2021

AN INTERACTIVE NLP-BASED AI SYSTEM

COMP3074 - COURSEWORK 1

MICHAEL STEVENSON

Introduction

For this project I wanted to create a chatbot that did a few things. I wanted to ensure it could handle some minimal small talk with name handling, included a text-based game, handled intent classification and sentiment analysis, and did some information retrieval too.

Although it is obvious that this kind of chatbot would not exist in the real world, I thought I was a good way of fulfilling the requirements of the coursework and if nothing else, could be fun.

Background

As I made my chatbot have quite a few features, all of which I didn't go too deep into, there are many other chatbots that have done these things before. Although these chatbots have chosen 1 or 2 things and gone deeper into them.

"FAQchat as an Information Retrieval System by Bayan Abu Shawar, Eric Atwell and Andrew Roberts" is an inspiration for my chatbot. This chatbot answers FAQ's by comparing the query to the database to answer the users' query. My chatbot could be used in this same manner.

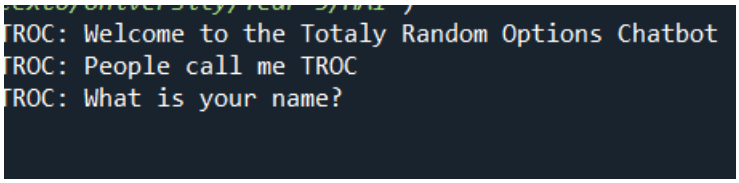
Proposed system

My system used the Stanford Question Answer Dataset for its information retrieval. This dataset is very large (hence why I chose it) and as a result made the program run very slowly. Therefor I chose to sample the first 450 lines from the dataset, all of which are about Beyonce. Thus, the IR section of the chatbot answered questions about Beyonce only.

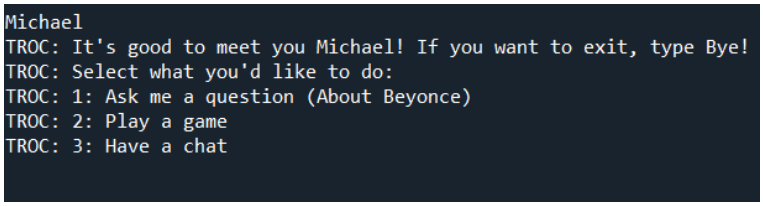
Functionality

Identity management

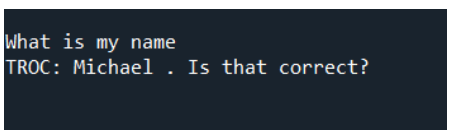
My chatbot first off introduces itself and asks for the users' name.



Once the user has input their name the system moves on and starts its intent classification by asking the user what they would like to do, with numbered options. Once the user entered the corresponding number they would be taken to the relevant option.



Note that at any point after inputting their name the user cans ask what their name is and change it if need be. When asked if the name it has for you is correct, saying no allows you to change it.



From here the chatbot breaks into 3 different sub chatbots as such.

Information retrieval

Once option 1 is selected the system will ask you what you want to know about Beyonce. Ideally this would not be the case and the system would allow you to ask anything, however the large dataset and my computing power didn't allow for that.

Once the user asks a question, the system would find the best fitting response from its database using cosine similarity.

Here we can see what happens when I asked when she left Destiny Child. The system returns the question-and-answer pair that it thinks best matches the query. Note that the question asked doesn't need to be a direct match to the question in the database and still the intended information is returned.

```
TROC: What would you like to know about Beyonce?

When did she leave Destinys child?
TROC: when did beyonce leave destiny's child and become a solo singer?,in 2003.
```

From here, the system will ask the user if they have any other questions, if the user says no the system will shut down, if they ask another question then the system will find its best response. This loops until the user says no.

Text-based game

The text-based game was never intended to be any kind of AI, instead it used simple if conditions which took the users input and played out the options as such by using text matching. The game created is very short and would need a lot of lines of code to make it a long game. However, it had the desired affect and gave the same feel as the original books from my childhood.

Small talk

The chat section of the chatbot used 3 different datasets. One for positive words, one for negative words and one for small talk questions.

The negative and positive datasets where tokenized into individual words and used to analyse the sentiment of the users input when asked how they were. By comparing the users input to the datasets the system could respond appropriately to the sentiment of the user.

The small talk dataset was tokenized into sentences and after asking the user how they were, would ask them simple small talk questions.

