COMP3074-HAI - Coursework 1: Question Answering Chatbot

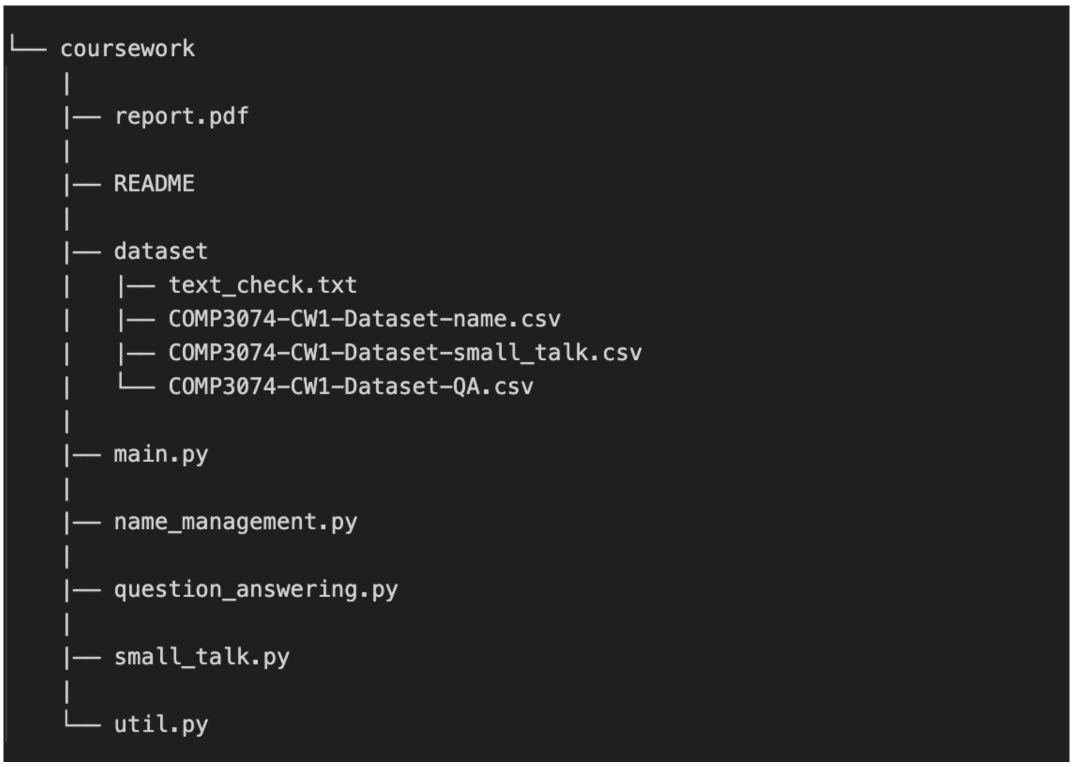
Pinyuan Feng
School of Computer Science
University of Nottingham
scypf1@nottingham.ac.uk
(1722words)

Abstract

This is the first coursework of the module COMP3074 Human-AI Interaction on the topic of Building a simple Question Answering (QA) Chatbot. For the coding part of the coursework, all codes are implemented in Python and some of them are modified or reused based on the lab examples and exercises. In this report, the design of the system is comprehensively delivered, including *intent matching*, *name management*, *small talk,* and *question answering*, followed by a description of some 'smart' functionalities. Next, the evaluation of the system is critically demonstrated based on testing feedback. Lastly, how the system will be improved in the future is further discussed.

System Design

The figure below shows the structure of coding files. Since *name management*, *small talk* and *question answering* are the core functionalities of the chatbot, I implement each functionality in an independent file. For each functionality, I have built 3 corresponding datasets in '.csv' format. And 'main.py' file contains the core logic of the system, which decides how files are called and how those functionalities are integrated. 'util.py' contains some common functions that are repeatedly called and used, such as functions related to text preprocessing.


