toc49423742)

[3.4.2 Beautiful Soup 28](#_Toc49423743)

[3.4.3 Conclusion 30](#_Toc49423744)

[4. Result 32](#_Toc49423745)

[4.1 Design Flow 32](#_Toc49423746)

[4.2 First Question – What part of body 34](#_Toc49423747)

[4.3 Second Question – Which muscle 34](#_Toc49423748)

[4.4 Third Question – Equipment checking 34](#_Toc49423749)

[4.5 With User Interface - Telegram 35](#_Toc49423750)

[4.6 Evaluation 35](#_Toc49423751)

[5. Conclusion 36](#_Toc49423752)

# Abstract

# Introduction

In this chapter, this essay will mention about the background and the motivation of this project. Also, introduce some primary concepts about the chatbot, natural language understanding and Rasa.

## Background

Machines can think. It is a prevalent story in a fiction movie or novel. Such as in movie Prometheus[[1]](#footnote-1), David is an AI robot which helps main actors to analyse information, answer questions and support them during their universe adventure. It is hard to believe that would become true in part of regular life one day in the few decades. However, it seems not absurd anymore.

In recent years, artificial intelligence has explored widely. One of the major topics to be investigated in this field is the chatbot which connects with Machine Learning, Big Data and Natural Language Processing. These research let the machine can understand what does human say and respond to a proper answer to people. For example, Watson is the question-answering system posted in natural language and developed by IBM. In 2011, it defeated the human player to win the first-place prize on Jeopardy![[2]](#footnote-2) Moreover, in 2018, google demo the Google Assistant to make an appointment in an actual phone call.[[3]](#footnote-3)

Due to those significant achievements from the chatbot, people are developing a lot of different type of application in the business market. Also, implement the chatbot on social media like Facebook, WhatsApp, Telegram, etc. to help people solve their problem. It not only reduces the personnel costs but also increases the convenient. That is why the chatbot has gotten popular.

## Motivation

## Chatbot

The chatbot is a software application designed for extended conversations and set up to mimic the human to human chat behaviour. Also, it used to conduct an online chat conversation via text instead of providing direct contact with a live human agent.

Chatbots are regularly utilized in dialogue systems for different purposes including client service, request routing, or for data collection. Some of the chatbot applications use extensive word-classification processes, Natural Language processors, and complicated AI, others scan for common keywords and generate replies using general phrases obtained from an associated library or database.

According to Mnasri (2019), there are two categories of property for the chatbot. One is social chatbots which designed to offer unstructured human-like conversations. The other is task-oriented chatbots which aimed to carry short conversations and accomplish simple tasks related to specific things.

J. Daniel and J. Martin (2018) indicated that chatbot architectures could be divide into two building approaches. First are rule-based systems which involve the early famous ELIZA and PARRY systems. Second are corpus-based systems. It analyses large datasets of human-human conversations, which can be done by using information retrieval or by using a machine translation typical example such as neural network sequence-to-sequence systems, to learn to map from a user announcement to system response.

## NLU

Natural language understanding (NLU) is a part of natural language processing (NLP) in artificial intelligence that uses software to understand what the meaning is from sentences in text or speech.

Natural language processing (NLP) is a subset of AI, and it involves programming computers to process large volumes of language data. It comprises numerous tasks that split natural language into smaller components to understand the relationships between those components and how they work together. Typical tasks include parsing, speech identification, tagging part-of-speech, and data extraction. NLP focuses mainly on transforming text into structured data.

NLU is a vital and challenging subset of natural language processing (NLP). NLU is narrower in purpose, focusing mainly on machine reading comprehension. Although NLU and NLP understand human language, NLU is tasked with interacting with untrained individuals and understanding their intent. It means that NLU goes beyond understanding words and interprets meaning.

NLU uses algorithms to reduce human speech into a structured ontology. AI can find out things like intent, time, locations, and sentiments. For example, I want to reserve a table for two people at the Birmingham restaurant on the 4th of September might break down into this:

intent: [reserve a table] for reservation,

intent: [two people] for how many people,

time: [4th of September] for reservation date,

and locations: [Birmingham restaurant] for the restaurant name.

