Dylan Kapustka, Abed Ahmed

Dr. Mazidi

CS 4349.001

November 14, 2021

Chatbot Report

System Description:

This is a simple chatbot created in PyTorch utilizing some basic Natural Language Processing (NLP) techniques. First and foremost, we are unable to just pass the input sentence that our user provides into our neural net (as is). Therefore, we need to convert our text so that our network can understand it properly. To do this, we are using what is known as the bag of words (bow) method. Essentially what this process does is disregard grammar and even word order but keeps the multiplicity. To create a bag of words, we first need training data. We have decided to utilize a JSON file for our knowledgebase that we have labeled as "intents." We will go through this knowledge base and take all our pattern and response words that exist within the base and use these words to calculate the bag of words for each sentence our user enters.

Before we can even think about calculating the bow, we still need to apply some NLP techniques. We decided to use Tokenization and Stemming to create accurate training data. We tokenized any incoming sentence from our user as well as tokenized and stemmed the sentences within our knowledgebase patterns.

```
#loop through intents patterns
for intent in intents['intents']:
    tag = intent['tag']
    tags.append(tag)
    for pattern in intent['patterns']:
        w = tokenize(pattern)
```

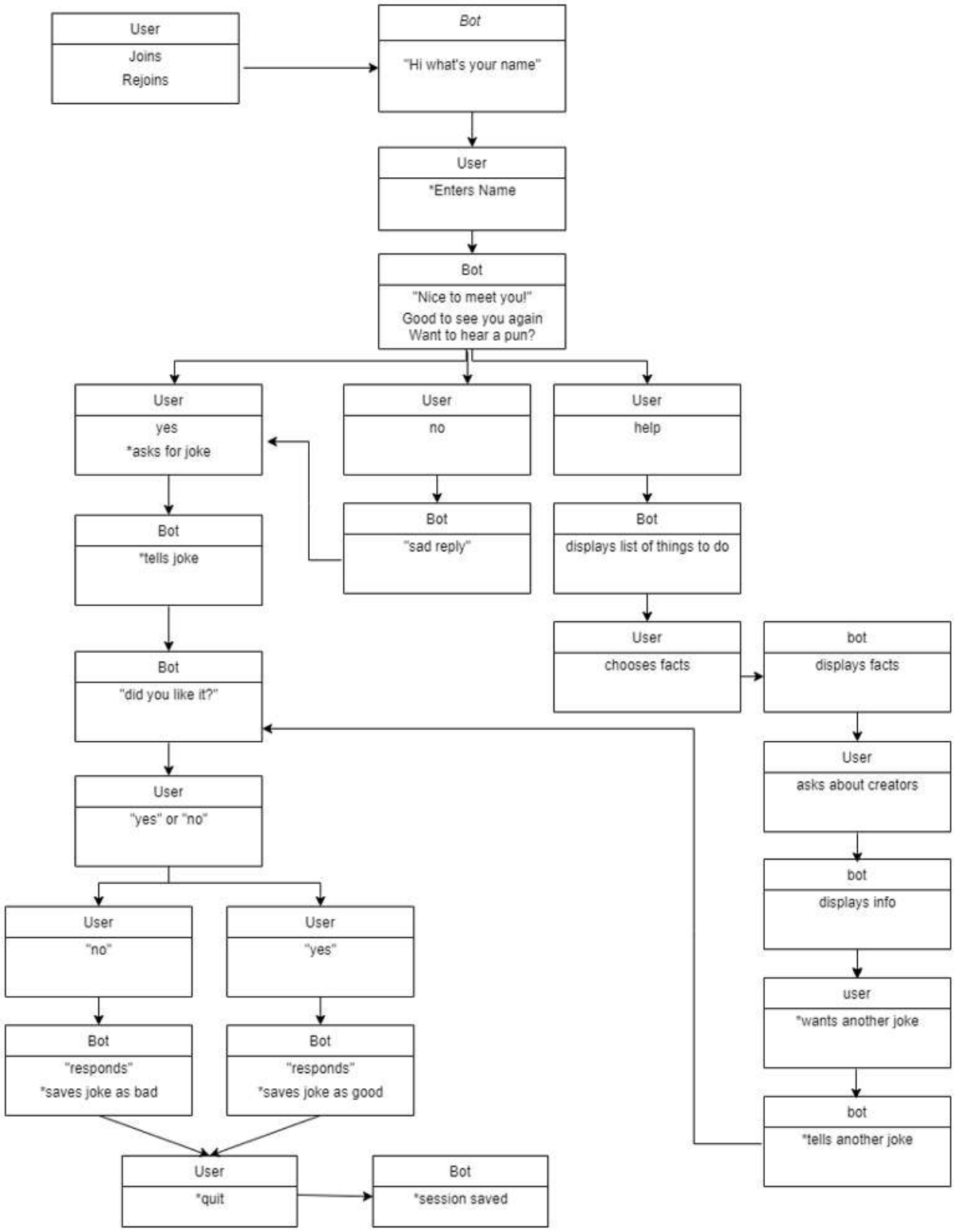
```
sentence = tokenize(sentence)
```

```
# apply stemming and lowercasing
ignore_words = ['?', '.', '!', ',', "'s"]
all_words = [stem(w) for w in all_words if w not in ignore_words]
# remove duplicates and sort
all_words = sorted(set(all_words))
```

