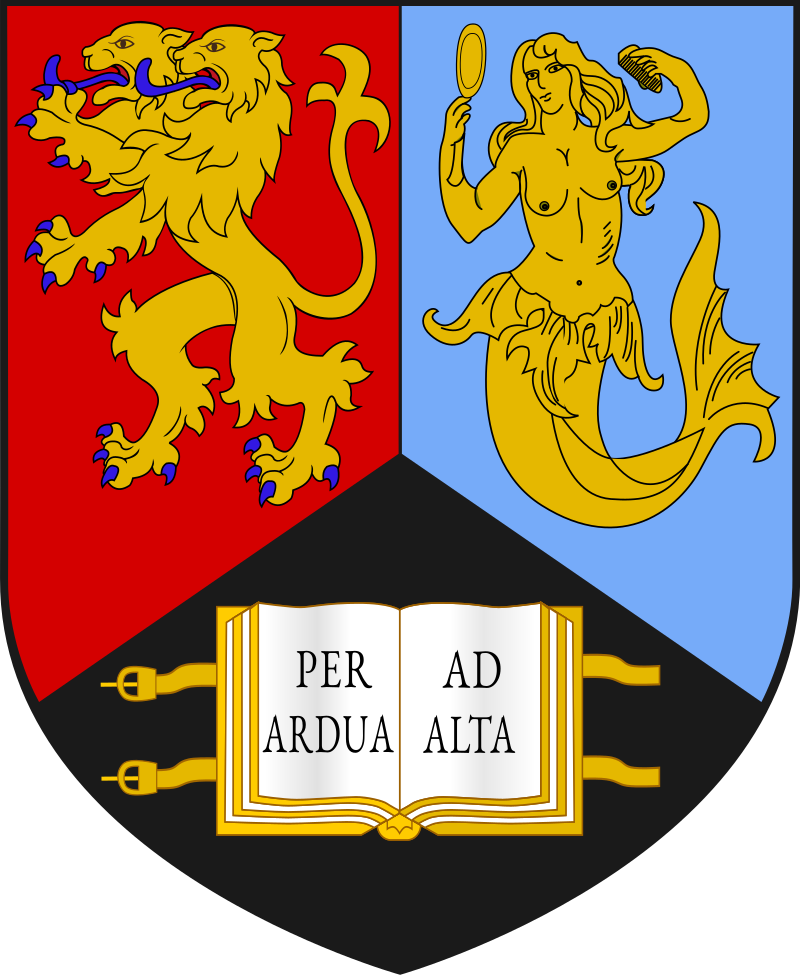
**Bodybuilding Exercise Recommendation Chatbot**

By

Ta-Yu Mar

Student ID: 2020984



Supervisor: Dr. Martin Escardo

A thesis submitted to the University of Birmingham

For the degree of MSc in Conversion Computer Science

School of Computer Science

University of Birmingham

Birmingham, UK

September 2020

**Content**

[Abstract 1](#_Toc49948824)

[1. Introduction 2](#_Toc49948825)

[1.1 Background 2](#_Toc49948826)

[1.2 Motivation 3](#_Toc49948827)

[1.3 Chatbot 3](#_Toc49948828)

[1.4 NLU 4](#_Toc49948829)

[1.5 Rasa 5](#_Toc49948830)

[2. Literature Review 6](#_Toc49948831)

[2.1 Chatbot 6](#_Toc49948832)

[2.1.1 Social chatbot 6](#_Toc49948833)

[2.1.2 Task-oriented chatbot 9](#_Toc49948834)

[2.2 NLU 11](#_Toc49948835)

[2.3 Rasa 12](#_Toc49948836)

[2.3.1 Rasa Architecture 12](#_Toc49948837)

[2.3.2 Compare NLU framework 13](#_Toc49948838)

[2.4 Summary 15](#_Toc49948839)

[3. Methodology 16](#_Toc49948840)

[3.1 Pipeline Components 16](#_Toc49948841)

[3.1.1 Pipeline in Rasa 16](#_Toc49948842)

[3.1.2 Pipeline choosing in project 17](#_Toc49948843)

[3.2 Policies 18](#_Toc49948844)

[3.2.1 Policies in Rasa 18](#_Toc49948845)

[3.3 Stories 19](#_Toc49948846)

[3.3.1 What is story? 19](#_Toc49948847)

[3.3.2 Interactive Learning 19](#_Toc49948848)

[3.3.3 Conclusion 22](#_Toc49948849)

[3.4 Integrated to Telegram 22](#_Toc49948850)

[3.5 Exercise Dataset 23](#_Toc49948851)

[3.5.1 What is Web Crawler 23](#_Toc49948852)

[3.5.2 Beautiful Soup 24](#_Toc49948853)

[3.5.3 Conclusion 26](#_Toc49948854)

[3.6 Summary 26](#_Toc49948855)

[4. Result 27](#_Toc49948856)

[4.1 Design Flow 27](#_Toc49948857)

[4.2 Exercise Dataset 30](#_Toc49948858)

[4.3 #Question1 – What part of body 32](#_Toc49948859)

[4.4 #Question2 – Which muscle 36](#_Toc49948860)

[4.5 #Question3 – What equipment 39](#_Toc49948861)

[4.6 With User Interface – Telegram 42](#_Toc49948862)

[4.7 Evaluation 45](#_Toc49948863)

[5. Conclusion 47](#_Toc49948864)

[Reference 49](#_Toc49948865)

[Appendices 50](#_Toc49948866)

[Appendix A: Chatbot 50](#_Toc49948867)

[Appendix B: Telegram 51](#_Toc49948868)

[Appendix C: Publish to Telegram 51](#_Toc49948869)

[Appendix D: Web Crawler 52](#_Toc49948870)

# Abstract

This project aimed to develop the chatbot, and it can help people get the recommendation of bodybuilding exercise via the conversion, also using this chatbot can prevent injury and know how to do the exercise correctly.

To construct the chatbot, through understanding what the chatbot categories are, how makes natural language understanding work.

Further, use Rasa to be the development environment. It provides the pipeline, policy, and stories to help the machine learn what the user means and make a correct response.

Besides, using python and the web crawler to obtain the training dataset. We use the Telegram to be our user interface. Rasa provides the connection to integrate.

**Keyword:** Chatbot, Natural Language Understanding, Task-Oriented Chatbot, Rasa, Telegram, Web Crawler

# Introduction

This purpose of the project is helping people via conversation to search the workout exercise and using this chatbot to prevent any uncertain information and protect to get an injury. So, 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

Bodybuilding is getting popular recently. Most people would like to through the workout to keep themselves stay ideal shapes. However, if we did not know how to do the exercise correctly, it may cause injury. In my personal experience, when I was a beginner, I did the hard work on searching the video, blogs, etc. to help myself to gain bodybuilding knowledge. Therefore, if there is an application could help people to do more with less, people must increase their intent to do the training.

Furthermore, since Siri has been invented and AlphaGo has defeated human, these topics raise my willing to realise the AI field more. Therefore, this project aims to build a chatbot by combining bodybuilding to help people can find the exercise they want to know.

## 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)[[4]](#footnote-4) 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[[5]](#footnote-5) 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[[6]](#footnote-6). 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.

## Rasa

Rasa is an open-source Machine Learning framework[[7]](#footnote-7) 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.

# 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 categories which are using different methods and purpose to construct. First one is social chatbot. Second is task-oriented chatbot.

### Social chatbot

The social chatbot can perform like a real human to talk. There is much research about social chatbots such as ELIZA (Weizenbaum, 1966), ALICE (Wallace, 2009) and XiaoIce (Shum et al., 2018).

Microsoft makes XiaoIce, and Microsoft is a famous company as well. So, we decide this section 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.

### 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.

## NLU

**Introduction**

What is Natural Language Understanding? A more comprehensive description can be found in the paper from Dahlgren et al. (1998).

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.

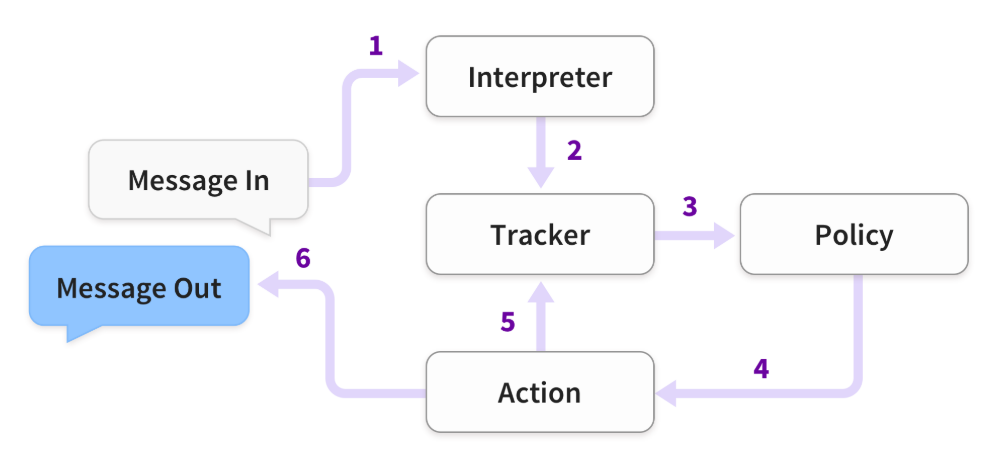


Figure 1

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 using Rasa to build my chatbot.

There are some blogs[[8]](#footnote-8)[[9]](#footnote-9)[[10]](#footnote-10) mention 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.

**Operation**

The developer needs to understand how to program Python and requires artificial intelligence knowledge.

Dialogflow is easy to use, and do not need any specific techies to create the chatbot.

**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 what are components in this project. Section 3.3 is about the story. Section 3.4 shows that how to connect to other service. Finally, section 3.5 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**

This type of 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.

**Text Tokenizers**

Tokenizers[[11]](#footnote-11) is processing the string, text into a list of tokens by tokenizing or splitting.

**Text Featurizers**

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

**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.

**Entity Extractors**

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

### Pipeline choosing in project

The following components are used in this project. These components help the chatbot to process the input message and give the proper classification.

**World Vector Sources**

* 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.

**Text Tokenizers**

* SpacyTokenizer

This creates tokens using the spaCy tokenizer.

**Text Featurizers**

* SpacyFeaturizer

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

* 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.

**Intent Classifiers**

* 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.

**Entity Extractors**

* CRFEntityExtractor

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

## 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.

**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.

**Memoization Policy**

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

## Stories

### What is story?

Rasa stories[[12]](#footnote-12) 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[[13]](#footnote-13) 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[[14]](#footnote-14), 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

Figure 2

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

Figure 3

一張含有 鳥 的圖片

自動產生的描述

Figure 4

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.

