To-do List Chatbot

Prepared by

Yuxin Lin

Electrical Engineering

University of Waterloo

January 13th, 2020

Table of Contents

1 Introduction

There are many chatbots out there. Why are chatbots useful? What makes this chatbot special? Where did the idea come from? What technology will be used? All these questions will be answered in this section.

1.1 Background

As the technologies of machine learning and deep learning develop, Natural Language Processing (NLP) is affecting people's everyday lives. NLP is the analysis and processing of text. Some common use cases of NLP are machine translation, summarization, entity extraction, sentiment analysis, hate speech detection, and chatbot. In this project, we are building a chatbot. There are many chatbots on the commercial market such as Amazon Echo, Google Home, and Apple's Siri. To build a chatbot, we first need to understand what the user wants by determining the user's intent. Then based on the intent, we extract valuable information or entities from the conversation to take action for the user. In most of the cases, one message is not enough: the chatbot needs appropriate context, asks for missing information, and guides the user through the process.

1.2 Inspiration

In this project, the robot built is referred to as Hazel. I have two cats: Hazel and Siri. Since Apple has already coined the name Siri, I will take on the challenge and name my chatbot Hazel, hoping that my chatbot will one day become better than Apple's Siri. Hazel is designed as a personal task manager and reminder. The users can create, read, update, and delete their upcoming events by telling Hazel what to do with WeChat, and Hazel will be able to save, recall, or delete the information based on their intents.

2 Technologies Involved

We use several tools to achieve the functions and improve the performance. In the second section, we are going to discuss the functionalities of these tools and how they are implemented in this project.

2.1 Rasa NLU and SpaCy

Rasa NLU is an open-source natural language processing tool for intent classification and entity extraction in chatbots.[1] When using Rasa NLU, the income messages are processed by a sequence of components, which is called processing pipeline. One of the most important processing pipelines is spacy_sklearn. Spacy_sklearn pipeline uses pre-trained word vectors from either GloVe or fastText, and it is especially useful when there are not much training data.[2]

SpaCy is an open-source library for natural language processing. The raw text can be translated by spaCy to a Doc object, that comes with a variety of annotations. SpaCy features an entity recognition system. The default model identifies a variety of named and numeric entities, including companies, locations, organizations, and products. The users are also allowed to add arbitrary classes to the entity recognition system, and update the model with new examples. SpaCy also supports many other languages other than English, which may be useful as the project scales.[3] More details will be discussed in section 2.4.

2.2 Regular Expression

Regular expression is a pre-defined pattern such as ([1-9][0-9]*). This pattern matches with all number that does not lead with a zero. For example, 987 will be matched because 987 is a number that does not start with zero; however 023 will not be matched since it starts with a zero. Using Python, we can match a user's message with this pattern to find all numbers without a

leading zero. Regular expressions are simple to use, robust and helpful in our project, which will be discussed in more detail in Section 2.4.

2.3 SQLite

SQLite 3 is a lite version of a Structured Query Language (SQL) database. It is easy to use and great for project prototyping. Sqlite3 also has great support for Python, which will be discussed in more details in Section 2.4. SQLite 3 supports select with sorting, delete by id, read by id, update by id, and many more features.

2.4 Utilization

To determine the user's intent, we are using Rasa. We configure Rasa to use Scikit-Learn under the hood. Scikit-learn then uses Support Vector Machine (SVM) to train a classifier to determine the user's intent. [2] In terms of training data, we defined various intents with sample messages that align with the specific intent. For example, "Create a new task", "Make a new reminder", and "Remind me to do my homework" are all examples of intent to create an entry in the database.

With chatbots this sophisticated, we need multiple exchanges of messages to accomplish one job. Therefore the chatbot needs to save the information it gathers through the conversation until the user confirms to finalize the action. Therefore, there could be intents that do not make sense in the context. For example, a user cannot delete Note B when the user is in the middle of creating Note A. There are also valid intents such as the user gives an affirmative response when the chatbot asks the user to confirm the details. With this confirmation example, the user could say no and modify the details.