~~~
└── coursework
    |
    |── report.pdf
    |
    |── README
    |
    |── dataset
    |   |── text_check.txt
    |   |── COMP3074-CW1-Dataset-name.csv
    |   |── COMP3074-CW1-Dataset-small_talk.csv
    |   └── COMP3074-CW1-Dataset-QA.csv
    |
    |── main.py
    |
    |── name_management.py
    |
    |── question_answering.py
    |
    |── small_talk.py
    |
    └── util.py
```]
~~~

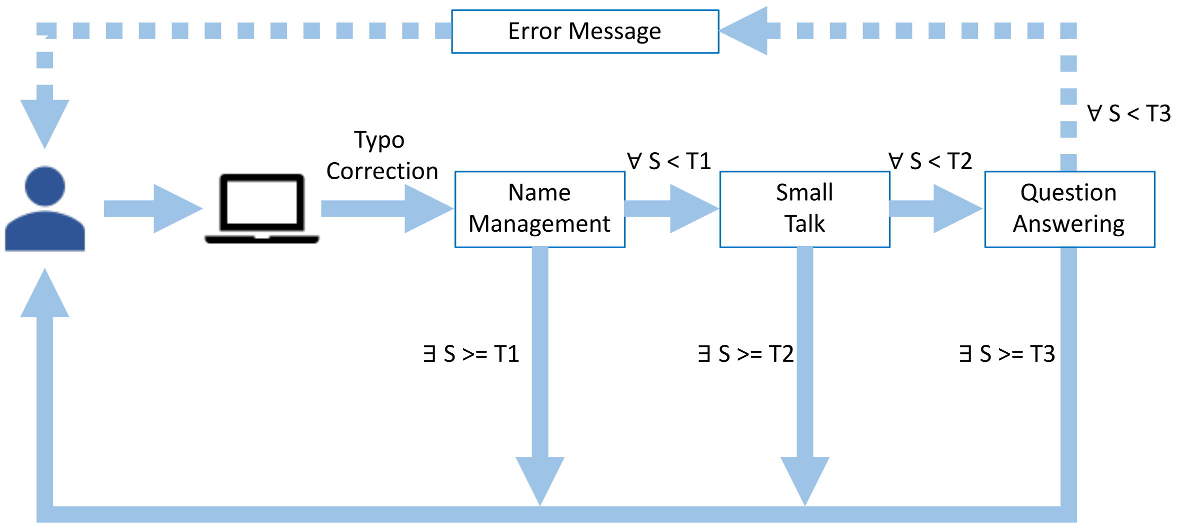
The main reasons why I separate the codes into different files are that 1) it could facilitate unit testing and debugging, where each functionality could be assessed independently; 2) the structure is clearer, which makes the design system easy to understand.

Intent Matching

The idea of intent matching follows the workflow below. Generally, I designed a pipeline to check the user input as well as output the system response. I set priorities for the system to analyze the user input, where the system will check the *name management* in the first place, and then *small talk*, and finally *question answering*. The order is dependent on how much time the task should take. *Name management* just allows the system to handle the name-related questions, while *QA* requires the system to search the answer from thousands of data. Putting *QA* in the first place may cause delays if the user input is actually relevant to the other two intents and thus has a negative impact on the user experience. Besides, the order depends on how specific the task is. The *QA* dataset contains a number of *QA* pairs, while less data is involved in *name management* task because the task is specific and clear. Biases could be largely alleviated when putting *QA* in the end.

Note:

- S means Similarity value
- T means Threshold



The reason why I haven’t designed an intent classifier here are that 1) the datasets are unevenly distributed, which means we’d better treat those intent tasks in different ways instead of using the same classifier; 2) it may take a long time than expected, where query iteratively compare with each question to find the most possible intent and then repeatedly search for the answer based on that intent; 3) it’s not efficient to apply different text-preprocessing techniques to three tasks in the same classifier. For *QA*, we should clean the stop words to find the keywords, while stop word filtering cannot be applied to the rest tasks, otherwise, the whole sentence might be removed (e.g. ‘Who am I?’ in *name management* task).