## Integrated to Telegram

This section will introduce how does Rasa connect the chatbot to other platforms and how to register the chatbot in Telegram.

**Rasa**

If we have to connect to other services like the Facebook Message, Telegram, Slack and so on, Rasa[[15]](#footnote-15) uses "credentials.yml" to be the interface and act as a bridge to connect each other.

As below figure, need to get the "access\_token" from Telegram and set this key in yml file. Verify is a slot for your chatbot name. Webhook\_url is the address for your web services.

一張含有 鳥 的圖片

自動產生的描述

Figure 5

**Telegram**

Telegram[[16]](#footnote-16) provides an application which calls "Bot Father" to help the developer to create and publish their chatbot on the internet.

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

Figure 6

## 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[[17]](#footnote-17) 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[[18]](#footnote-18) 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

Figure 7

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

Figure 8

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



Figure 9

**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

Figure 10

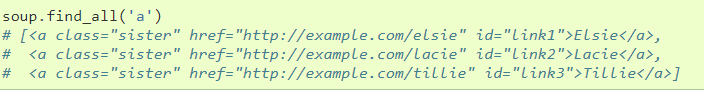


Figure 11

### 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.

## Summary

These methodologies help us have a basic concept and ability to build a chatbot. Through Rasa's pipeline, we can process the training data. Also, the policy makes the chatbot can choose the exact decision. Besides, the stories help us adjust the process of judgment. Further, Rasa presents the interface to connect with the Telegram and publish the chatbot to the internet. Finally, Beautiful Soup gives the web crawler function and helps us download the data from the website by using Python.

# 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 is the data in the project, which includes web crawler dataset and training dataset. Section 4.3 to 4.5 shows the result when receiving the message from the user in the chatbot. Section 4.6 introduces the connection with User Interface – Telegram and the outcome. 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?". 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 to 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

Figure 12

**The Second Question**

The second question is according to the previous answer and checking the specific muscle which the user wants to train. Similarly, we can suppose that the user would have three types of reaction.

The first one is that users already know what they require and return the answer - the specific muscle they want and move on to the next question.

The second condition is that the user may not be familiar with the specific muscle name in this assigned body. So, they need more detail information to recognise what the muscles are in the body. After the chatbot giving the user more knowledge, the chatbot will wait for the user response.

The third one is a random recommendation. Some users may do not care what choice they have. The chatbot will pick randomly one of the muscles based on the previous answer.

一張含有 文字, 地圖 的圖片

自動產生的描述

Figure 13

**The Third Question**

The Third question is checking what the equipment does the user want to use. Likewise, we still assume that the user would have three types of response.

The first one is that users already know what they want to use and return the answer. Then chatbot will base on those answers to find the proper exercise to the user.

The second condition is that the user may no have any idea about what the equipment there are. Therefore, they need a list of those facilities to help them make a choice. After the chatbot giving the user more knowledge, the chatbot will wait for the user response.

The third one is a random suggestion. The chatbot will pick randomly one equipment.

一張含有 文字 的圖片

自動產生的描述

Figure 14

## Exercise Dataset

**Exercise Dataset**

At the beginning of the project, we cannot find the question-answering dataset which related to bodybuilding exercise. For this reason, we turn the topic to search for the workout exercise. Then we found a website which contains the resource we need. So, we referred the website and built the dataset by ourselves.

We refer the website - ExRx.net to be our dataset resource. We are using the web crawler to download and classify the muscles and exercises. We have to understand the HTML file (figure 15) and find out the keyword we need. Then download the data into the CSV file as figure 16.



Figure 15

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

Figure 16

In figure 17, we use the same method to make the relation between the part of the body and muscle. Moreover, we add one more column manually for mapping general muscle name and muscle name on the website.

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

Figure 17

**Dataset in Rasa**

Based on data from web crawler, making the NLU dataset to let the chatbot learn the body, muscle names, etc. For example, the figure 18 shows that the intent, entity, and value for training the chatbot to understand the part of the body.

一張含有 螢幕擷取畫面, 手機 的圖片

自動產生的描述

Figure 18

This project is using stories to train the chatbot to decide what the next action should do. The following picture is an example show that the happy path which means the users input the message without error.

一張含有 螢幕擷取畫面, 手機 的圖片

自動產生的描述

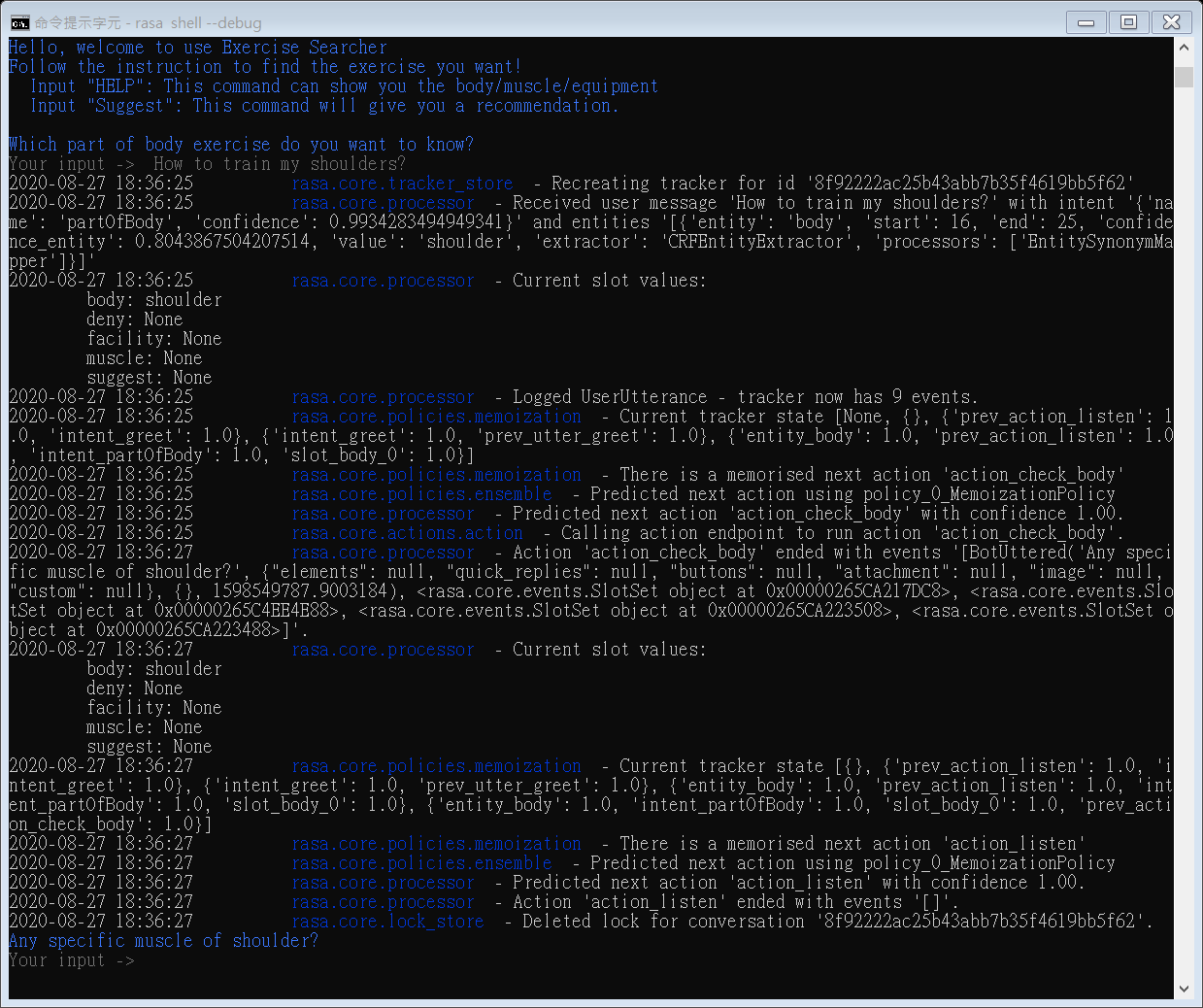
Figure 19

## #Question1 – What part of body

In this section, this essay will show that the result based on the introduction of the previous section. Rasa provides the debug mode shell to help the developer to check the prediction rate, slots value and execution result.

**Get part of body**

As the figure 20, the user inputs the "How to train my shoulders?"(A). Rasa will analyse this sentence and find out what the keyword is. Then return the next action - "Any specific muscle of shoulder."(B).



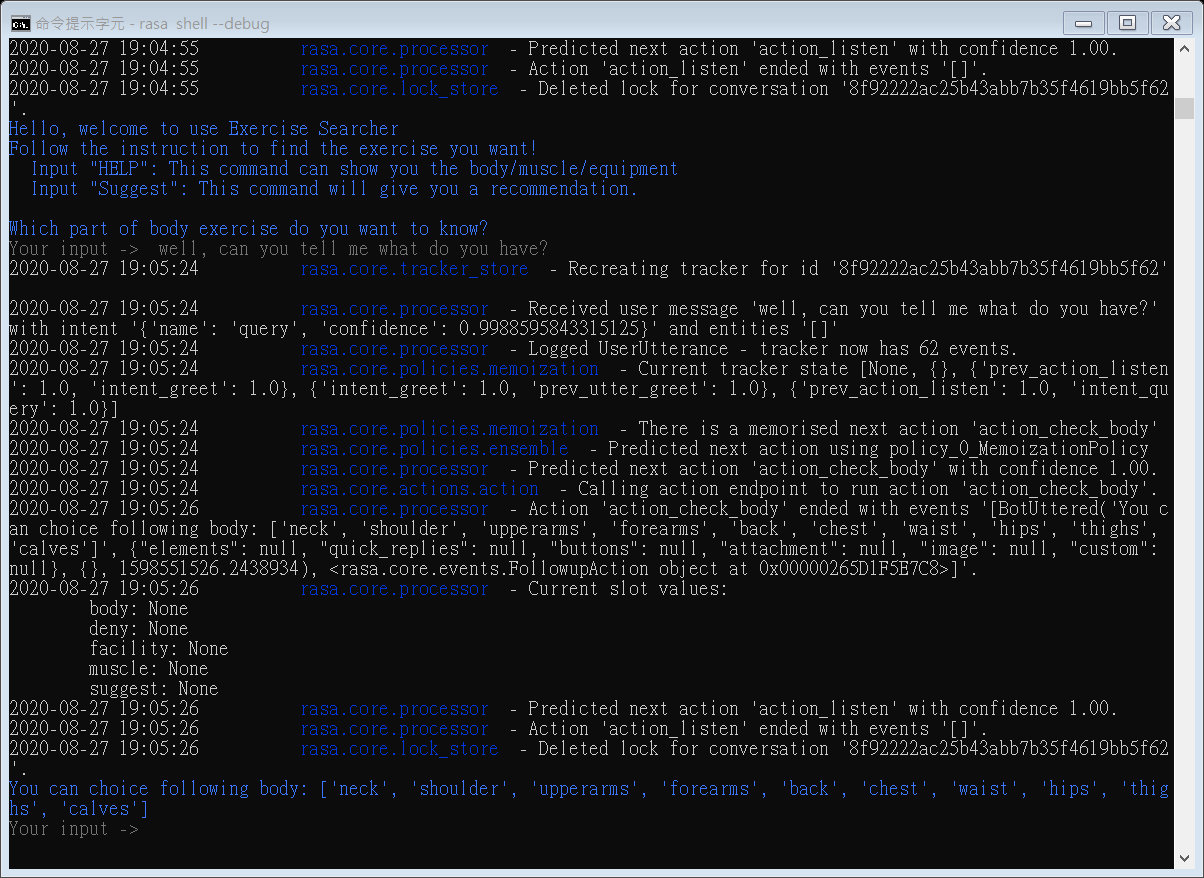
**(A)**

**(B)**

Figure 20

**Need detail information**

As the figure 21, the user inputs the "Well, can you tell me what do you have?"(C). After the processing, the chatbot shows more detail information(D) to the user.