<https://www.uctoday.com/collaboration/what-is-natural-language-understanding/>

<https://blog.rasa.com/nlp-vs-nlu-whats-the-difference/>

<https://en.wikipedia.org/wiki/Natural-language_understanding>

<https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/>

## Rasa

Rasa is an open-source Machine Learning framework for making chatbots. It supports not only text but also voice-based dialogue. Additionally, it can connect to messaging social media such as Facebook, Telegram, etc., and APIs. Rasa also has an external feature named Rasa X, which uses actual conversations to enhance the chatbot and building chatbot as well.

Rasa helps you create contextual assistants capable of having layered conversations with lots of backwards and forwards. For a human to have a significant exchange with a contextual assistant, the assistant needs to be capable to use context to build on something that was previously discussing.

<https://rasa.com/>

<https://pypi.org/project/rasa-nlu/>

# Literature Review

In this chapter, it will cite different approaches to analysis the chatbot. Section 2.1 lists several past studies for chatbot architecture and applications. Section 2.2 introduces the purpose of the NLU and some NLU studies. Section 2.3 explains how does the Rasa architecture work and contains some critical difference with another framework - Dialogflow. Through these differences to understand why to choose Rasa.

## Chatbot

In this section, it will offer two case studies which are using different methods to construct. First one is social chatbot. Second is task-oriented chatbot.

### Social chatbot

Here will introduce a famous social chatbot example – XiaoIce.

**Introduction**

In May 2014, Microsoft released XiaoIce, which is one of the famous social chatbots. It is like a real friend who can understand users' emotional needs and engage communications.

A recent study by Shum et al. (2018) concluded that challenges and opportunities with social chatbots. With the reproduction of smartphones and the progression of broadband wireless technology, social chatbots are created to assist people's needs for interaction, affection, and social belonging. Consequently, social chatbots must be able to identify the emotion and trace emotional changes during a conversation.

This chatbot also needs the ability to handle multiple tasks when users are talking about the context of random chats. For this reason, the chatbot has to develop a set of ways to serve users' demands. Therefore, the chatbot takes time to talk like a human, express results, offering perspectives, generating new topics to retain the conversation going.

**Design Principles**

The method introduced by Zhou et al. (2020) mentions three main design principles.

1. **IQ + EQ + Personality:**

Knowledge and memory modelling, image and natural language understanding, reasoning, generation, and prediction comprise IQ capacities. These parts are fundamental for the procession of dialogue ability. Additionally, meeting users' specific requests and accomplish their tasks are necessary for social chatbot.

EQ contains empathy and social skills. A social chatbot requires to have the ability to recognise the users' emotions from conversations, detects how the emotion change during chatting and knows the users' emotional requirements to achieve empathy ability. In other words, query understanding, user profiling, emotion detection, sentiment recognition, and dynamically tracking the mood of the user in a conversation are the key points to reach this goal.

Personality is represented as a particular set of actions, cognition, and emotional patterns that form an individual’s unique character.

1. **Conversation-turns Per Session:**

Conversation-turns Per Session (CPS) which is considered to the vital point for social chatbots. CPS is an average number of conversation-turns between the user and the chatbot during the conversation. If CPS is large, it means that the chatbot has better engaged.

1. **Hierarchical Decision-Making:**

To attract users’ interest, XiaoIce tries to promote a variety of conversation modes. For each conversation mode, it runs by a skill that handles a specific type of conversation segment.

XiaoIce uses Markov Decision Processes (MDPs) (Sutton, Precup, and Singh 1999) as the mathematical framework which aims to use a mathematically way to cast human-machine social chat as a hierarchical decision-making process.

Through the MDP and conversation with users, the chatbot would observe the current dialogue state and choose a proper option according to a hierarchical dialogue policy. Afterwards, the chatbot receives a reward and keeps watch new state until the dialogue end.

The design purpose of the chatbot is to obtain optimal policies and skills to maximize the expected CPS (rewards).

**System Architecture**

Zhou et al. (2020) also introduced that XiaoIce system architecture which consists of three layers, user experience, conversation engine and the data.

1. **User experience layer:**

This layer integrates to chat platforms. Moreover, it supports two modes. One is the full-duplex mode which deals with the voice-stream-based conversation. The other is taking turns mode, which handles message-based communications.

1. **Conversation engine layer:**

This layer contains several vital components which are dialogue manager, empathetic computing module, Core Chat, and dialogue skills. The dialogue manager decides to select dialogue skill or Core Chat using the dialogue policy to generate replies by tracking of the dialogue state. The empathetic computing module can understand the content of user input and the empathetic aspects of the conversation.