Question Answering

When inputting a query related to the *QA*, the system tries to find an answer from the dataset. Before searching, I pre-processed the text vector via stemming, because stemming takes less time to achieve the simplified text forms compared with lemmatization. Next, I applied TF-

IDF to the data, which decreases the weight for high-frequency words and increases the weight for less used words in a collection of documents,

$$idf(n,N) = 1 + log_2\left(\frac{N+1}{n+1}\right) \tag{1}$$

$$tf(t,d) = log_2(1 + freq(t.d)) \tag{2}$$

$$tf - idf = idf(n,N) * tf(t,d) \tag{3}$$

To compare the query with the text in the dataset, I applied cosine similarity function,

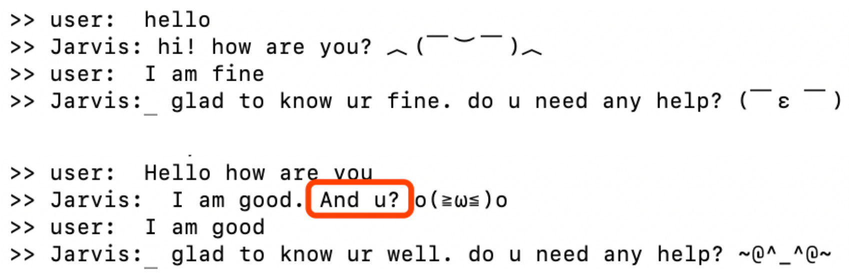
$$similarity = \frac{A\ B}{||A||\ ||B||} \tag{4}$$

Besides, a threshold is set to decide whether the query matches the *QA* data. Otherwise, an error message will be sent to the user. If there is a same question with different answers, the system will randomly pick one from them.

Small Talk

The small talk dataset is built to handle the greeting expressions, such as "How are you?", "Hello", etc. When searching the query, the query and all the text data will be vectorized into TF-IDF format. A threshold is set to evaluate the Cosine similarity. The data is not stemmed or lemmatized here, otherwise, the whole sentence might be removed.

To ensure a natural conversation, I have applied some "tricks" to the dataset. For the same question, different answers are added in the dataset allowing the system to randomly choose one of them. Besides, for some greeting expressions, they are both in "question" and "answer" data column of different *QA* pairs. The system will automatically raise some questions for user to answer and then respond back instead of always requiring the user to ask. In this way, the dataset could make the conversation natural. Let me use 2 examples to make it clear. if the user just says "hello", the system will greet and ask, "how are you?"; if the user greets and then asks, "How are you?", the system will reply to the user and ask back, "And you?".



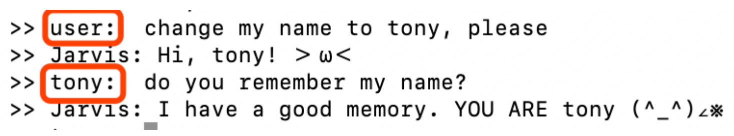
```
>> user:  hello
>> Jarvis: hi! how are you? ︿(￣︶￣)︿
>> user:  I am fine
>> Jarvis:_ glad to know ur fine. do u need any help? (￣ε ￣)

>> user:  Hello how are you
>> Jarvis:  I am good. And u? o(≧ω≦)o
>> user:  I am good
>> Jarvis:_ glad to know ur well. do u need any help? ~@^_^@~
```

Name Management

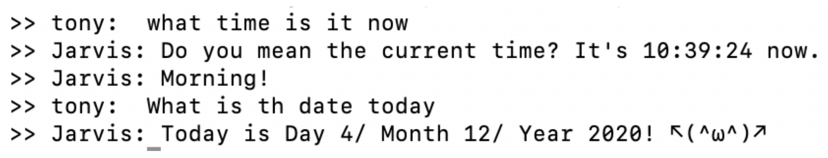
In this task, checking whether the user wants to change his/her username is required. I store several relevant words in a checking list, such as 'change', 'rename', etc. If the user input overlaps with the list, the system could know the user may want to change the username. Then, the system will extract the new name from the user input by filtering the stop words as well as the words in the checking list.

To ensure the system could output the username in some cases, like “Do you know my name?”, I have built a corresponding dataset consisting of similar questions. The query retrieval approach is the same as that of small talk.