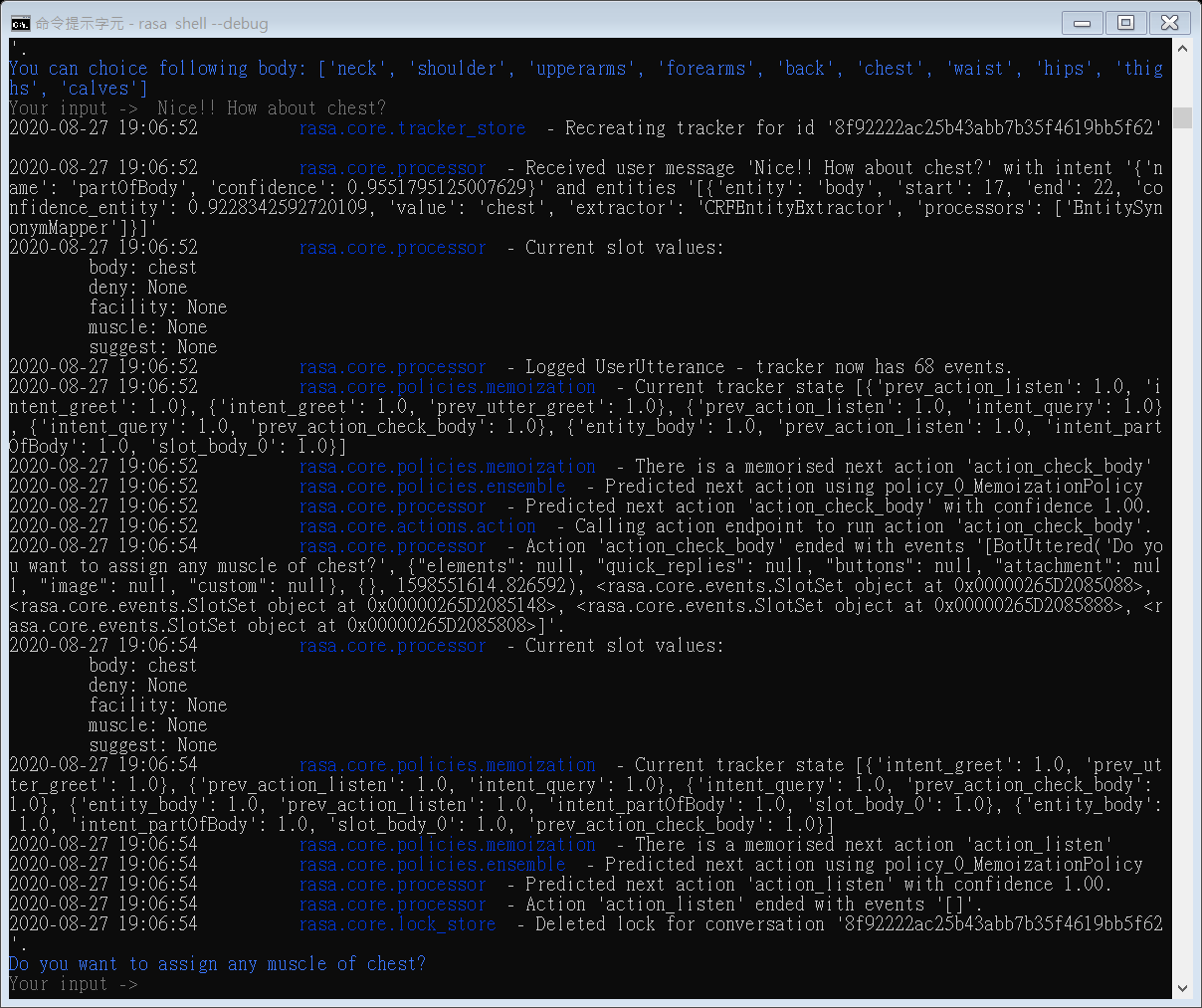


**(D)**

**(C)**

Figure 21

The chatbot waits for users to give what they want(E). Then the same as the previous step, the chatbot returns the next action(F).



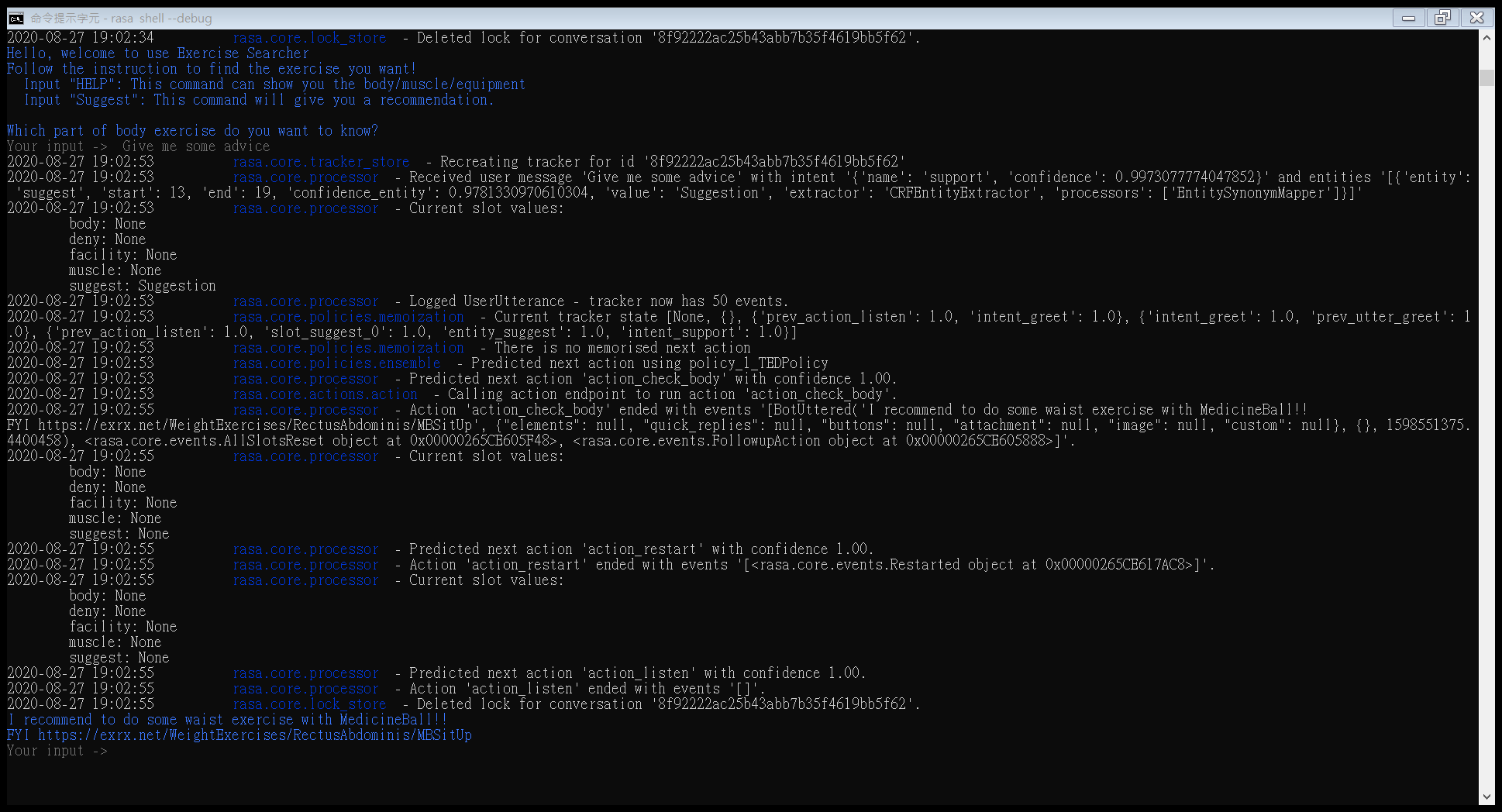
**(E)**

**(F)**

Figure 22

**Recommendation**

The following figure shows that the user asks to give the advice(G). The chatbot return the random exercise(H) back.