If the intent makes sense in the context, then the chatbot needs to extract valuable information from the message provided by the user. There are various ways to extract information from messages. One way is to use regular expressions (regex). When the user says “Show me more on task 1”, we first determine the intent is to read more about a task, then use regex to parse the index of the task. Now we fetch from the database and present the details of the requested task to the user. Another way to extract information from messages is to use Spacy to extract entities such as date, time, person, location, event, and organization. [3] For example, when the user says “I am having dinner with David at 3 pm at the KFC in New York”. Then Spacy tells us 3 pm is the time, David is a person, location is in New York, the event is having dinner, and the organization evolved is KFC. Another way to extract entities is to use Rasa. When defining the intent with example messages, we can also define the entities in the message. For example, “View Task [1](index)” shows the intent to read more, but it also says that number 1 is an entity named index. Rasa can tell us the intent and the entities extracted via interpreter parsing. [4]

Furthermore, we use sqlite3 as our database. We created a table called tasks. In the task table, we have columns such as date, location, people, description, and details. For each task, there is usually date or time involved. For example, the user might say “I need to finish my report before tomorrow at 3 pm”. We extract the entities involving date and time. Then we use python date parser library to convert “tomorrow 3 pm” to a python datetime object such as “datetime.datetime(2019, 1, 13, 10, 1, 30, 992870)”. Sqlite3 is extremely easy to use and is a useful asset in our application. [5]

All chatbots eventually need a user interface to connect with the users. In terms of the user interface, we decided to use WeChat due to its popularity and ease of access. WeChat has a simple user interface to ensure users are focused on the conversation. WeChat is also performant and easily

accessible by users. We utilize a python library called wxpy, which stands for WeChat Python. With Wechat Python, we can subscribe a listener to a friend or many friends. When a message is received, we pass this message to our main chatbot program. Then send the response back to the user.

3 Project Features & Demo

Based on all the tools and technologies we discussed in section 2, the features of the project will be introduced here. Hazel has 4 basic functions: create a new task, read the existing tasks, update the tasks, and delete the tasks. There will be a short demo of each function by the end of the subsections.

3.1 Create

When the users are in the initial state, they can either create or read their reminders. To create a new one, the user should give Hazel a command such as: “create a new reminder” or “make a task”. As long as Hazel indicate the intent of the user as “create_start”, it will move into the create flow and ask for a brief description from the user. After the user provides the description, the description will be saved in the database as a string. Then, Hazel will ask the user to provide more information about the task. Similarly, the details will also be saved for review. The content of description and details will be concatenated together for entity abstraction. Locations, people, and times/deadlines will be abstracted and saved separately. Hazel will send a summary of the reminder to the user for confirmation and then finish the create flow.