This will ensure accuracy of the tag prediction.

Once our tag prediction takes place and if the probability is above 75%, our bot will reference one of the responses from the decided intent tags and randomly generate a response. This will continue until the user quits the program.

Dialog Tree:



The following are real chat logs. Here, the user is joining the session for the first time:

Hey there! Welcome to the Punn-iest bot, Where I tell you the best puns! type (quit) to close and save our session!

What's your name?: Dylan

Punny Bot: Nice to meet you, Dylan! My name is Punny Bot.

Punny Bot: Would you like to hear a pun?

Dylan: not really

Punny Bot: I'm trying my best here

Dylan: try harder

Punny Bot: A friend said he didn’t understand cloning. I told him that makes two of us.

Punny Bot: Did you like that one?

Dylan: no

Punny Bot: I am terribly sorry you have bad taste, but I will avoid that joke from now on

Punny Bot: Would you like to hear another?

Dylan: no go away

Punny Bot: Well this is awkward..

Dylan: ok fine

Punny Bot: I went to that new restaurant, Karma. There’s no menu, you just get what you deserve.

Punny Bot: Did you like that one?

Dylan: yes

Punny Bot: I am glad you liked it! I'll go ahead and save that one for another time!

Punny Bot: Would you like to hear another?

Dylan: sure thing

Punny Bot: Imagine if Americans switched from pounds to kilograms overnight. There would be mass confusion!

Punny Bot: Did you like that one?

Dylan: yes

Punny Bot: I am glad you liked it! I'll go ahead and save that one for another time!

Punny Bot: Would you like to hear another?

Dylan: no, can we talk about something else?

Punny Bot: Ok! Breaks are healthy. Puns are what I am best at, but here's some things I can talk about:

History of Puns

Who my creators are

What I think about this class

Dylan: what is a pun anyway?

Punny Bot: The pun, also known as paronomasia, is a form of word play that exploits multiple meanings of a term, or of similar-sounding words, for an intended humorous or rhetorical effect

Dylan: tell me some pun facts

Punny Bot: Homonymic puns, another common type, arise from the exploitation of words which are both homographs and homophones.

Dylan: more pun facts

Punny Bot: Homonymic puns, another common type, arise from the exploitation of words which are both homographs and homophones.

Dylan: lol

Punny Bot: I know I am hilarious!

Dylan: ok another pun fact

Punny Bot: Human Language Technologies has been a great learning experience

Dylan: no wrong thing

Punny Bot: Ok, if you don't want to hear anything else I understand, your loss..

Dylan: pun fact please

Punny Bot: I do not understand...

Dylan: pun fact

Punny Bot: A compound pun is a statement that contains two or more puns. In this case, the wordplay cannot go into effect by utilizing the separate words or phrases of the puns that make up the entire statement.

Dylan: thank you

Punny Bot: Hi there, what can I do for you?

Dylan: lol

Punny Bot: Do you know anyone who can get me a comedy gig? Warning, if you say yes, I'll probably fire off a top tier joke

Dylan: yes

Punny Bot: Life as a professional yo-yoer has its ups and downs.

Punny Bot: Did you like that one?

Dylan: no

Punny Bot: I am terribly sorry you have bad taste, but I will avoid that joke from now on

Punny Bot: Would you like to hear another?

Dylan: who made you?

Punny Bot: I was created by Dylan Kapustka and Abed Ahmed. I am exactly a week old and ready for the world. They are the best people woooooo

Dylan: what do you think of this course?

Punny Bot: I love this class! I hope Dr. Mazidi curves haha

Dylan: Can you tell me my least favorite joke

Life as a professional yo-yoer has its ups and downs.

Punny Bot: Did I remember this one correctly?

Dylan: yes

Punny Bot: Awesome! I am glad I remembered that bad joke for you

Punny Bot: Would you like to hear another?

Dylan: no thanks

Punny Bot: Well this is awkward..