**(G)**

**(H)**

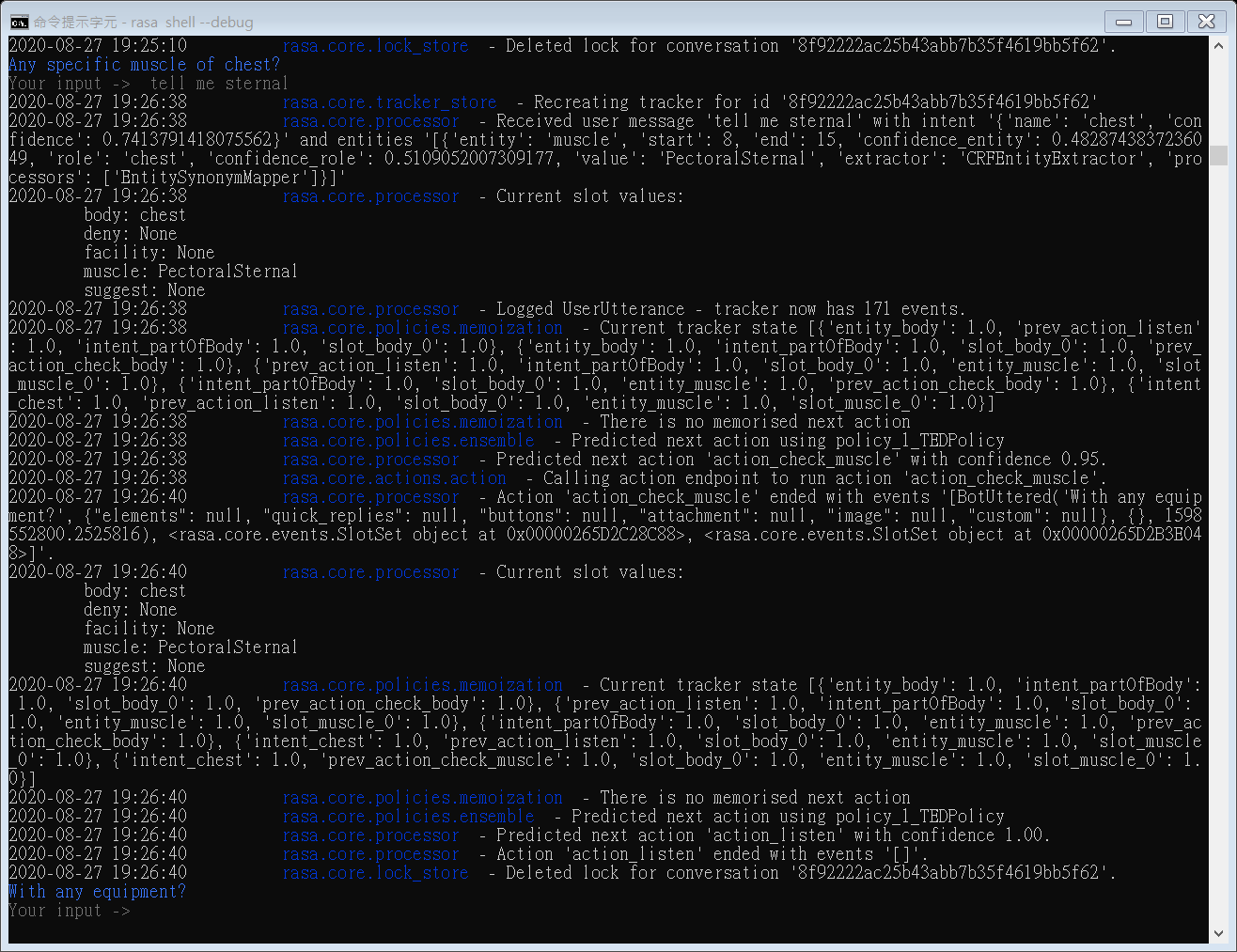
Figure 23

## #Question2 – Which muscle

This section baes on the previous answer to check the next question - what muscle.

**Get muscle**

As the figure 24, the user gives the specific muscle (I). Rasa will process and check this is valid then return the next action (J).



**(J)**

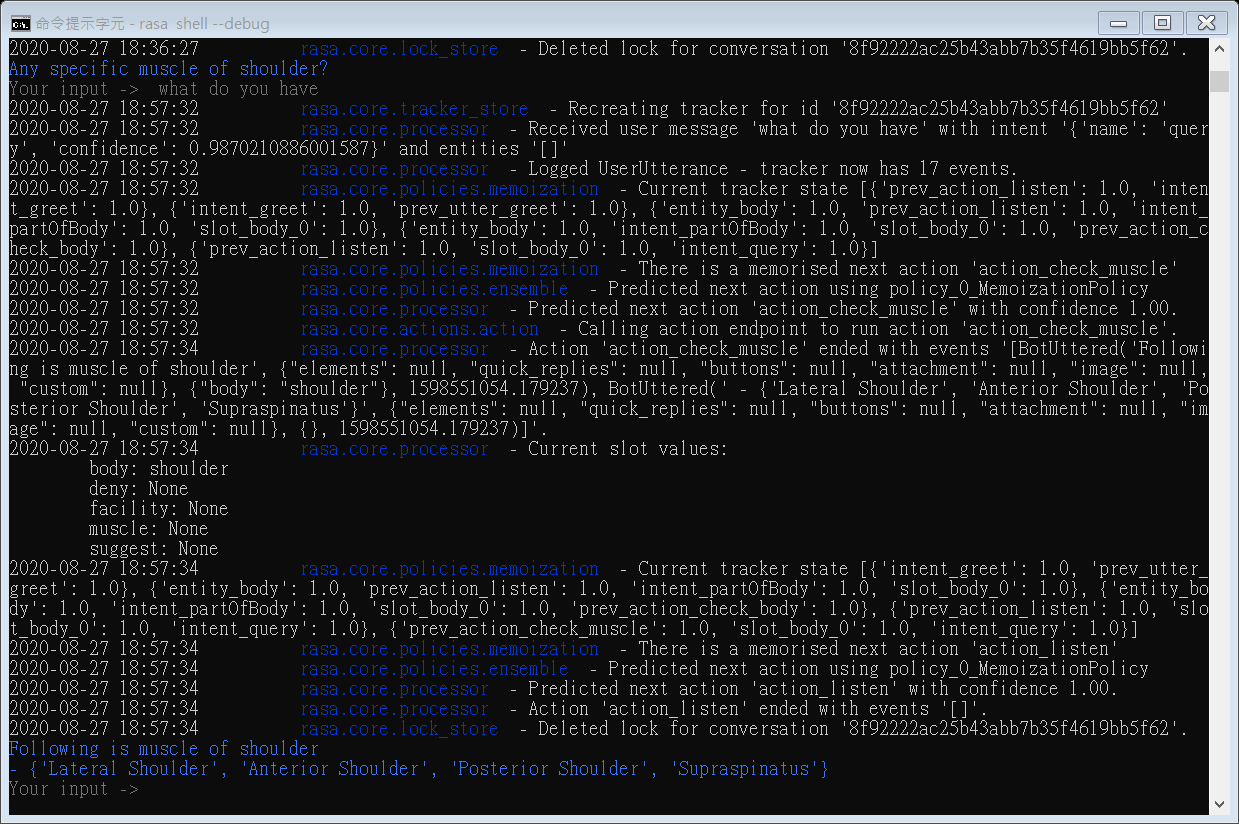
**(I)**

Figure 24

**Need detail information**

Figure 25 shows that the user wants to know the more about what muscle there is in the shoulder (K). The chatbot gives the detail information (L) back.

.

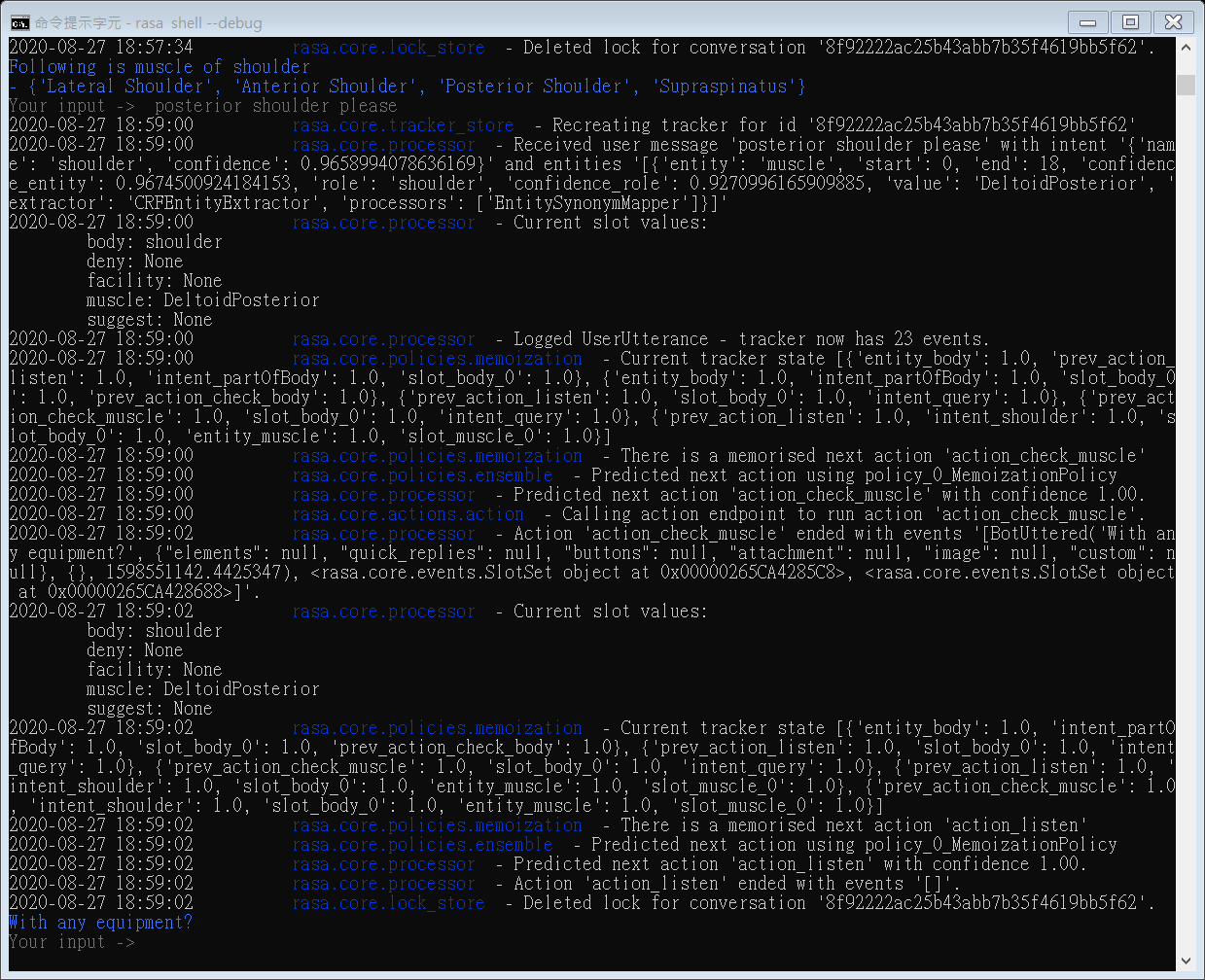


**(L)**

**(K)**

Figure 25

After getting detail, the user decides the one of muscles (M). The chatbot returns the correspond action (N).