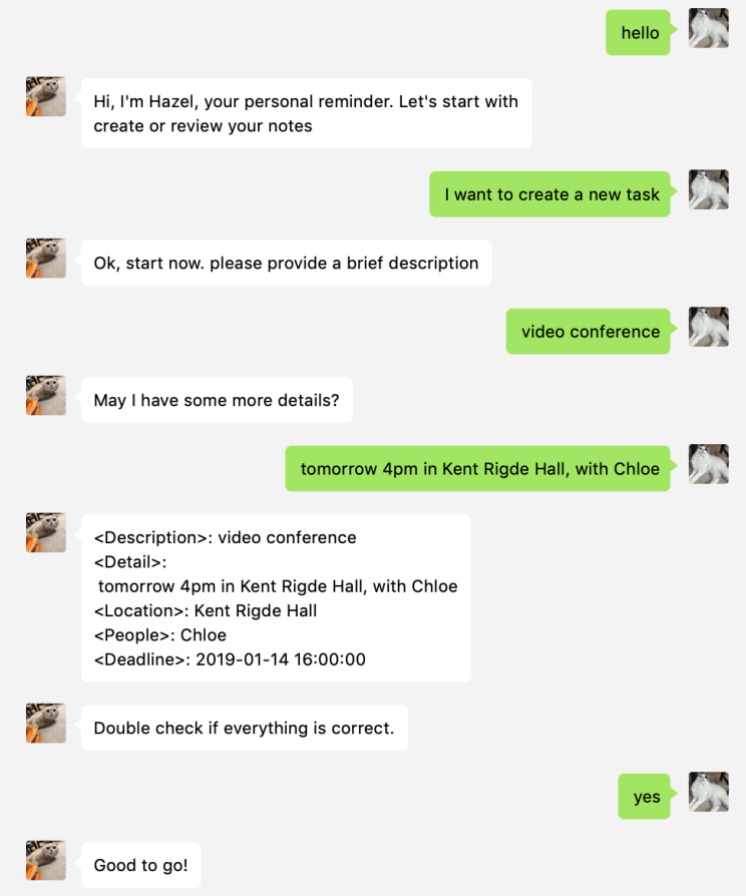

Figure 1: Create Flow Example

3.2 Read

To review the tasks, the users should give a command such as “show my tasks”. Hazel will be able to understand the intent as “read” and list all the tasks’ descriptions in the descending order of deadline. Also, an index will be assigned to each of the tasks. To read more details of a specified task, the user can simply call the index to open the summary. If a non-existing index is called, Hazel will send an error message to the user.