1. **Data layer:**

This layer consists of the different dataset, which is collected human conversational data, non-conversational data, knowledge graphs, etc.

**Conversation Engine**

Conversation Engine is the primary part of XiaoIce, and this engine helps XiaoIce makes the proper response to the user. According to the above system architecture, conversation contains Dialogue Manager, Empathetic Computing, Core Chat and Dialogue Skills. These components have more detail elements to accomplish the duty. Here will mention some important features.

1. **Dialogue Manager**

Dialogue Manager controls the whole dialogue system. It involves three parts. One of them is the Global State Tracker, which is responsible for tracking the state of the conversation. Another one is Dialogue Policy, which would base on the dialogue state to select an action. Either top-level policy to respond to the user's specific needs or a response suggested by a skill-specific low-level policy can activate the action. The last one is the Topic Manager, which depends on the dialogue turn to make classifier decide to switch a new topic. This behaviour is simulating that human change topic during a conversation.

1. **Contextual Query Understanding**

Contextual Query Understanding (CQU) contains below steps, First is named entity identification, which links all entity mentioned label to the entities saved in working memory of the state tracker and update new entities into the working memory. Next is Co-reference resolution which replaces all pronouns to the corresponding entity names. The last step is Sentence completion, which completes the sentence using contextual information if the sentence is not complete.

**Conclusion**

In summary, social chatbots have to behave like a human. Not only understand the user meaning and finish the task but also have a persona which is always reliable, sympathetic, affectionate and sense of humour. In these purposes, social chatbots have to contain IQ, EQ and personality. XiaoIce achieves this goal because of the system architecture. Primarily, the conversation engine layer helps that XiaoIce can understand what the user needs and response the proper action back. Moreover, this layer is the key to let chatbot obtain the personal characteristic.

[]<https://www.mitpressjournals.org/doi/pdf/10.1162/coli_a_00368>

[From Eliza to XiaoIce: challenges and opportunities with social chatbots]

### Task-oriented chatbot

**Introduction**

The majority of prior research has applied that Task-Oriented Chatbot aims to help the user to achieve their needs (Hussain S et.al, 2019). It means the task-oriented chatbot is designed for dealing with specific situations. For example, booking flight, hotel, accommodation, or scheduling an event, etc. The task-oriented chatbot is good at restricted domains. However, this kind of chatbots cannot process general knowledge or answer the question without their task domain. Instead, they are goal-oriented chatbots concentrated on serving people to accomplish a specific goal.

**Framework**

A series of recent studies has indicated that domain ontology is the primary frame of the recent task-based dialogue system (J. Daniel, J. Martin, 2018). Also, domain ontology is a knowledge structure that draws the sets of intentions the system can extract from user sentences. Additionally, the ontology determines one or more frames, each a collection of slots, and represents the values that each slot can take.

In 1977, Genial Understander System (GUS) introduced this influential frame-based architecture for travel planning (Bobrow et al., 1977). Furthermore, it has influenced most modern commercial digital assistants.

In GUS-style frame, the set of slots designate what the system needs to know. And the filler of each slot is restricted to a value of specific semantic type. The purpose of control architecture of frame-based dialogue systems is to collect the user intends and then process the associated action for the user. Most frame-based dialogue systems are hand-designed for the task by a dialogue designer.

One of the frameworks is using Natural Language Processing (NLU) to be the core of chat interaction. Handoyo et al. (2018) built a ticket chatbot by using Wit.AI, which is an application as one of the natural language interfaces for turning the sentence into structured data. Also, Wit.AI can detect particular intent word in sentences by a built-in block and allow the developer to define the customised entities.

**Conclusion**

In conclusion, the task-oriented chatbot needs to know the intent of the user sentences. And analysis these sentences to a specific entity. It will be filled into a slot or be detected as a special word. Through this procession, the chatbot executes a corresponding action which designed from developers.

[A survey on conversational agents/chatbots classification and design techniques]

## NLU

**Introduction**

What is Natural Language Understanding? A more comprehensive description can be found in the paper from Dahlgren et al. (1991).