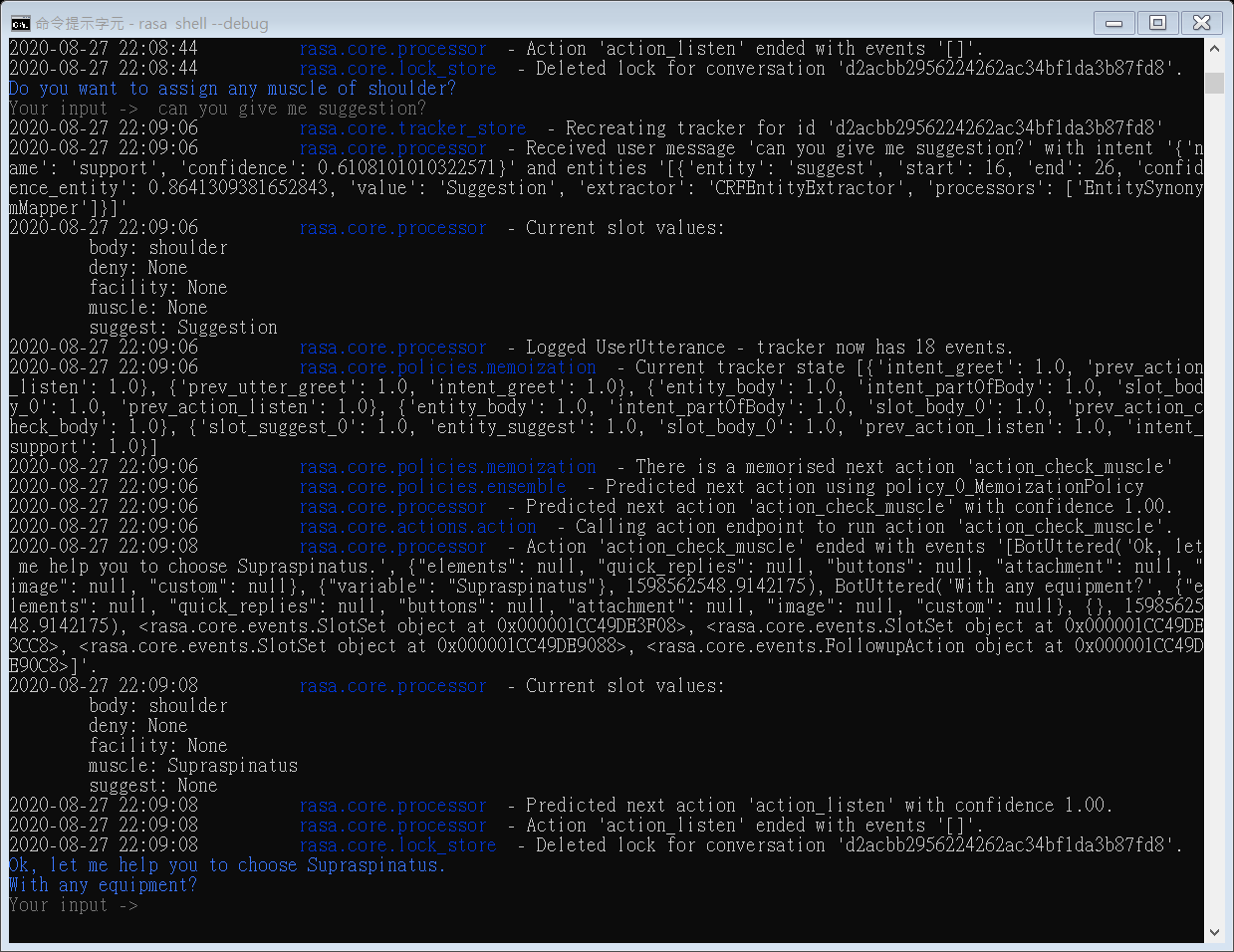
**(N)**

**(M)**

Figure 26

**Recommendation**

Figure 27 shows that the user asks chatbot provides the suggestion (O). The chatbot gives the specific muscle in shoulders (P).



**(P)**

**(O)**

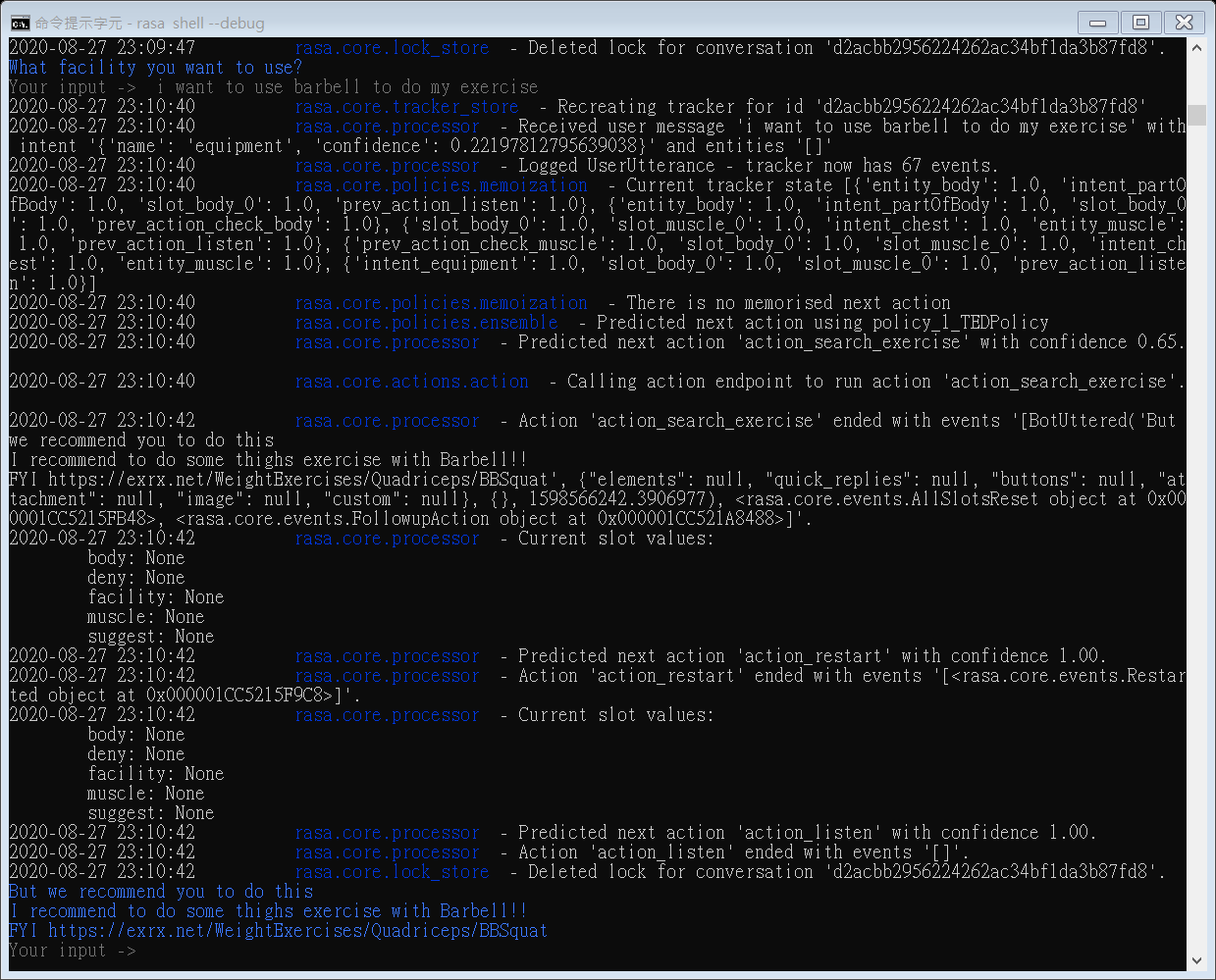
Figure 27

## #Question3 – What equipment

In this section, the chatbot will check what kinds of equipment that the user wants to use to train.

**Get equipment**

The user tells the equipment to the chatbot in Figure 28 – (Q). After the checking, the chatbot reply to a relevant result to the user.