Implementation

I made sure to keep the system modular by having each feature in its own file and 1 file for the start-up option menu of the chatbot which they called the other files based on the users input.

```
#runs game file
if(user_response=='2'):
    exec(open("game.py").read())

    
#runs chat file
if(user_response=="3"):
    exec(open("chat.py").read())

#runs question answer file
if(user_response=="1"):

   exec(open("QA.py").read())
```

Game

Inside the game file was mainly a selection of nested if conditions that took the user input and played out the option.

```
left_or_right = input("First choice... In front of you star
if left_or_right == "left":
    count = 0
    while count != 5:
        print(".", end="")
        time.sleep(2/5)
        count += 1

    ans = input(
        "\nNice, you follow the road and find a sword on th
```

Small talk

I chose to use the Natural Language Tool Kit (NLTK). NLTK is a platform for building programs in python, it works with human language to apply natural language processing (NLP).

First of all, I needed to extract the data from the CSV file and tokenize it. We can see how I implemented this in figure 1. To ensure tokens that are the same look the same, I used .lower() to take care of capitalisation (this was also used later with the user input).

```
#creates negative words dataset
#Extract data, tokenize it  and take care of casing
f = open('negative.csv', 'r', encoding='utf-8')
raw_neg= f.read().lower()
word_neg = nltk.word_tokenize(raw_neg)# converts to list of words

#creates positive words dataset
#Extract data, tokenize it  and take care of casing
f = open('positive.csv', 'r', encoding='utf-8')
raw_pos= f.read().lower()
word_pos = nltk.word_tokenize(raw_pos)# converts to list of words

#creates small talk dataset
#Extract data, tokenize it  and take care of casing
f = open('smalltalk.csv', 'r', encoding='utf-8')
raw_small= f.read().lower()
sent_small = nltk.sent_tokenize(raw_small)# converts to list of sentences
```

I used the word_tokenize() method to split a sentence into individual tokens/words and the sent_tokenize() method to split the document into sentences. The below images show how the sentence and word tokens looked after being processed.

```
['question.,text.', 'when did beyonce start becoming popular?,in the late 1990s.', 'what
areas did beyonce compete in when she was growing up?,singing and dancing.', "when did
beyonce leave destiny's child and become a solo singer?,in 2003.\nin what city and state did
beyonce  grow up?", ',"houston, texas."', 'in which decade did beyonce become famous?,late
1990s.', "in what r&b group was she the lead singer?,destiny's child.", 'what album made her
a worldwide known artist?,dangerously in love.', "who managed the destiny's child
group?,mathew knowles.", 'when did beyoncé rise to fame?,late 1990s.', "what role did
beyoncé have in destiny's child?,lead singer.", 'what was the first album beyoncé released
as a solo artist?,dangerously in love.', 'when did beyoncé release dangerously in love?,in
2003.\nhow many grammy awards did beyoncé win for her first solo album?,five.', "what was
beyoncé's role in destiny's child?,lead singer.", "what was the name of beyoncé's first solo
album?,dangerously in love.", '"after her second solo album, what other entertainment
venture did beyonce explore?",acting.', 'which artist did beyonce marry?,jay z.', '"to set
```

```
['volatility', 'vomit', 'vomited', 'vomiting', 'vomits', 'v
'wail', 'wallow', 'wane', 'waning', 'wanton', 'war-like',
'warned', 'warning', 'warp', 'warped', 'wary', 'washed-out
'wastefulness', 'wasting', 'water-down', 'watered-down',
'weakening', 'weaker', 'weakness', 'weaknesses', 'wearines
'weed', 'weep', 'weird', 'weirdly', 'wheedle', 'whimper',
'whips', 'whore', 'whores', 'wicked', 'wickedly', 'wickedr
```

Alongside the datasets I hardcoded a few examples that would be used for greetings, saying thank you and asking the user how they are.

From here the chatbot used the users input and if conditions to see if their input matched with the negative or positive dataset and returned an appropriate comment before asking them a random question from the dataset of small talk questions.

```
if(user_response in raw_pos):
        print("TROC: awesome!")
        while(user_response!='bye'):
            print("TROC: "+random.choice(sent_small))
            user_response = input()
            user_response=user_response.lower()
```

Information retrieval

With the text from the SQUAD dataset being tokenized (as seen above) I could remove the stop words. Stop words are commonly used word (such as “the”, “a”, “an”, “in”) that we might want to ignore as they have no impact on the sentence. I imported the stop words library using “from nltk.corpus import stopwords”, this meant I could filter out the stop words and be left with my vocabulary or corpus. Here we can see the data being read, tokenized, and stop words being taken out to create the vocabulary of filtered document.

```
#Extract data, tokenize it  and take care of casing
f = open('SQuAD_Beyonce.csv', 'r', encoding='utf-8')
raw_content = f.read().lower()
sent_tokens = nltk.sent_tokenize(raw_content)# converts
word_tokens = nltk.word_tokenize(raw_content)# converts
stop_words = set(stopwords.words('english'))
#Checks if each word in the word tokens is a stop word.
filtered_doc = []

for word in word_tokens:
    if word not in stop_words:
        filtered_doc.append(word)
#Vocabulary is the filtered document.
vocabulary=filtered_doc
```