The Natural Language Understanding (NLU) is using a similar way which is human understanding to understand a natural language. The natural language is highly ambiguous and redundant. Because the same word may have many different meanings or the same meaning sentence can use a different way to express. Therefore, the NLU module is used to analyse this complicated structure and resolve its meaning layer by layer.

The NLU module arrests the combinatorial explosion, which is many possible meanings and structures could give to words and phrases in a natural language. This problem has happened in the earlier trials to parse and interpret the natural language on a computer. And then this leads to failure. Moreover, the NLU module avoids the complicated issue of common-sense reasoning and solve these problems and implements accurate interpretations.

**Application**

According to Hussain S et al. (2019), Task-oriented chatbot also uses the Natural Language Understanding technology to fill the slot. NLU helps to extract three things from users' utterance. First is domain classification for checking what kind of domain the user is talking, such as flights, travel, restaurant, etc. Second is user intent determination, which is a goal for finding what the user trying to accomplish—for example, the task of the searching for a movie, adding a new schedule, etc. The last is slot filling. The system extracts the distinct slots and fillers from understanding the user's utterance connect their intent.

**Conclusion**

Overall, NLU is used to understand what the human wants to express and what purpose the user intends to do. And process the sentence to specific slots and fillers.

## Rasa

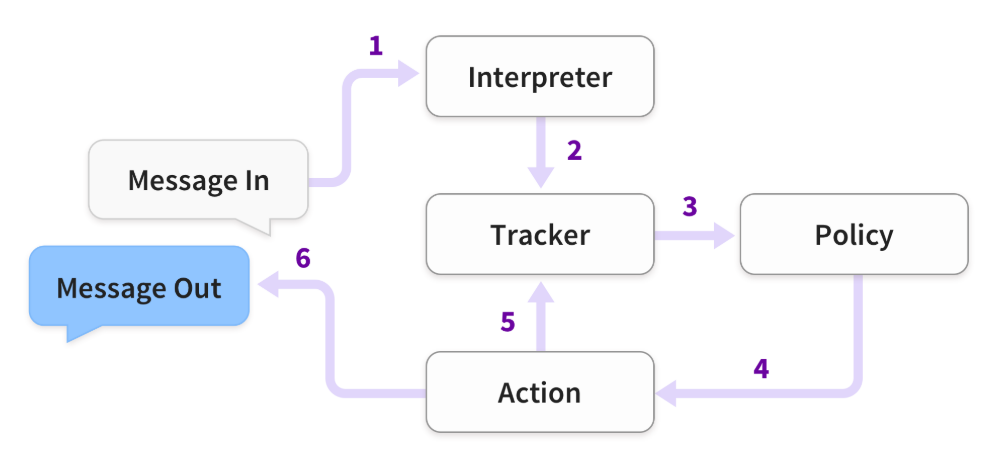
### Rasa Architecture

**Introduction**

According to Rasa document (Bocklisch et al., 2017), Rasa's architecture is modular designing which helps other systems easy to integrate. Rasa Core also can connect with other NLU services as a dialogue manager. Rasa Core and Rasa NLU services can expose HTTP APIs so both can be used in different programming languages, even though the code of Rasa is using Python.

**Architecture**

Dialogue state saves into a tracker object. For each conversation session, there is one tracker object which is an only stateful component in the system. The tracker will store slots and the log of the events which led to the specific state during the conversation. Moreover, replaying the events can reconstruct the state of a conversation.



As the figure shows, this is the steps of rasa architecture work when Rasa receive a user message and take a set of steps. “

1. *A message is received and passed to an Interpreter (e.g. Rasa NLU) to extract the intent, entities, and any other structured information.*
2. *The Tracker maintains conversation state. It receives a notification that a new message has been received.*
3. *The policy receives the current state of the tracker.*
4. *The policy chooses which action to take next.*
5. *The chosen action is logged by the tracker.*
6. *The action is executed (this may include sending a message to the user).*
7. *If the predicted action is not ‘listen’, go back to step 3.*

“(Bocklisch et al., 2017)

**Action**

Dialogue management is considered as a classification problem. Rasa Core predicts which action should execute from a predefined list for each iteration. The action can be a method to execute or reply a simple sentence to the user. After running action, it would be passed a tracker instance. Therefore, the tracker can make use of any related data (slots, previous utterances, and the results of previous actions) collected over the history of the conversation.