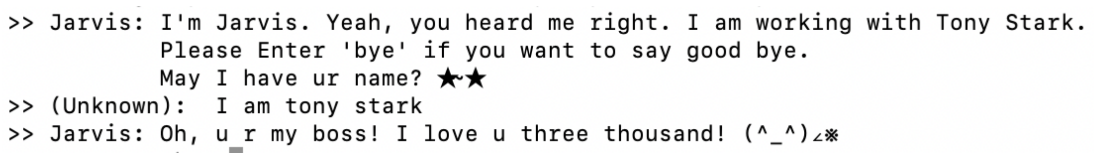
```
>> user:  change my name to tony, please
>> Jarvis: Hi, tony! >ω<
>> tony:  do you remember my name?
>> Jarvis: I have a good memory. YOU ARE tony (^_^)∠※
```

‘SMART’ Design
The system will be able to reply the current time and today’s date. Based on the time and date, the system will give a small talk.



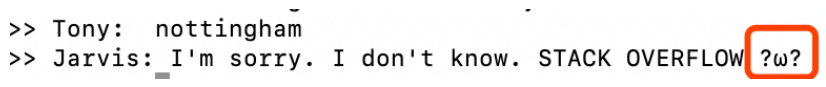
```
>> tony:  what time is it now
>> Jarvis: Do you mean the current time? It's 10:39:24 now.
>> Jarvis: Morning!
>> tony:  What is th date today
>> Jarvis: Today is Day 4/ Month 12/ Year 2020! ↖(^ω^)↗
```

The name of the chatbot is inspired by the movie ‘Iron Man’. Jarvis is an artificial intelligence created by Tony Stark. There is an ‘Easter Egg’ in the system when the user uses the name ‘Tony Stark’ **in the beginning**.



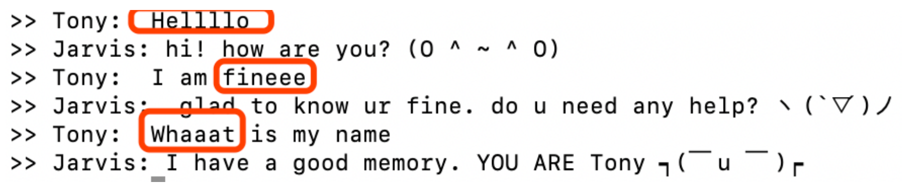
```
>> Jarvis: I'm Jarvis. Yeah, you heard me right. I am working with Tony Stark.
           Please Enter 'bye' if you want to say good bye.
           May I have ur name? ★★
>> (Unknown):  I am tony stark
>> Jarvis: Oh, u r my boss! I love u three thousand! (^_^)∠※
```

To make the system more like a human, I created an emotion library consisting of 30 different emoji. Every time the system gives a response, it will be followed by an emoji randomly selected from the library.



```
>> Tony:  nottingham
>> Jarvis: I'm sorry. I don't know. STACK OVERFLOW ?ω?
```

For each user input, I have designed a typo correction functionality to correct the typo. To be honest, this part of code is partially used and modified with reference to the online resources. Generally, it could handle over 80% of typos.



```
>> Tony:  Hellllo
>> Jarvis: hi! how are you? (O ^ ~ ^ O)
>> Tony:  I am fineee
>> Jarvis:  glad to know ur fine. do u need any help? ヽ(`▽)ノ
>> Tony:  Whaaat is my name
>> Jarvis: I have a good memory. YOU ARE Tony ┐(￣u￣)┌
```

Evaluation

The following result is based on the provided 10 testing queries on Moodle.

Q1
```
>> Tony:  Hello, how are you?
>> Jarvis:  I am good. And u? \*^o^*//
```

Q2
```
>> Tony:  What's my name?
>> Jarvis: I have a good memory. YOU ARE Tony (^_^)∠※
```

Q3
```
>> Tony:  How many people play in a Hockey team?
>> Jarvis: -sticking is the name of two infractions in the sport of ice hockey that may
occur when a player intentionally or inadvertently plays with his or her stick above the
 height of the shoulders or above the cross bar of a hockey goal. ╮(╯3╰)╭
```

Q4
```
>> Tony:  How many people live in Atlanta
>> Jarvis: Atlanta (, stressed , locally ) is the capital of and the most populous city
in the U.S. state of Georgia , with an estimated 2011 population of 432,427. <(￣︶￣)>
```

Q5
```
>> Tony:  What are the big ten?