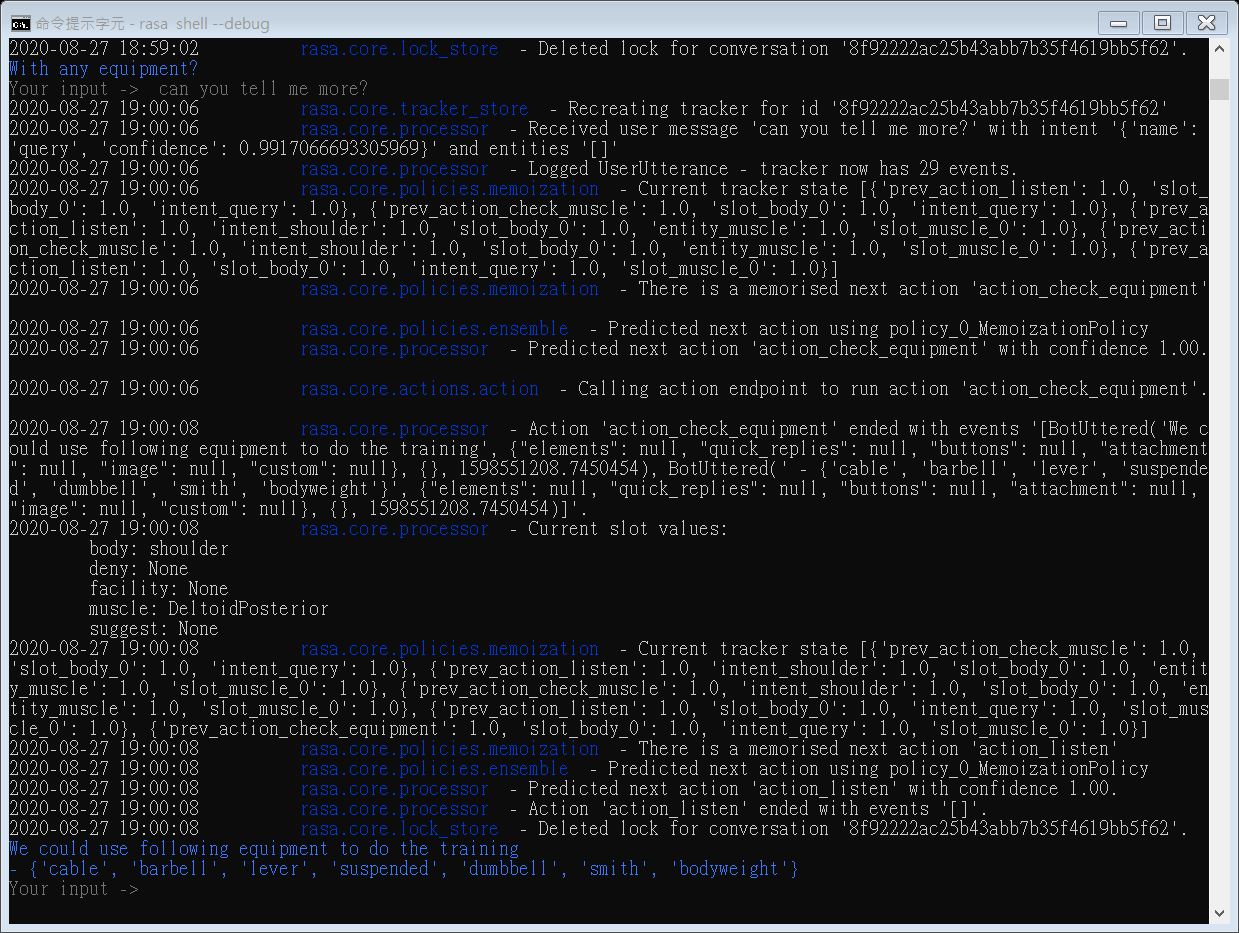
**(R)**

**(Q)**

Figure 28

**Need detail information**

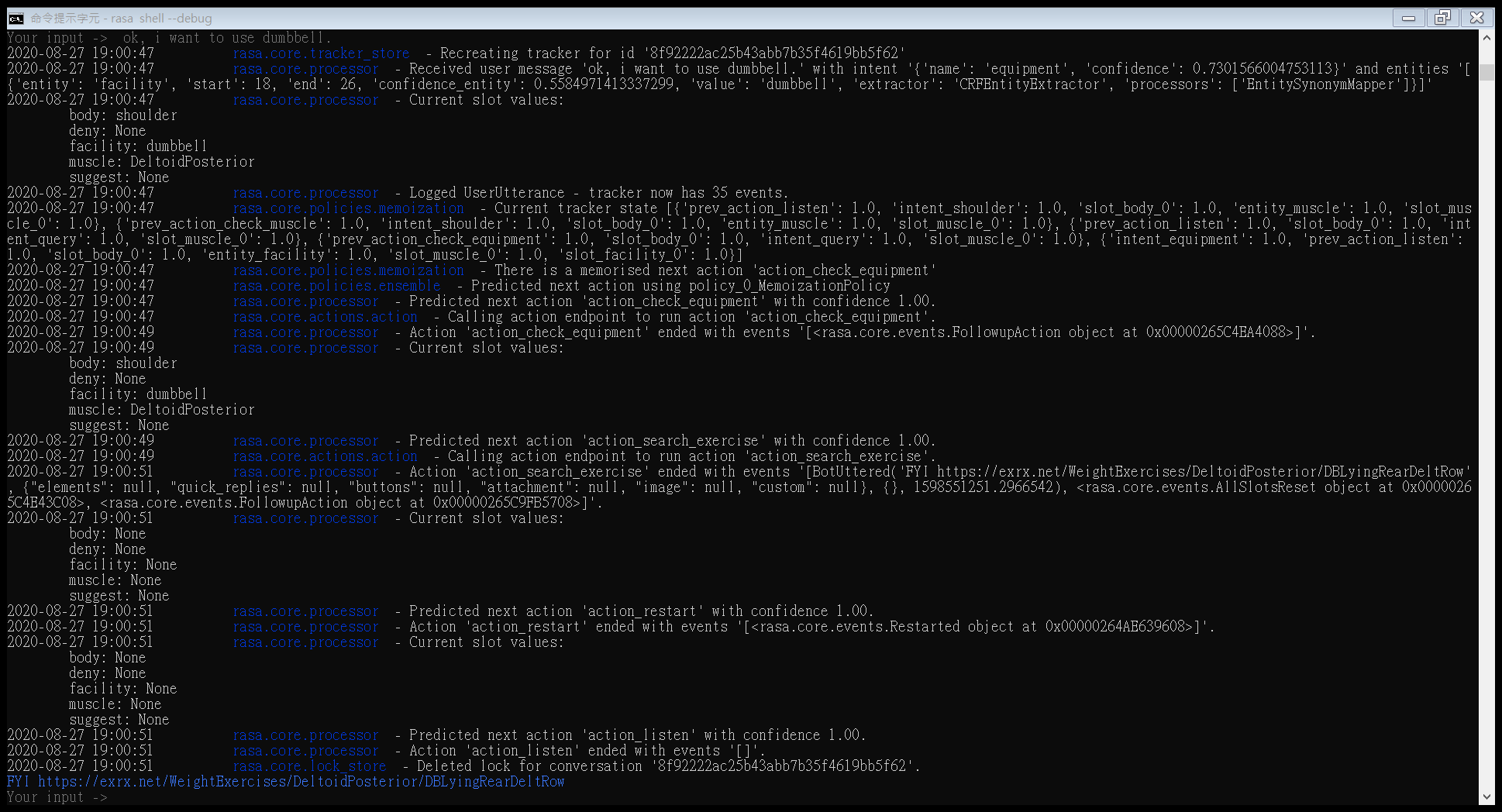
In figure 29, the user has no idea what should choose. So, ask for detail information (S). And chatbot shows the relevant facilities (T) and waits for the user’s feedback. Afterwards, in figure 30, the user chooses one of the facilities (U). Then the chatbot according to these answers to return an exercise (V).



**(T)**

**(S)**

Figure 29



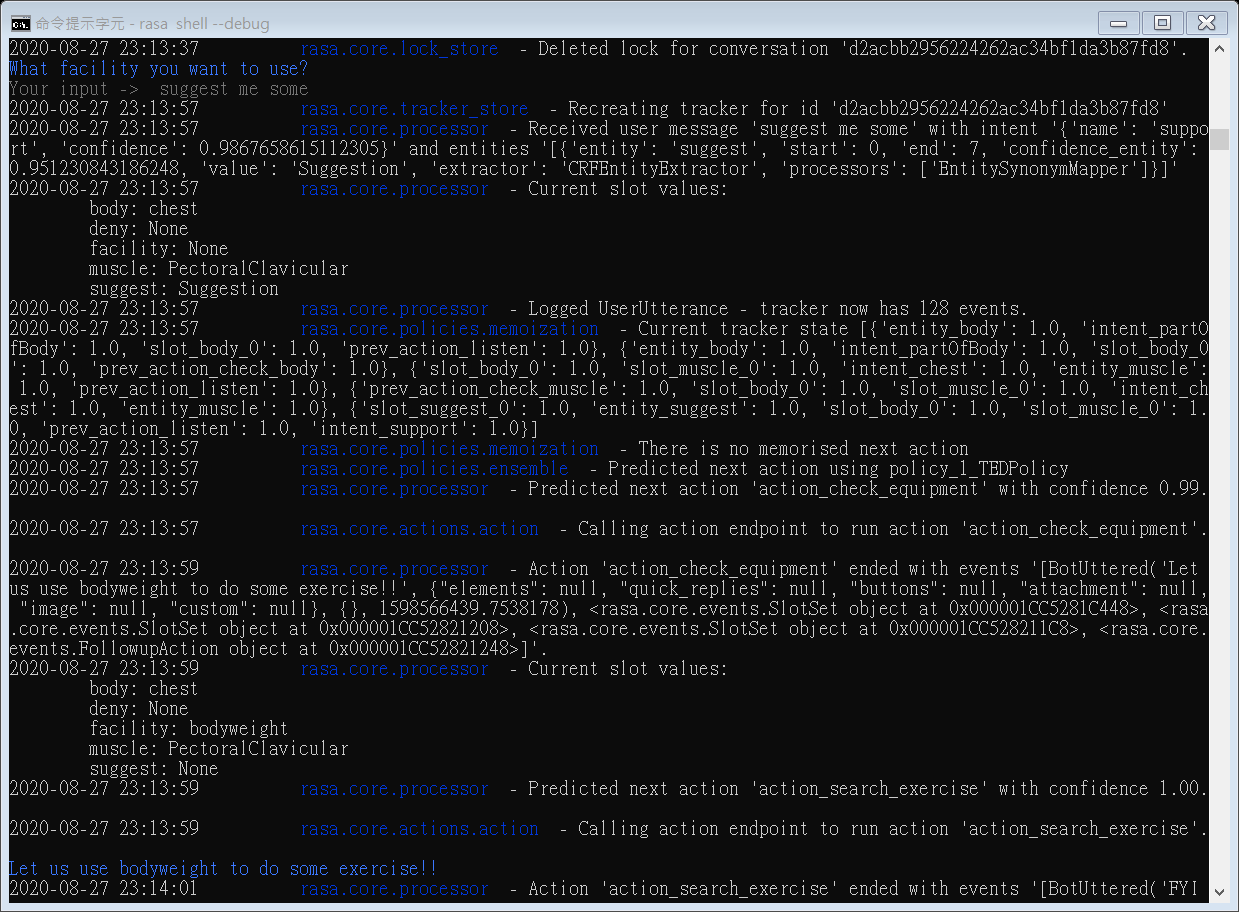
**(V)**

**(U)**

Figure 30

Recommendation

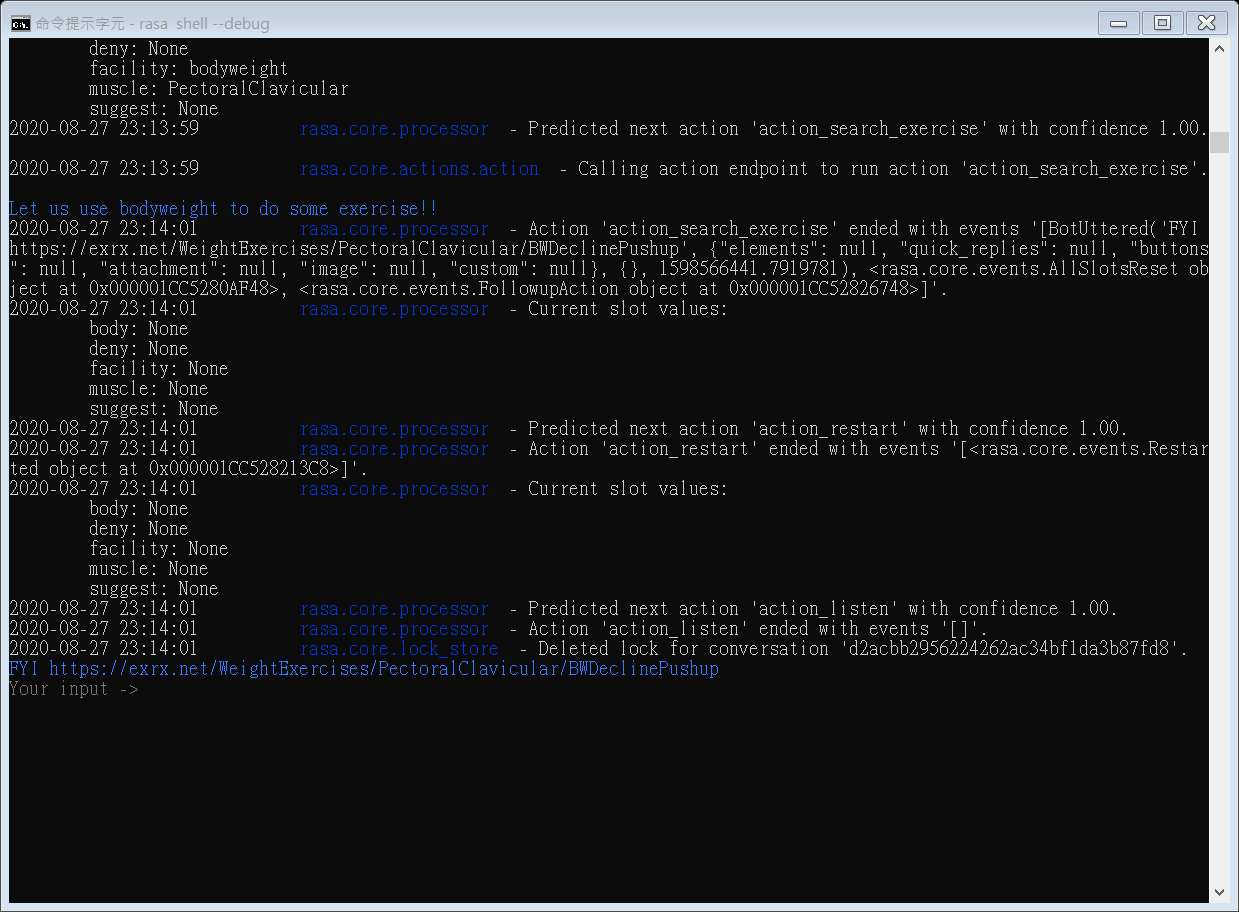
Figure 31 and figure 32 shows that the user needs the suggestion for equipment (W). The chatbot returns the facility (X) and the final result (Y).