**Policy**

The policy is responsible for selecting the next action to execute from the given tracker object. The featurizer instantiates, along with the policy, also creates a vector representation of the current dialogue state base on the given tracker.

**Conclusion**

Rasa easily connects with other programming language modules or NLU services because of modular design. Through the interpreter, tracker, policy and action to understand what does the user mean. And then using tracker and policy to decide what the corresponding action should do.

### Compare NLU framework

There are many different frameworks to develop chatbots on the internet like google Dialogflow and Rasa. This essay will compare Rasa and Dialogflow and explain why I use Rasa to build my chatbot.

(https://www.ideas2it.com/blogs/battle-of-the-bots-rasa-vs-google-dialogflow-vs-aws-lex/)

https://chatbotslife.com/dialogflow-vs-rasa-which-one-to-choose-206fb98b0e90

https://medium.com/@CobusGreyling/a-comparison-of-eight-chatbot-environments-41ad3b795dc7

Lakshmanan (2020) mentions the meaningful difference between Rasa and Dialogflow. It is including model design, integration, and customisation. Here, propose main points influence this dissertation using Rasa.

**Intent/Entity Model:**

Rasa is using the concept of a pipeline for intent and entity. They have adapted a recent state-of-the-art model – Bidirectional Encoder Representation from Transformer (BERT) into the pipeline, which helps developers to build a better model. Moreover, Rasa supports another advantage of combining our customise model into the pipeline for any task.

Dialogflow is a solid platform which contains good models and pre-trained entities. However, it does not support custom models.

**Conversation Flow:**

Rasa is based on the data or conversation flow from given and processes by a transformer model (recent state-of-the-art model). Through this method, it can help developers to create their flow generically.

Dialogflow has dialogue management which drives the conversation by the context provided for intents. Also, it is slightly rule-based and not as flexible.

**Model Tweaking:**

Rasa is open-source, which helps the developer base on their needs to estimate and configure the model.

In Dialogflow, models are closed base. Developers cannot modify or evaluate the models.

**API Integration:**

Any external API such as SQL, Graph DB or our original API can be integrated into Rasa.

Dialogflow provides API integration as well. But Rasa supports the Knowledge graph integration which is helpful to let the developer process a better conversation.

**Deployment:**

Rasa can be set up at the could-based and on-premise as well.

Dialogflow is cloud-based platforms.

**Customization:**

The developer can use Rasa to customise the business logic to implement, model, deployment, and integration. Furthermore, the developer can use SDK which Rasa provides to execute your custom logic.

Dialogflow can let developer customise training data and the input rule-based dialogue flow. But it does not support any customisation based on the business needs.

**Conclusion**

This is an essential finding in the understanding of the difference between these two frameworks. We could know that Rasa is more flexible to function customised to implement into the chatbot. Additionally, using these customisation supports can help the program be better and more general. Although the developer requires technical knowledge and python coding skill, Rasa still provides better performance than Dialogflow.

## 2.4 Summary

These findings provide a basic idea for building a chatbot. And what kind of chatbot match this project. Also, realise what the theory about Natural Language Understanding in the chatbot. Furthermore, compare several NLU frameworks such as Rasa and Dialogflow to make sure what are the advantages that can help development. Finally, according to the above introduction, this project belongs to the task-oriented chatbot and using Rasa to become the framework.

# Methodology

In this chapter, it will introduce what the pipeline, the policy and the story is in Rasa. Through using these to make the chatbot smarter to handle the process. Section 3.1 and 3.2 will mention what pipeline components and policies are in Rasa and why this project needs these. Section 3.3 is about the story. Finally, section 3.4 is how to get the exercise dataset by the web crawler.

## Pipeline Components

In Rasa, we have to set a sequence of components to help us process the incoming message. According to Rasa docs, these components are run one after another. Additionally, we must define these components in config.yml. Further, the NLU pipeline allows us to customise the model and finetune in the dataset.

### Pipeline in Rasa

The following is the categories of pipeline and the description of what the purpose is if choosing this pipeline. It can help us to clarify what we need in the development of chatbot.

**World Vector Sources**

The following components are pre-trained models. We can use this if we need pre-trained word vectors. Pre-trained word vectors are an efficient way to get started with fewer data because the word vectors based on large amounts of data to train.

* *SpacyNLP*

This is used to initialise spaCy structures. Every spaCy component relies on this pipeline. It should be put at the start of the pipeline if using any spaCy components.