I then created methods to lemmatize the user query. This removed inflectional endings only and to return the base or dictionary form of a word or the lemma. This meant searching the corpus for the user query would be more accurate. We can see how I implement this here:

```
lemmer = nltk.stem.WordNetLemmatizer()
def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]
remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

However, after I had taken care of the word processing, and the stop words, I had an user warning telling me that my stopwords were inconsistent with my pre-processing. The issue was that the stop words needed to be processed also. We can see this here:

```
#lemmers the stop words to match the preprocessing
def _lem(self, token):
    if (token in stop_words):
        return token
    return self.lemmer.lem(token)
```

To get a response to the users query I implemented sklearn; Scikit-learn is a machine learning library for the Python programming language, to apply the Term Frequency-Inverse Document Frequency (tfidf) and the cosine similarity.

TfidfVectorizer is used to rescale the frequency of words by how often they appear in all documents. This means relevant words are scored higher than words such as “the”, “and” “they” etc.

Here is the function that gets the best response for the user:

```
#Function to get best answer to question
def response(user_response):
    response=''
#takes user input
    sent_tokens.append(user_response)
#Convert a collection of raw documents to a matrix of TF-IDF features
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens)
    vals = cosine_similarity(tfidf[-1], tfidf)
    
#the responses comes from here.
    idx=vals.argsort()[0][-2]
    
#flattens and sorts the vector into linear.
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]

#if theres no good similarity
    if(req_tfidf==0):
        response=response+"I am sorry! I don't understand you"
        return response
#if there is a similarity
    else:
        response = response+ sent_tokens[idx]
        return response
```

Once the user response is passed to this function, it adds it to the sentence tokens and applies tfidf to get a real-valued vector in vector space. This is then compared, with cosine similarity, to the corpus to get the similarity of the pair of vectors using their dot product and dividing that by the product of their norms. Using this formula, we can find out the similarity between the query and the sentence tokens in the corpus.

Cosine Similarity (query, corpus) = dot product (query, corpus) / ||query|| * ||corpus|

This gives us scores for the documents that are similar to the query:

```
(1200, 2001)  0.19369447947221194
(1200, 4452)  0.18123164617916415
(1200, 7116)  0.36970391705712735
(1200, 6660)  0.18485195852856368
(1200, 526)   0.14862348797542027
(1200, 6922)  0.18123164617916415
(1200, 5858)  0.15375825090926146
(1200, 5179)  0.15826442079292893
(1200, 5182)  0.143217113887727
(1200, 323)   0.18485195852856368
(1201, 4453)  0.6433032795157884
(1201, 9405)  0.5966721215470667
(1201, 526)   0.47973249830791515
```

These values are then sorted, and the best score is returned as the response.

```
[0.         0.         0.         ... 0.18473389 0.64667728
0.646677284084067
```

Evaluation

To evaluate my system I did 2 things; user testing and benchmark testing.

User test

User A Scenario: “I want to be able to play a small text-based game and then shut the program down when finished”

Assumptions: The game makes sense, and I am able to win the game.

	Yes	No
Did game load?	X	
Did the game shut down after use?	X	
Was you able to win the game?	X	
Did the game make sense?	X	

User B scenario: “I want to be able to interactive with the chatbot, change my name and have some small talk”

Assumptions: The small talk will feel natural

	Yes	No
Could you change your name?	X	
Did you have interaction?	X	
Did this interaction feel natural?		X

User C: “I want to be able to ask the chatbot some questions and get a satisfactory answer”

Assumptions: Chatbot gives the correct, or best answer

	Yes	No
Could you get to the QA section	X	
Could you ask questions?	X	
Did you get satisfactory answers?	X	

Benchmarking single systems

For my system to be able to function on my computer, I created a test dataset from SQUAD. This dataset, by chance, was the Beyonce dataset. I created a benchmark test of 10 questions, some of which are worded identical to the dataset, others that would require a bit more work.

Question to ask	Expected answer
When did Beyonce start becoming popular?	in the late 1990s.
What areas did Beyonce compete in when she was growing up?	singing and dancing.
When did she leave Destiny' Child?	In 2003.
In what city and state did Beyonce grow up?	Houston, Texas.
In which decade did she become famous?	late 1990s.
In what R&B group was she the lead singer?	Destiny's Child.
What album made her known worldwide?	Dangerously in Love.
Who managed Destiny's Child?	Mathew Knowles.
Who is she married to?	Jay Z
What role did Beyonce have in Destiny's Child?	lead singer.

Below we can see the results from asking the chatbot these questions. I am pleased that on every single question it gets the intended answer. However, there are a few instances where it also returns a line from an unintended question due to sentence tokens. Here we have green being as expected, and orange being not as expected.

Question to ask	Expected answer
When did Beyonce start becoming popular?	in the late 1990s.
What areas did Beyonce compete in when she was growing up?	singing and dancing.
When did she leave Destiny' Child?	In 2003.
In what city and state did Beyonce grow up?	Houston, Texas.
In which decade did she become famous?	late 1990s.
In what R&B group was she the lead singer?	Destiny's Child.
What album made her known worldwide?	Dangerously in Love.
Who managed Destiny's Child?	Mathew Knowles.
Who is she married to?	Jay Z
What role did Beyonce have in Destiny's Child?	lead singer.