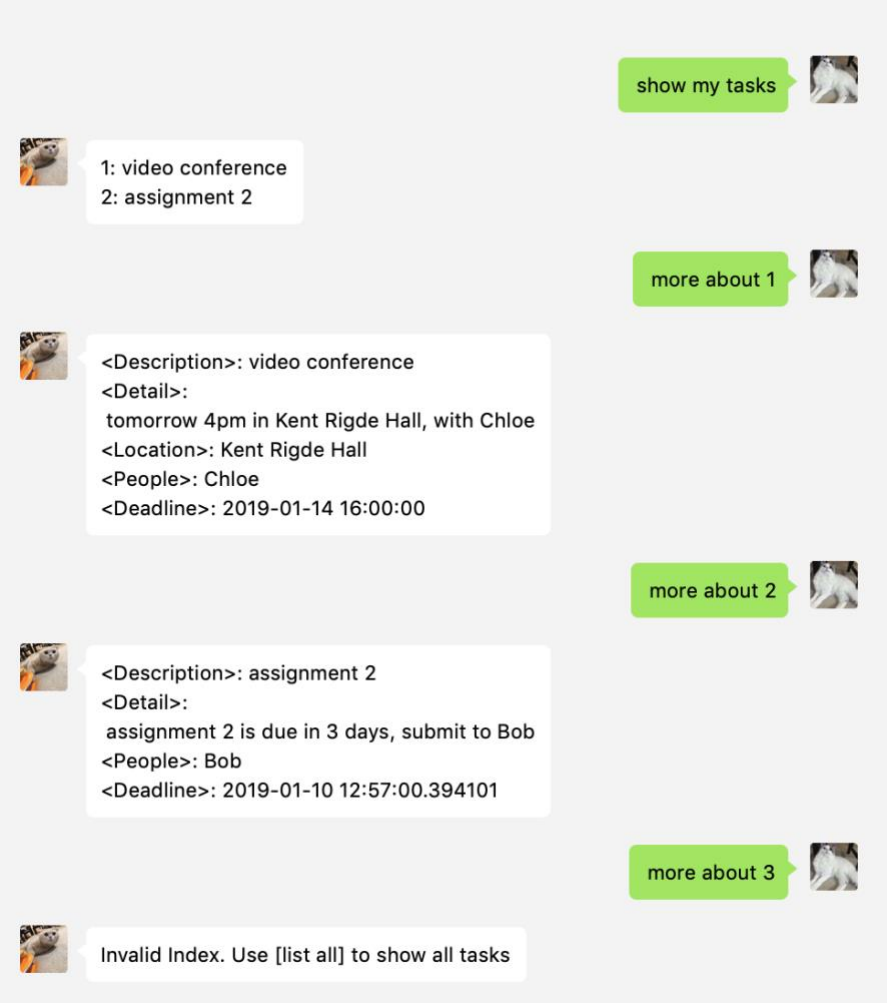

Figure 2: Read Flow Example 1

Users can also choose to review the tasks due on a specified date. For example, the user can use the command “show me my tasks today”. All the tasks with the deadlines on that date will be print out. If there are no such tasks exist, Hazel will suggest the users to have a look at other upcoming events with deadlines, and the users can decide if they want to read those events. This add some intelligence to the chatbot that it forwardly offers some suggestions to the users.

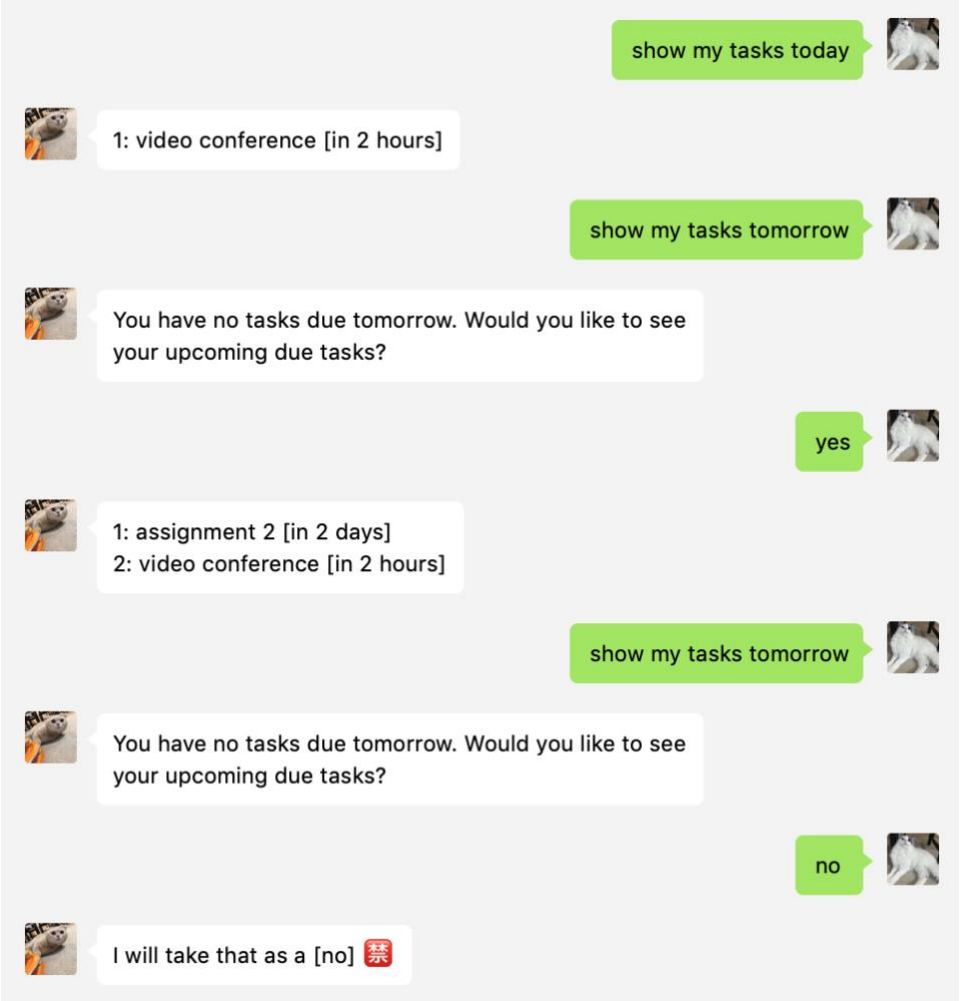

Figure 3: Read Flow Example 2

3.3 Update

The saved tasks can be updated at any time. To modify an event, the user should be at the state of reading the particular event. When the intent is classified as “update”, Hazel will ask for the content of the update. The users are not required to repeat the whole create progress. Instead, they can only type in the new information to update or change. Hazel will be responsible for abstracting the new entities from the updated information and modified the tasks. There will be an update history generated in the detail section.