* *MitieNLP*

This is used to initialise MITIE structure. Every MITIE component relies on this pipeline. It should be put at the start of the pipeline if using any MITIE components.

* *HFTransformersNLP*

This is used to initialise pre-trained language model from HuggingFace’s Transformers library. The component uses language model specific tokenisation and featurisation to analyse sequence and sentence level representations for each example in the training data.

**Text Tokenizers**

Tokenizers is processing the string, text into a list of tokens by tokenizing or splitting.

* *WhitespaceTokenizer*

This creates a token for every whitespace separated character sequence.

* *JiebaTokenizer*

It is specifically to handle the Chinese language. Creates tokens using the Jieba tokenizer for the Chinese language.

* *MitieTokenizer*

This creates tokens using the MITIE tokenizer.

* *SpacyTokenizer*

This creates tokens using the spaCy tokenizer.

* *ConveRTTokenizer*

This creates tokens using the ConveRT tokenizer.

* *LanguageModelTokenizer*

This creates tokens using the pre-trained language model specified in upstream HFTransformersNLP component.

**Text Featurizers**

The featurizer can convert the tokens and their characteristics into features that can be adopted by machine learning algorithms.

* *MitieFeaturizer*

This creates a vector representation of user message, features for entity extraction, intent classification, and response classification using the MITIE featurizer.

* *SpacyFeaturizer*

This creates a vector representation of user message, features for entity extraction, intent classification, and response classification using the spaCy featurizer.

* *ConveRTFeaturizer*

This creates a vector representation of user message by the default signature computing, features for entity extraction, intent classification, and response using ConveRT model.

* *LanguageModelFeaturizer*

This creates a vector representation of user message by the pre-trained language model in HFTransformersNLP component computing, features for entity extraction, intent classification, and response using a pre-trained language model.

* *RegexFeaturizer*

RegexFeaturizer produces a list of regular expressions defined in the training data format during training. For each regex, if the expression can be found in the user message, a feature will be set marking.

* *CountVectorsFeaturizer*

This creates a bag-of-words representation of user messages, intents, and responses using sklearn CountVectorizer. All tokens which compose only of digits will be distributed to the same feature.

* *LexicalSyntacticFeaturizer*

This creates lexical and syntactic features for a user message to help entity extraction.

**Intent Classifiers**

Intent classifiers designate one of the intents defined in the domain file to incoming user messages and generate intention with the confidence level.

* *MitieIntentClassifier*

This classifier applies MITIE to achieve intent classification. Also, it uses a multi-class linear SVM with a sparse linear kernel.

* *SklearnIntentClassifier*

This SklearnIntentClassifier trains a linear SVM with a grid search to be optimized.

* *EmbeddingIntentClassifier*

The classifier embeds user inputs and intent labels into the same space. And according to StarSpace algorithm, through maximizing the similarity between inputs and labels to train supervised embeddings.

* *KeywordIntentClassifier*

This classifier uses keywords to search for a message. Also, it only searches for an exact match of the keyword string in the user message, including case sensitive.

* *DIETClassifier*

This classifier is using a multi-task structure for intent classification and entity recognition.

**Entity Extractors**

Entity extractors extract entities such as names, location, or some other specific words from the user message.

* *MitieEntityExtractor*

This entity extractor uses the MITIE entity extraction to discover entities in user message.

* *SpacyEntityExtractor*

This intent extractor uses spaCy which adopts a statistical BILOU transition model predicts the entities of the message.

* *EntitySynonymMapper*

This intent extractor will let the detected entity value map to the corresponding value if the training data includes these defined synonyms. For example, the component can allow mapping the entities "New York" to "NY."

* *CRFEntityExtractor*

This extractor uses conditional random fields (CRF) to perform named entity recognition.

* *DucklingHTTPExtractor*

This extractor supports to identify dates, numbers, distances, and other structured entities and normalize them.

## Policies

According to section 2.3.1, this essay indicates that the policy is responsible for picking the next action.

### Policies in Rasa

Following is the description of what the specific policy does. It can help us find a proper policy to make chatbot do the correct action when the user sends the message. Afterwards, based on what we need to add to config.yml.

**Keras Policy**

The KerasPolicy adopts a neural network implemented in Keras to choose the next action. The architecture can override in the KerasPolicy.model\_architecture method. Otherwise, the default is based on the LSTM.