```
When did Beyonce start becoming popular?
TROC: when did beyonce start becoming popular?,in the late 1990s.
TROC: Anything else?

What areas did Beyonce compete in when she was growing up?
TROC: what areas did beyonce compete in when she was growing up?,singing and dancing.
TROC: Anything else?

When did she leave Destiny' Child?
TROC: when did beyonce leave destiny's child and become a solo singer?,in 2003.
in what city and state did beyonce  grow up?
TROC: Anything else?

In what city and state did Beyonce grow up?
TROC: when did beyonce leave destiny's child and become a solo singer?,in 2003.
in what city and state did beyonce  grow up?
TROC: Anything else?

In which decade did she become famous?
TROC: in which decade did beyonce become famous?,late 1990s.
TROC: Anything else?

In what R&B group was she the lead singer?
TROC: in what r&b group was she the lead singer?,destiny's child.
TROC: Anything else?

What album made her known worldwide?
TROC: what album made her a worldwide known artist?,dangerously in love.
TROC: Anything else?

Who managed Destiny's Child?
TROC: who managed the destiny's child group?,mathew knowles.
TROC: Anything else?

Who is she married to?
TROC: who is beyoncé married to?,jay z.
what is the name of beyoncé's alter-ego?,sasha fierce.
TROC: Anything else?

What role did Beyonce have in Destiny's Child?
TROC: what role did beyoncé have in destiny's child?,lead singer.
TROC: Anything else?
```

Discussion

Overall, I am quite pleased with the results from the chatbot, especially the main part of it; the IR. It does what I set out for it to do with little errors and although in this example I only used a fraction of the SQuAD this would work just as well with the full SQuAD set given enough computing power.

My benchmark results showed that my chatbot returned the expected result 70% of the time and for the remaining 30% it gave the correct answer but also gave some additional answer. I can see that this is because my sentence tokens do not process the csv file correctly. In the question asking who Beyonce is married to, we can see that the token does not end with the full stop after “Jay z”. Instead, it creates a new line and starts the next question-answer pair. This means 2 sentences are being made into 1 sentence token. If I could have fixed this, then my chatbot would have returned the expected result 100% of the time.

```
01/06/2005.", "what was the name of beyoncé's second solo album?,b'day.", '"what was
beyoncé\'s first acting job, in 2006?",dreamgirls.', "who is beyonce married to?,jay z.
\nwhat is the name of beyonce's alter-ego?,sasha fierce.", '"in her music, what are some
recurring elements in them?', '","love, relationships, and monogamy."', 'time magazine
```

The other 2 elements of my chatbot; the game and the small talk both worked as expected. Allowing the user to have minimal exchanges with the system, and handling identity management. They were very simple and included a lot of “dumb AI”. In future attempts it would be useful to improve on this

by not relying on strict matching; for example, if the user replied with a negative word that wasn't in the negative or positive dataset then the system would shut down and although the dataset is very large and includes most answers it can still break. The same applies with things like name changing, the user input must include "change" and "name" and wording this differently doesn't trigger the name change feature.

Also, there are some biases in the sentiment dataset (Twitter dataset). For example, the words "black", "male" and "left-leaning" are all in the negative datasets. Therefore, it might be wise to not use this dataset at all in the future, or at least edit it to remove any biases.

Conclusion

In conclusion I think I was able to meet the criteria for the project and provided a chatbot with 3 features that had little issues and good usability. The evaluation was very positive, the users' assumptions were met, and the benchmark results were good.