Dylan: goodbye

Punny Bot: Have a nice day

Dylan: quit

User rejoins the session – (Same name entered)

Hey there! Welcome to the Punn-iest bot, Where I tell you the best puns! type (quit) to close and save our session!

What's your name?: Dylan

Punny Bot: Welcome back, Dylan! I see you missed me!

Punny Bot: Would you like to hear a pun?

Dylan: tell me one of my favorites

Imagine if Americans switched from pounds to kilograms overnight. There would be mass confusion!

Did I remember this one correctly?

Dylan: no

Punny Bot: You're a liar!! Haha just kidding, I'll go ahead and remove that joke from the list then

Punny Bot: Would you like to hear another?

Dylan: yes

Punny Bot: I'm afraid of negative numbers. I'll stop at nothing to avoid them.

Punny Bot: Did you like that one?

Dylan: yes

Punny Bot: I am glad you liked it! I'll go ahead and save that one for another time!

Punny Bot: Would you like to hear another?

Dylan: no

Punny Bot: Well this is awkward..

Dylan: why is this awkward?

Punny Bot: You're annoying, stop trying to break my bot >:(

Dylan: why am I annoying?

Punny Bot: alt-f4

Dylan: ok tell me a joke

Punny Bot: I’m reading a book about anti-gravity. It’s impossible to put down.

Punny Bot: Did you like that one?

Dylan: no

Punny Bot: I am terribly sorry you have bad taste, but I will avoid that joke from now on

Punny Bot: Would you like to hear another?

Dylan: no

Punny Bot: I'm trying my best here

Dylan: goodbye

Punny Bot: Bye! Come back again soon.

Dylan: quit

Breakdown:

As you can see, we have a somewhat fluid and functional bot when it comes to knowing what the user is looking for. Though it is nowhere near perfect, it gets the job done, if the user is honest with how he or she decides to use it. I want to explain how we are storing our user’s name, likes, and dislikes:

For a somewhat mid-level explanation, we are constantly pickling and unpickling a dictionary, and that is how the bot remembers previous interactions. It uses the names as keys and user objects as values. The user objects store information about previous interactions, favorite puns, and least favorite puns. Here are some brief code snippets:

```
class Person:

    def __init__(self, name):
        self.name = name
        self.badpuns = list()
        self.goodpuns = list()
```

if exists() is to check if pickle file has been created. Meaning the session has already been used by one user (not necessarily the same one):

```
person_facts = {}
if exists('datastore'):
    with open('datastore', 'rb') as file:

        person_facts = pickle.load(file)
```

When the user rejoins a session and uses the same name:

```
else:
    current_person = person_facts[name.lower()]
    print(f"{bot_name}: Welcome back,", current_person.name.title() + "!" " I see you missed me!")
    print(f"{bot_name}: Would you like to hear a pun?")
```

After the user is done with the bot dialog and decides to quit the program, we will pickle our dictionary and save the session info:

```
if sentence == "quit":
    picklefile = open('datastore', 'wb')
    pickle.dump(person_facts, picklefile)
    picklefile.close()
    quit()
```