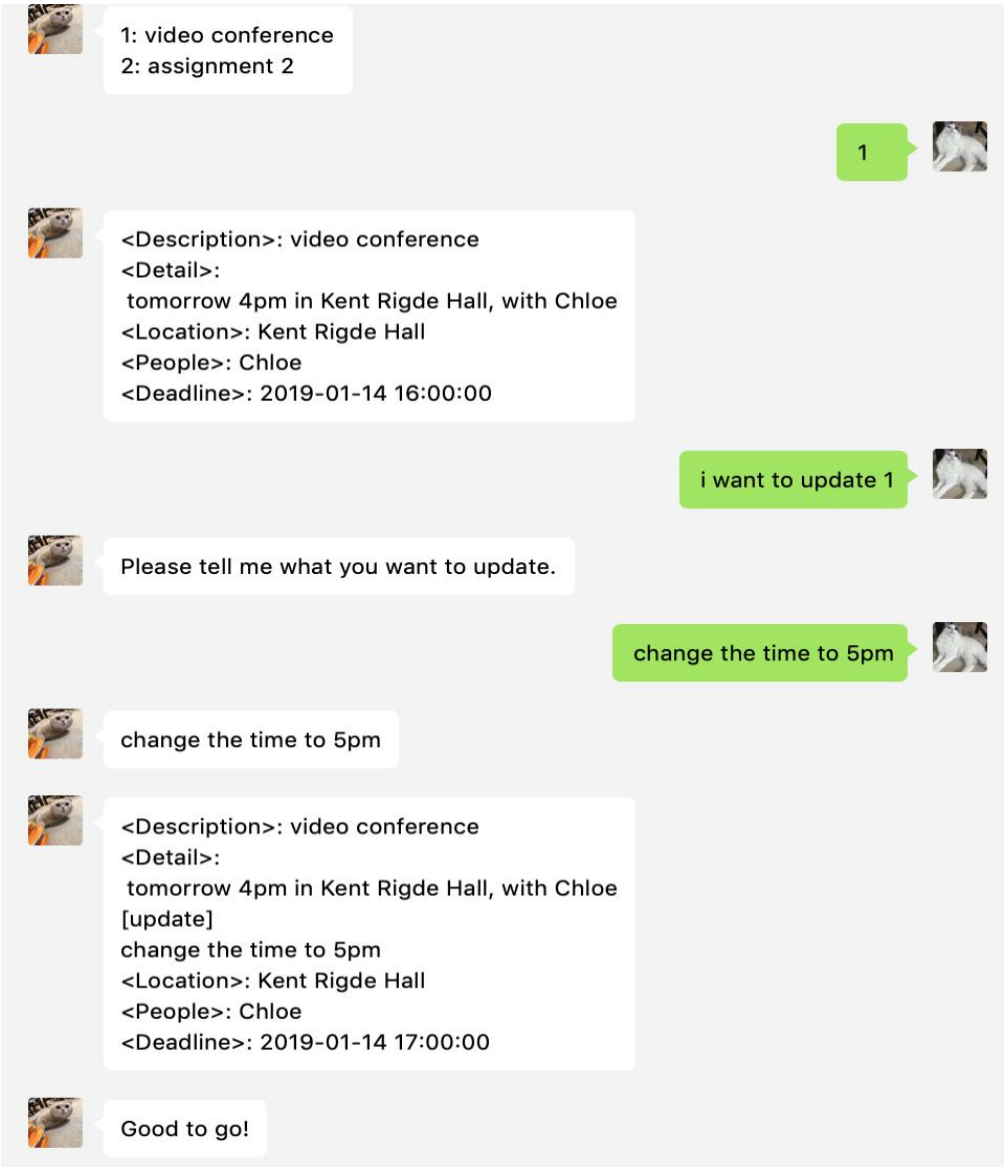

Figure 4: Update Flow Example

3.4 Delete

Users can choose to delete the tasks which are completed or no longer useful. To delete a task, the user should be in the state of reading and use the index of a particular task. For example, the user can type in “delete 1” to delete the first task of the todo list.