>> Jarvis: Its twelve member institutions (which are primarily flagship research univers
ities in their respective states, well-regarded academically, and with relatively large
student enrollment) are located primarily in the Midwest , stretching from Nebraska in t
he west to Penn State in the east. ㄟ(▔皿▔ㄟ)
```

Q6
```
>> Tony:  Who is Isaac Newton?
>> Jarvis: Sir Isaac Newton (25 December 164220 March 1727) was an English physicist and
 mathematician who is widely regarded as one of the most influential scientists of all t
ime and as a key figure in the scientific revolution . ⊙﹏⊙
```

Q7
```
>> Tony:  What is single malt scotch?
>> Jarvis: Single Malt Scotch is single malt whisky made in Scotland using a pot still d
istillation process at a single distillery , with malted barley as the only grain ingred
ient. (￣3￣)a
```

Q8
```
>> Tony:  What is the best Dim Sum?
>> Jarvis: Dim sum () refers to a style of Cantonese food prepared as small bite-sized o
r individual portions of food traditionally served in small steamer baskets or on small
plates. (O ^ ~ ^ O)
```

Q9
```
>> Tony:  What is mustard made of?
>> Jarvis: Mustard (or yellow sauce) is a condiment made from the seeds of a mustard pla
nt (white or yellow mustard, Sinapis hirta ; brown or Indian mustard, Brassica juncea ;
or black mustard, B. nigra ). ︿(￣︶￣)︿
```

Q10
```
>> Tony:  What is 奶茶?
>> Jarvis: I'm sorry. I don't know. STACK OVERFLOW ヽ(`▽)ノ
```

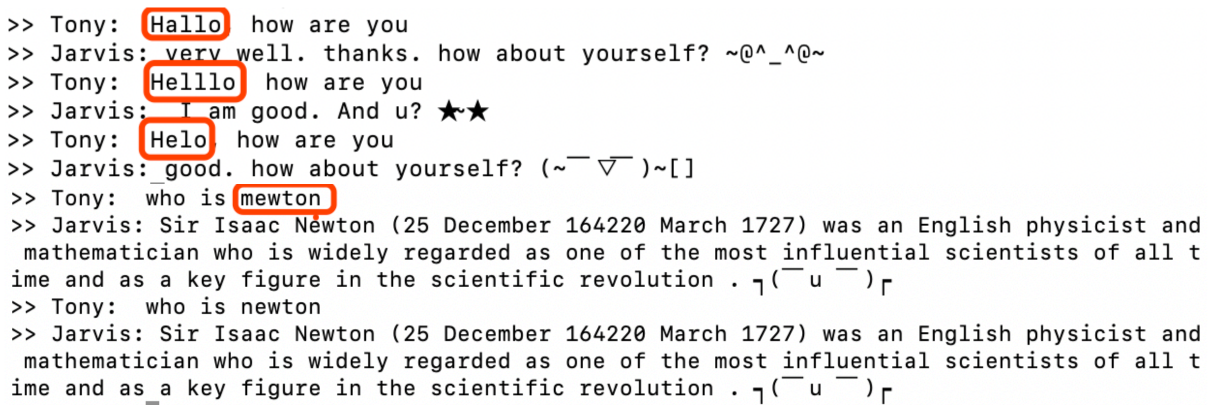
Functionality

The chatbot has been tested with different kinds of user inputs. The results indicate that the chatbot performs relatively well. The whole conversation is implemented in a loop and different intents could be effectively classified each time. As for name management, the system allows the user to change the name, and also the system can record the username for further interaction. Besides, the system handles the common small talk well with the support of my dataset.

For *QA*, the system is able to answer most of the questions, but it cannot completely understand what the user really wants to ask, because words like “how”, “what” are treated as stop words. Although the system could not perfectly answer the question, where the answer is just selected from the database without further processing, at least it could match the keywords correctly.