**(X)**

**(W)**

Figure 31



**(Y)**

Figure 32

## With User Interface – Telegram

Rasa provides the integrated feature to publish the chatbot to public social media, such as Facebook Message, Telegram, etc. As figure 33, follow Rasa instruction to create a local web host.

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

Figure 33

In figure 34, based on the Rasa and telegram official setting to get the token and register the bot. And the webhook\_url is using Ngrok which allows displaying a web server working on the local device to the internet.

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

Figure 34

Figure 35 is Ngrok which presents the local machine to the internet.

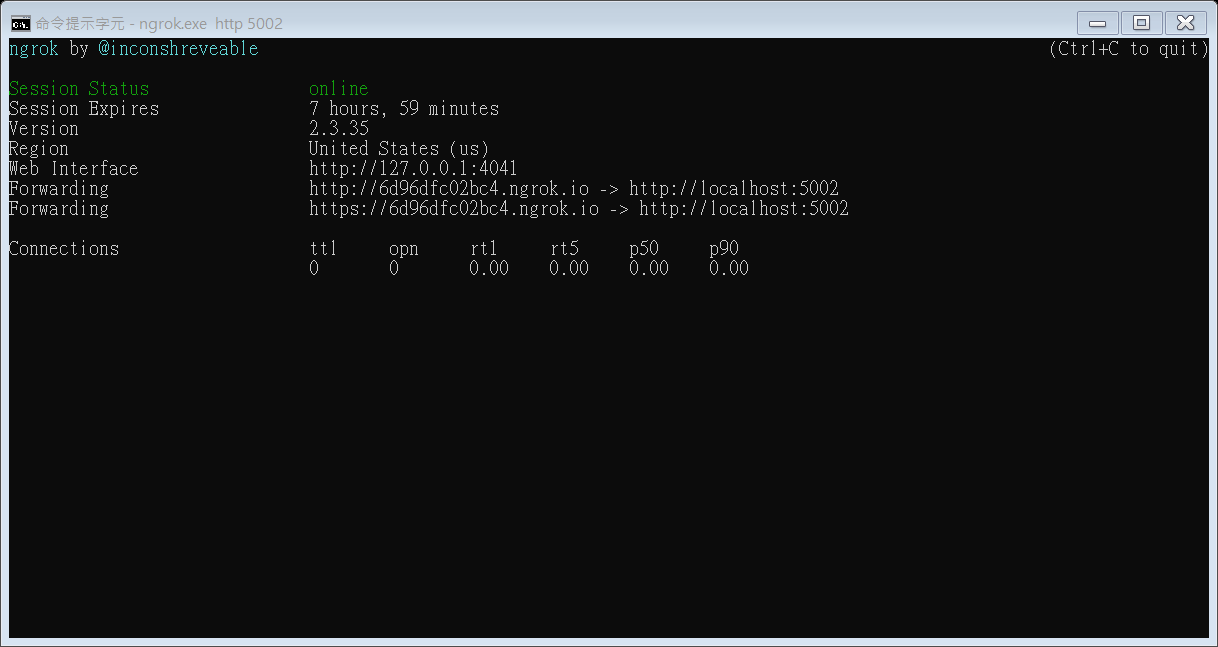


Figure 35

Figure 36 is using Rasa "run" command to let the chatbot start to run the action. The command is rasa run actions --actions actions

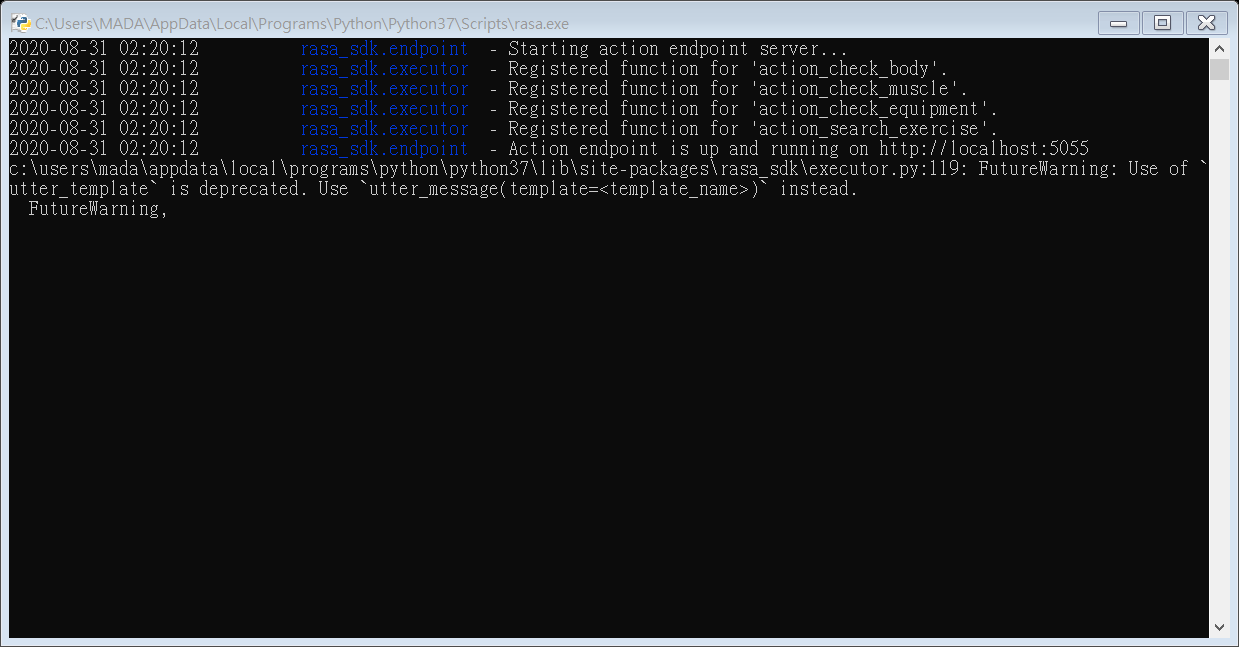


Figure 36

Figure 37 is using Rasa "run" command to let the chatbot based on credentials.yml and connect to the server and telegram. The command is rasa run --port 5002 -m models --credentials credentials.yml --enable-api --log-file out.log --endpoints endpoints.yml

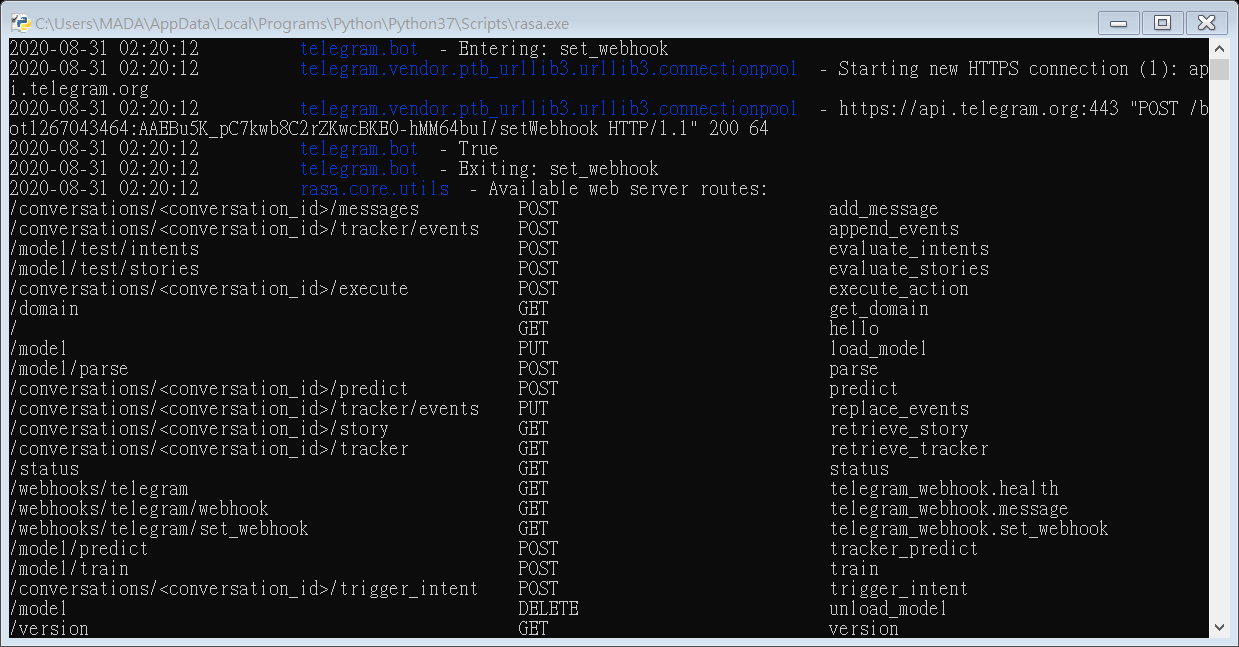


Figure 37

Figure 38 and 39 are the execution result on the Telegram.

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

Figure 38

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

Figure 39

## Evaluation

Our results demonstrated that the chatbot essential functions. It works very well in the happy path which is users giving the utterance without any uncertain meaning.