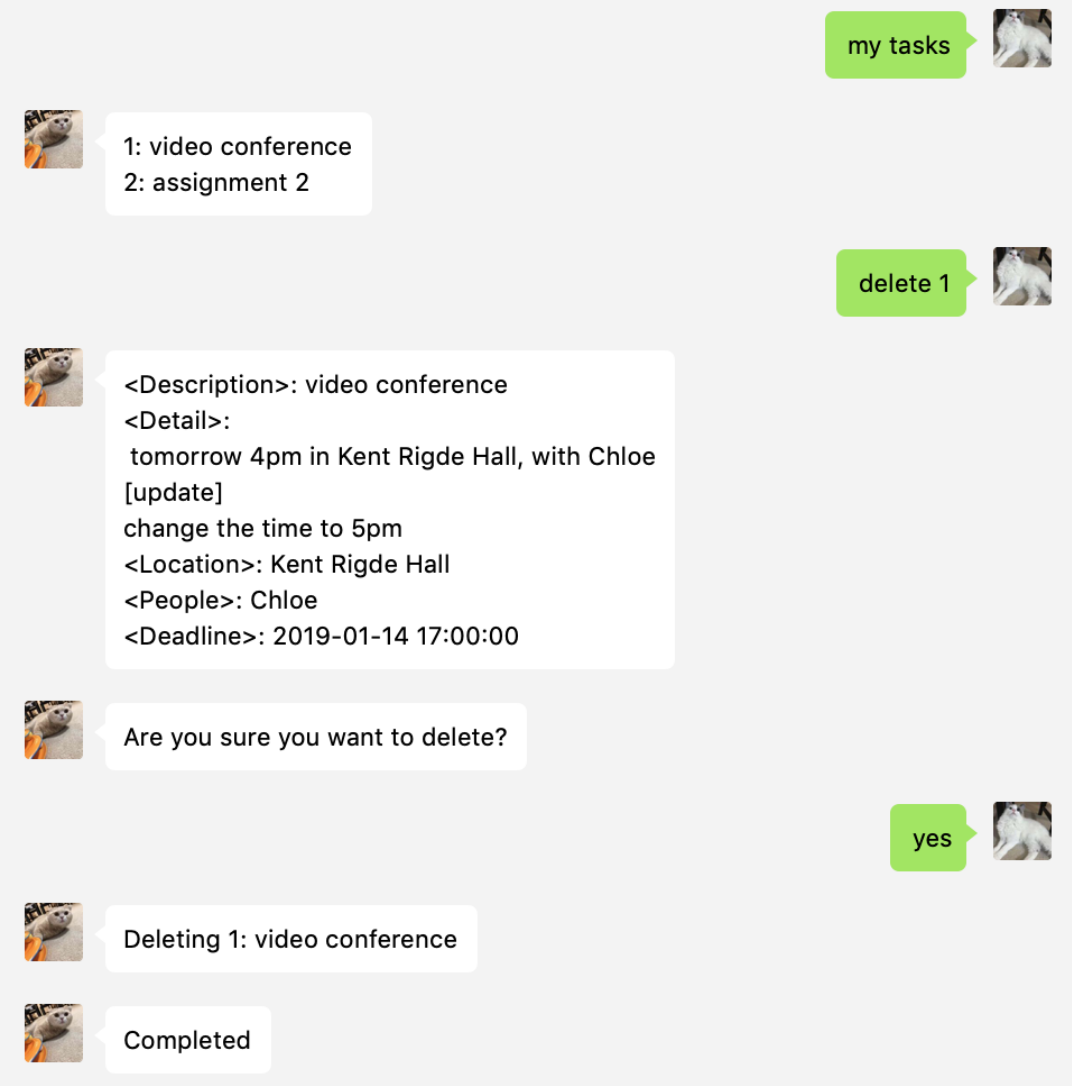

Figure 5: Delete Flow Example 1

Once a task is deleted, the indexes will be updated accordingly.