**TED Policy**

The Transformer Embedding Dialogue (TED) Policy has a pre-defined structure, which composes of following.

1. Pre-transformer embedding layer obtains an input vector which connects user intent and entities, previous system actions, slots and active forms for each time step.
2. Feed to the transformer.
3. Implement a dense layer to the output of the transformer to get embeddings of dialogue for each time step.
4. Use a dense layer to build embeddings for system actions for each time step.
5. Based on the StarSpace idea, compute the similarity between the dialogue embedding and embedded system actions.

**Mapping Policy**

The Mapping Policy is used to map an intent to a specific action and assigned by providing an intent with the property triggers. The bot will execute the mapped action when it obtains a message of triggering intent.

**Memoization Policy**

The Memorization Policy remembers the conversation in training data. It predicts the next action with the confidence level.

**Augmented Memoization Policy**

The Augmented Memoization Policy, which is similar to Memoization Policy, remembers examples from training stories for up to max\_history turns. Furthermore, it has a forgetting design which will forget some steps in the conversation history and try to find an equivalent in the stories with the reduced history.

**Fallback Policy**

The Fallback Policy triggers a fallback action when at least one of the following happens.

1. The confidence of intent recognition confidence below nlu\_threshold.
2. The confidence difference between the highest-ranked and the second-highest ranked intent less than ambiguity\_threshold.
3. If no dialogue policies predict action with confidence higher than core\_threshold.

**Two-Stage Fallback Policy**

The Two-Stage Fallback Policy processes low NLU confidence in multiple stages by reducing ambiguous from user input.

1. Suppose NLU prediction has a low confidence score, it will ask the user to confirm the classification of intent.

* If the user affirms, the story continues.
* If the user denies, the bot will ask the user to rephrase their message.

1. Rephrasing:

* The story continues if the classification of rephrased intent is confident.
* Otherwise, the bot will ask the user to affirm the classified intent.

1. Second affirmation:

* The story continues if the user affirms.
* If the user denies, the ultimate fallback action will be triggered.

**Form Policy**

The Form Policy is an expansion of the MemoizationPolicy which manipulates the filling of forms. When calling a Form Action, the Form Policy will predict the Form Action until filling all the required slots.

## Stories

### What is story?

Rasa stories are a frame of training data. Rasa Core dialogue management models are used to train by the stories. The story is a specific format for presenting a conversation between the user and the AI chatbot. The user inputs are represented as corresponding intents or entities (if necessary), while the responses of the AI chatbot are designated as corresponding action names.

When editing stories, the story does not need to fill the specific contents of messages that the user may send. Instead, using the advantage of the output from the NLU pipeline, which helps to use the combination of intent and entities to show to all the possible messages the users can send to mean the similar thing.

### Interactive Learning

Interactive learning is an effective way to train the AI assistant and through training stories when talking to the chatbot. Also, it is the easiest way to find out any mistakes the bot made. The bot will ask to check the feedback after every intent classification and response prediction it made during the interactive learning process. Furthermore, after training, Rasa can export the NLU dialogue training examples and stack to the beginning training data sample for all communications with AI assistant in an interactive learning process.

In Rasa framework, it is easy to get in the interactive mode, key in “rasa run actions --actions actions & rasa interactive” at the terminal. Then Rasa will autorun the interactive mode. In this mode, Rasa will stop after the prediction made by NLU and Core and confirm with the developer whether it is correct before proceeding. As the following example figure, the chat history and other information will show on the screen. And it detected that the user message is greeting. So, type “y” to wait Rasa make next prediction until Rasa gives the wrong prediction.

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

The following two figures are the scenario for the wrong prediction. Press “n” if Rasa makes the wrong prediction. Then Rasa will prompt the list of the possible execution, which are the reply message or the action for the next step to ask the developer to choose the correct one.

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 鳥 的圖片

自動產生的描述

After the interactive learning, all the conversation stories and NLU data will update in “Stories.md”. Repeat these steps to help the chatbot make the decision be more accuracy.

### Conclusion

Rasa story is powerful and easy use. It is a training dataset for helping the chatbot can respond correctly. Through the pipeline and stories give a proper prediction. Furthermore, the interactive mode provides more detail information to let the developer know what the forecast is, does the next action is correct. Also, the developer can finetune actions, and the newest results can stack with previous data. It can let the chatbot become more completive.