However, suppose users use a different way to ask the chatbot question which did not exist in the training dataset or paraphrase the problem description even with the same meaning. In that case, the chatbot may return the wrong intent or entity.

Additionally, the chatbot only understands the formal name of muscles; for example, Latissimus dorsi is the part of the back muscle. But most people are saying "lats" in general. If these general names did not exist in NLU dataset, the chatbot could not recognise what the muscle is. In this project, we have already added some of them. But there still have some names we may not know. In the future, we should append those names to make this project complete.

Furthermore, in sport science, some training is not only using only one muscle. For example, doing squat will use the hips and thigh. But in this project dataset, this exercise is in the category of hips. Because without the professional people's help and data resource, we cannot make the dataset perfect. In the future, we should consult the sports science filed people to help us build the training dataset to make it better.

In conclusion, this exercise searching chatbot can finish the fundamental question-answering task, but it still has many aspects that could improve. One is the HCI part, for leading the user gives proper input message and the analysis of the synonym and semantic meaning. Another is sport exercise, need the professional people to help us to build the exercise dataset. Also, we need to know the knowledge of the muscle names to let the user can base their experience to ask the chatbot question. Further studies should investigate these mentioned parts to make this project better.

# Conclusion

**What I learned**

During this process of doing the dissertation, I learned a lot, not only new knowledge but also a new programming language. Such as what is Natural Language Process, chatbot tech, Python, and so on. This knowledge just a little part of the Artificial Intelligence field, because we have to finish the dissertation in a short period. The understanding of the chatbot is not enough. Therefore, I still need to enhance the background and experience to get used to explaining the whole chatbot flow.

Besides, I learned the python skill on programming the chatbot and web crawler. For example, how to import the open resource, how to use for loop, array, list and dictionary to use these skills to make the dataset from the exercise website and build the chatbot actions.

**Difficulty and Limitation**

At the project beginning, this project's aim was helping people figure out how to resolve their workout questions. However, I cannot find a proper dataset which is a question-answering base and tried to ask on the StackOverflow and Reddit. But it did not get any replies. So, I changed to find the exercise data to be the dataset and modified the plan to let people use this chatbot to find out what the exercise they want to know. This situation cost me some time to decide the new goal and find new dataset.

Additionally, I am a conventional computer science student, and previous major is Electrical Engineer. The artificial intelligence is a new thing for me, and I spent much time to find the literature resource and analyse what the best way to finish this project is. Nonetheless, only around three months can complete this project. The time does not permit me to know every detail about NPU, NLP and some other technical knowledge further.

Further, bodybuilding is a highly professional subject. In this project, the user only can answer the basic questions because of the dataset composed of specific muscle and one corresponding exercise. This chatbot does not include the terms of the antagonist, synergy muscle and generic names.

**Summary**

Overall, our results demonstrate a powerful task-oriented chatbot. It can base on the user's message to return the proper answer. Through understanding the categories of the chatbot to know the task-oriented chatbot is aiming to support people fix their problem. Also, using Rasa to help us build the three main strategies for processing the user question. Rasa provides the NLU to train the chatbot to apprehend the sentence meaning and classify the intent and entity. Besides, the training data is coming from the web crawler to download and organise to the CSV file and nlu.md. Furthermore, using Rasa's pipeline, policy and stories to do the correct action when the user inputs the texts. In the end, Rasa and Telegram provide the interface, tokens, and running the command to connect each other and publish the chatbot to the internet.

Although the chatbot performance very well, there still have some drawbacks to improve. Such as recognise generic muscle names and identify synonym and semantic meaning. If there has more time to complete the project, it could be better and more powerful.

# Reference

* Mnasri, M. (2019) *Recent advances in conversational NLP : Towards the standardization of Chatbot building*.
* Daniel, J. andMartin, J. H. (2018) *Speech and Language Processing*.
* Weizenbaum, J. (1966) *ELIZA-A Computer Program For the Study of Natural Language Communication Between Man and Machine*, Communications of the ACM.
* Wallace, R. S. (2003) *The Elements of AIML Style*
* Shum, H.-Y., He, X.-D. andLi, D. (2018) *‘From Eliza to XiaoIce: challenges and opportunities with social chatbots’*, Front Inform Technol Electron Eng, 19(1), p. 10. doi: 10.1631/FITEE.1700826.
* Zhou, L. et al. (2020) *‘The design and implementation of xiaoice, an empathetic social chatbot’*, Computational Linguistics, 46(1), pp. 53–93. doi: 10.1162/COLI\_a\_00368.
* Sutton, Richard S., Doina Precup, and Satinder P. Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 112(1–2):181–211. [An earlier version appeared as Technical Report 98–74, Department of Computer Science, University of Massachusetts, Amherst, MA 01003.]
* Hussain, S., Ameri Sianaki, O. and Ababneh, N. (2019) *‘A Survey on Conversational Agents/Chatbots Classification and Design Techniques’*, Advances in Intelligent Systems and Computing, 927, pp. 946–956. doi: 10.1007/978-3-030-15035-8\_93.
* Bobrow DG, Kaplan RM, Kay M, Norman DA, Thompson H, and Winograd T. Gus, *a frame-driven dialog system*. Artificial Intelligence, Vol. 8(2), 1977, pp. 155-173.
* Handoyo, E. et al. (2018) ‘*Ticketing Chatbot Service using Serverless NLP Technology’*, in Proceedings - 2018 5th International Conference on Information Technology, Computer and Electrical Engineering, ICITACEE 2018. Institute of Electrical and Electronics Engineers Inc., pp. 325–330. doi: 10.1109/ICITACEE.2018.8576921.
* Dahlgren et al. (1997) *Natural language understanding system*
* Bocklisch, T. et al. (2017) *Rasa: Open Source Language Understanding and Dialogue Management*.

# Appendices

To run the project application, all project source code can be found on the GitLab repository using the below link:

<https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2019/txm986>

Chatbot source code is under /ChatBot\_NLU

Web Crawler source code is under /ExerciseData

All source code is with Python 3.7.7.

## Appendix A: Chatbot

**Environment:**

1. The chatbot source code can be found at /ChatBot\_NLU
2. Download Python 3.7.7
3. Install the requirement by command: pip install -r “requirement.txt”

**Training Data**

The server can be configured in “endpoints.yml”. This is for training and running the dialogue management model.

1. Open the terminal and run the custom action server command:

rasa run action

1. Open a new terminal and train the rasa core model:

rasa train

**Interactive Mode**

Using this mode to help you to train the action process.

1. Open the terminal and make sure action server is running

rasa run action

1. Open a new terminal and run the interactive mode session:

rasa interactive

**Execute the chatbot on Rasa shell**

1. Open the terminal and run the custom action server command:

rasa run action

1. Open a new terminal and load the Rasa shell

rasa shell

## Appendix B: Telegram

Register a Telegram account and follow the official document and instruction to create a public chatbot service. It will provide the access token and you can set the chatbot’s name.

**Telegram official document:**

<https://core.telegram.org/bots>

<https://core.telegram.org/bots/api>

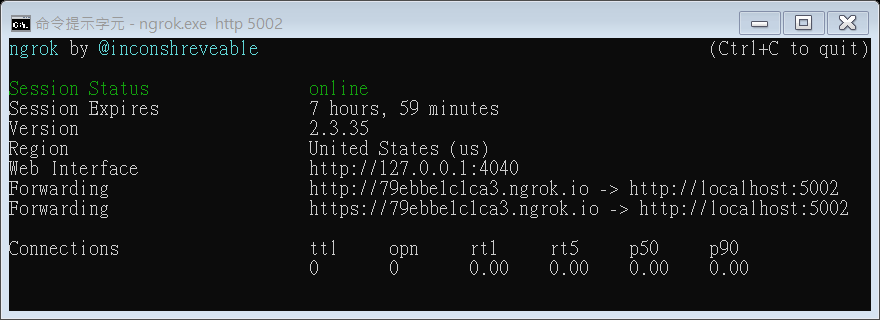
**Bot Father:**

<https://web.telegram.org/#/im?p=@BotFather>

## Appendix C: Publish to Telegram

You can use “ngrok” to help you do this: <https://ngrok.com/>

Run the command: ngrok http 5002, and you will get the below information.



Modify the “credentials.yml” and follow Rasa official setting:

<https://rasa.com/docs/rasa/user-guide/connectors/telegram/>

一張含有 鳥 的圖片

自動產生的描述

Access\_token is the token when registering the Bot Father.

Verify: This is the chatbot name when create the chatbot on Telegram.

Webhook\_url: The “your\_url.com” is filling the HTTP to <https://79ebbe1c1ca3.ngrok.io>

After finishing with above setting, we can start to publish our chatbot to Telegram.

1. The ngrok cannot close, should keep open.
2. Open the terminal and make sure action server is running

rasa run actions --actions actions

1. Open a new terminal and run the setting as the below.

rasa run --port 5002 -m models --credentials credentials.yml --enable-api --log-file out.log --endpoints endpoints.yml

## Appendix D: Web Crawler

The Web Crawler files are at /ExerciseData

Run python command to install Beautiful Soup:

pip install BeautifulSoup4

And compiler the crawler.py to get the csv file:

python crawler.py

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)
4. <https://www.uctoday.com/collaboration/what-is-natural-language-understanding/> [↑](#footnote-ref-4)
5. <https://en.wikipedia.org/wiki/Natural-language_understanding> [↑](#footnote-ref-5)
6. <https://blog.rasa.com/nlp-vs-nlu-whats-the-difference/> [↑](#footnote-ref-6)
7. <https://rasa.com/> [↑](#footnote-ref-7)
8. <https://www.ideas2it.com/blogs/battle-of-the-bots-rasa-vs-google-dialogflow-vs-aws-lex/> [↑](#footnote-ref-8)
9. <https://chatbotslife.com/dialogflow-vs-rasa-which-one-to-choose-206fb98b0e90> [↑](#footnote-ref-9)
10. <https://medium.com/@CobusGreyling/a-comparison-of-eight-chatbot-environments-41ad3b795dc7> [↑](#footnote-ref-10)
11. <https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/> [↑](#footnote-ref-11)
12. <https://rasa.com/docs/rasa/core/stories/> [↑](#footnote-ref-12)
13. <https://blog.rasa.com/designing-rasa-training-stories/> [↑](#footnote-ref-13)
14. <https://rasa.com/docs/rasa/core/interactive-learning/> [↑](#footnote-ref-14)
15. <https://rasa.com/docs/rasa/user-guide/connectors/telegram/#id1> [↑](#footnote-ref-15)
16. <https://core.telegram.org/bots> [↑](#footnote-ref-16)
17. <https://en.wikipedia.org/wiki/Web_crawler> [↑](#footnote-ref-17)
18. <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> [↑](#footnote-ref-18)