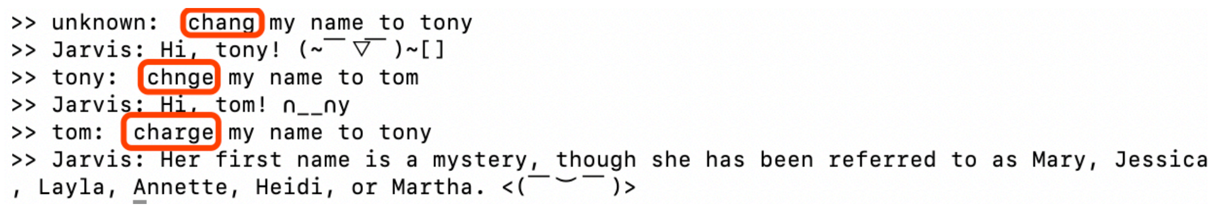
Performance

Thanks to the typo-fixing mechanism adopted, the system is able to deal with some typos and output the expected response. Some examples are presented below.



```
>> Tony:  Hallo  how are you
>> Jarvis: very well. thanks. how about yourself? ~@^_^@~
>> Tony:  Helllo  how are you
>> Jarvis:  I am good. And u? ★★
>> Tony:  Helo  how are you
>> Jarvis: good. how about yourself? (~￣▽￣)~[]
>> Tony:  who is mewton
>> Jarvis: Sir Isaac Newton (25 December 164220 March 1727) was an English physicist and
 mathematician who is widely regarded as one of the most influential scientists of all t
ime and as a key figure in the scientific revolution . ┑(￣u ￣)┍
>> Tony:  who is newton
>> Jarvis: Sir Isaac Newton (25 December 164220 March 1727) was an English physicist and
 mathematician who is widely regarded as one of the most influential scientists of all t
ime and as a key figure in the scientific revolution . ┑(￣u ￣)┍
```

However, nothing is perfect, because not all typos can be correctly fixed if the word is replaced by another word. As you can see below, the system cannot correct the word ‘charge’ to ‘change’.



```
>> unknown:  chang my name to tony
>> Jarvis: Hi, tony! (~￣▽￣)~[]
>> tony:  chnge my name to tom
>> Jarvis: Hi, tom! ი__იy
>> tom:  charge my name to tony
>> Jarvis: Her first name is a mystery, though she has been referred to as Mary, Jessica
, Layla, Annette, Heidi, or Martha. <(￣︶￣)>
```

Affect

Generally, my chatbot is able to provide a natural conversation. The answers to the small talk is randomly selected so that the user can receive different kinds of responses. Besides, different emoji are added to express human-like feelings. Some smart designs are implemented to give user surprise, such as time and date broadcasting, typo correction. However, some imperfect answers may influence user experience.

Future Direction

From the above discussion, we could see the system is not perfectly from different aspects. Here, I show you some typical shortcomings of the system as well as corresponding possible solutions.

Problems	Solutions (assuming unlimited resources)
The system misclassifies the intent occasionally. It is possible the straightforward design causes biases in intent matching.	The design of the system could be improved. It can firstly identify the intent and then search queries based on that intent. This also needs improvement of dataset, where the intent types could be indicated.
The system is not able to handle more expressions about small talk and name management.	The only solution is to enlarge the dataset and provide more ask-reply pairs
The system cannot semantically understand the question. (cannot distinguish "how", "what", etc.)	If possible, semantic understanding could be introduced into the system. With different questions, the answer could be processed before responding.
The typo correction can only fix misspelled words.	Semantic understanding could be introduced into the typo correction, where system is able to guess what the user really wants to express.
All the emoji are about positive feelings even if the expression is negative	Emoji could also be changed based on the semantic understanding module in different dialogues.
Stemming may not parse the text very well, but lemmatization is slow	Lemmatization based on tags could be used if the computational power is sufficient to lemmatize the text quickly.
The chatbot is not intelligent enough and can only be used in a specific scenario and cannot answer various questions.	Different type of datasets could be built. For the system design, deep neural networks can be trained. Things like RNN, LSTM could be used to handle the memory-based on conversation.
It is 'satire' that the chatbot cannot really chat with users.	The system can add voice input and output functionalities to improve the user interaction.