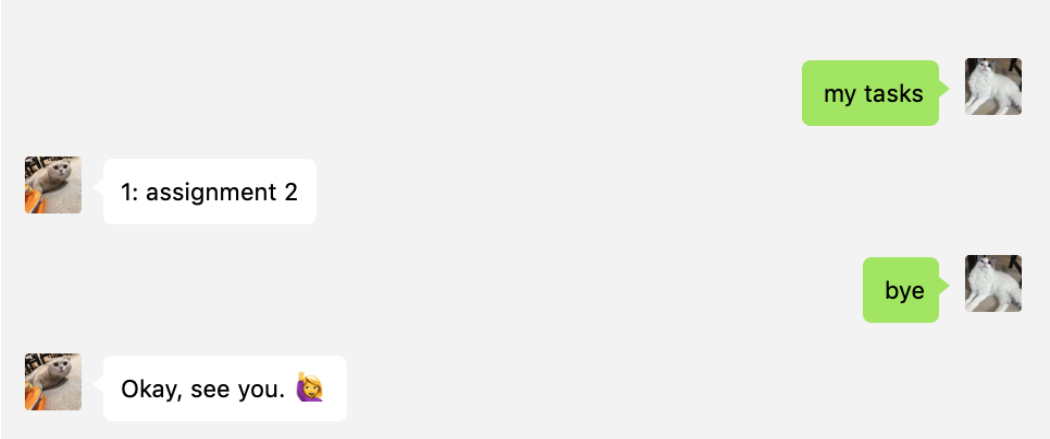

Figure 6: Delete Flow Example 2

4 Summary & Extension

In this section, a summary or reflection of the project will be drawn. Furthermore, some features for future development will be discussed.

4.1 Summary

This project opens a door for me in the field of natural language processing. So many great tools and technologies were introduced to me, which is awesome. Prof. Zhang gave us many great resources on various technologies. I had enough tools to get started and still have the room to explore on my own. Overall, I am proud of what I have accomplished throughout the project. I also feel confident about the future since I also learned how to obtain the resources I need to succeed. I would strongly recommend anyone with similar prior experience and interest to be part of the experience.

4.2 Extension of the project

Building this project is quite a journey. I still see many areas for improvements. From the demo, observe that typing is arduous work and defeats the purpose of making our lives easier. WeChat's biggest feature is audio communication. Therefore, we are considering using API to convert audio to text. Users can speak to Hazel, making the entire process much easier. Wechat Python library supports downloading of audio messages. Then we can utilize Google Audio to Text API [6] or IBM Watson Audio to Text API [7] to obtain text from audio. Once we obtain the message in text, we can feed the text to the main chatbot program similar to what we have at the moment.

Another useful feature to have is a reminder feature. As mentioned before, when the task is associated with a deadline, we can send a message to remind the user that Task A is about to due. For example, the user has a report due this Sunday. Then the chatbot can send a message to the user on 7 days, 3 days and 1 day before the deadline to avoid missing the deadline. This feature requires the chatbot program to run 24/7, therefore requires a server to accomplish this feature. Furthermore, this feature also requires a WeChat account dedicated to this chatbot. Turing Robot is a web service that provides developers with a WeChat account dedicated to chatbot services but charges a fee based on usage. [8]

Reference

[1] https://rasa.com/docs/get_started_step1/

[2] https://rasa.com/docs/nlu/0.12.1/pipeline/

[3] https://spacy.io/usage/linguistic-features

[4] https://rasa.com/docs/nlu/master/entities/

[5] https://docs.python.org/2/library/sqlite3.html

[6] https://cloud.google.com/speech-to-text/

[7] https://www.ibm.com/watson/services/speech-to-text/

[8] http://www.tuling123.com/price/index.jhtml