<https://blog.rasa.com/designing-rasa-training-stories/>

<https://rasa.com/docs/rasa/core/stories/>

<https://rasa.com/docs/rasa/core/interactive-learning/>

## Exercise Dataset

In this section, the essay will introduce web crawler because, in this project, we cannot find any open and proper dataset. However, we found the website which presents the related data we need. So, we decide to use this technic to help us get the dataset.

### What is Web Crawler

The web crawler purpose is web indexing which is using the internet bot to scan the world wide web systematically. Some websites or web search engines would use this technic to expand their web content or build the index of other websites. The web crawler copies pages and through a search engine to index the download the pages so that the user can explore more efficiently.

### Beautiful Soup

Beautiful Soup is a web crawler library based on Python designed for quick turnaround projects like screen-scraping.

Beautiful Soup has some quite useful, straightforward methods and using Python for navigating, searching and modifying a parse tree. Also, it has a toolkit for analysing the document and extracting what you need. Additionally, Beautiful Soup automatically converts incoming data to Unicode and output the reports to UTF-8. Or you can define the encoding by yourself. Moreover, Beautiful Soup owns the Python parsers like html5lib, which allows trying different parsing strategies.

**Parser**

In Beautiful Soup, it can use the different parser to parse the document. Each has its drawbacks and advantages.

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

In this paper, we choose Python’s html.parser to be our parser. The reason we use this parse is that this is standard parser in Python. We do not need to install others to parse the texts. Although speed and lenient performance are not the best, it still performs well.

**Constructor**

Use BeautifulSoup as a constructor to parse the document.

一張含有 鳥 的圖片

自動產生的描述

The document will be converted to Unicode as well as HTML entities are converted to Unicode characters.

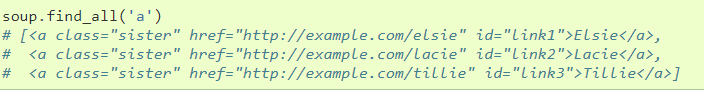


**Find all method**

After passing the document into BeautifulSoup as a Unicode file, we can use "find\_all" to find the specific HTML tag to search the section we need. For example, we search <a> tag, and it will return the corresponding information.

一張含有 螢幕擷取畫面 的圖片

自動產生的描述



### Conclusion

The web crawler is useful and powerful to help us to download the information from the website. Although the Beautiful Soup needs some HTML and Python knowledge to understand what part of the information we are looking. It is still straightforward to use, and easy to choose different parse strategies.

<https://en.wikipedia.org/wiki/Web_crawler>

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

<https://docs.python.org/3/library/html.parser.html>

[Tokenization] <https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/>

# Result

In this chapter, this paper will mention the result of this project. In section 4.1, it will discuss the flow chart, which presents the whole process in this project and what the behaviour is when getting the user message. Section 4.2 to 4.4 shows the result when receiving the message from the user in the chatbot. Section 4.5 introduces the connection with User Interface – Telegram and the result. The last section is the evaluation.

## Design Flow

**Overview**

The chatbot is designed for three main part question, through these questions to recommend what user want to know the exercise. Moreover, the chatbot will depend on the user's utterances to decide what the action should do.

**The First Question**

The first question is "what part of the body would you want to train?". Please refer following description and flow chart.

This question is to understand what the user wants to know. We can suppose that the user would have three types of feedback. The first one is the user knows what he/she needs and answer the question - part of the body and move on the next question.

The second condition is that the user may be a beginner; users have no idea about what part of the body them can train. So, they may need more detail to realise what part of the body they can choose. After the chatbot giving the user more information, the chatbot will wait for the user response.

The last one is a random recommendation. Some users may do not care what they need; tell them some exercise they can do.

一張含有 文字 的圖片

自動產生的描述

**The Second Question**

一張含有 文字, 地圖 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

## First Question – What part of body

## Second Question – Which muscle

## Third Question – Equipment checking

## With User Interface - Telegram

## Evaluation

# Conclusion

1. <https://www.imdb.com/title/tt1446714/> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Watson_(computer)> [↑](#footnote-ref-2)
3. <https://www.theverge.com/2018/5/8/17332070/google-assistant-makes-phone-call-demo-duplex-io-2018> [↑](#footnote-ref-3)