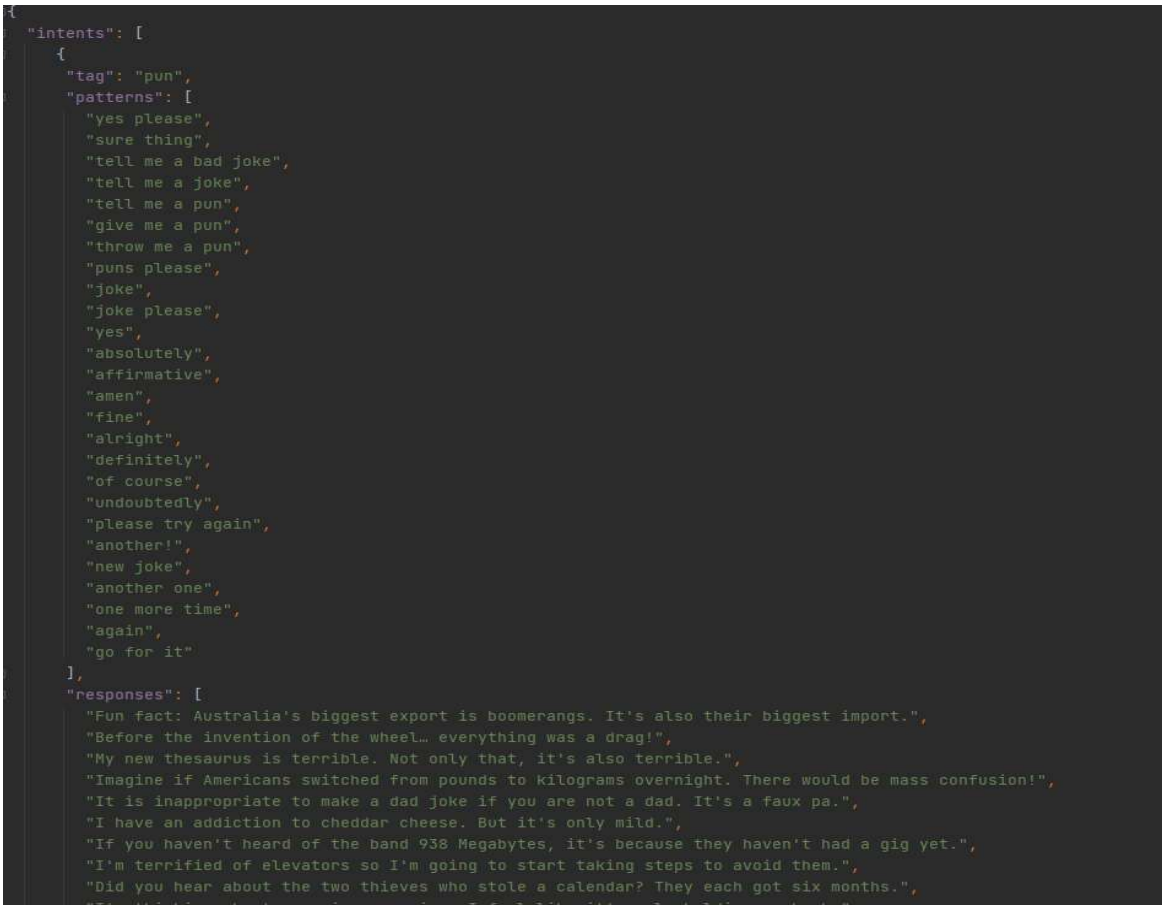
I want to point out something I did intentionally when the user rejoins the session. When I ask the bot to tell me one of my favorite jokes and it does, I respond by telling the bot, that the joke remembered was not the right one. Even though the bot remembered the correct joke, I wanted to show how we can remove the joke from the list of remembered jokes, if the user decides he or she likes or dislikes that joke after all. When I said “no” the bot went ahead and removed the joke from our list of remembered jokes. You can see the code below:

```
if tag =='favorite':
    if len(current_person.goodpuns) == 0:
        print(f"{bot_name}: It seems you hated all my jokes! I couldn't find any good puns you saved")
        print(f"{bot_name}: Would you like to hear another?")
        break
    else:
        goodpun = random.choice(current_person.goodpuns)
        print(goodpun)
        print("Did I remember this one correctly?")
        check_good = input(name.title() + ":")
        while "yes" != check_good and "no" != check_good:
            print(f"{bot_name}: pleasee, just tell me. Keep it simple, yes or no")
            check_good = input(name.title() + ": ")
        if check_good == "yes":
            print(f"{bot_name}: Awesome! I am glad I remembered that good joke for you")
            print(f"{bot_name}: Would you like to hear another?")
            break
        elif check_good == "no":
            current_person.goodpuns.remove(goodpun)
            print(f"{bot_name}: You're a liar!! Haha just kidding, I'll go ahead and remove that "
                  "joke from the list then")
            print(f"{bot_name}: Would you like to hear another?")
            break
```

The same type of functionality exists for our disliked puns as well. You can ask the bot to tell you puns you liked or disliked, and the bot will randomly select one of them from the saved list and output it to you. If you didn't dislike or like any puns, then you will see a message stating such.

Now I want to show you our bot's knowledge base and how it is structured.

Our bot is trained a json file, and the structure is very easy to understand. We have what is known as "intents" (what we think the user is asking), and for each intent we have a "tag" which is basically our class label. We have different "patterns" for each tag such as "Hi, Hey, how are you?" asked by the user. To get different responses for each tag, we use "responses" and randomly generate a response from "responses." Whenever a new sentence or question comes in, our bot tries to classify it in one of our tags. If the bot recognizes that our question or sentence is a greeting, then it will go to the greeting "tag" and randomly generate one of the chosen responses. Below are some screenshots of our knowledge base, and the complementary code as to how our logic complements it:



```
{
  "intents": [
    {
     "tag": "pun",
     "patterns": [
       "yes please",
       "sure thing",
       "tell me a bad joke",
       "tell me a joke",
       "tell me a pun",
       "give me a pun",
       "throw me a pun",
       "puns please",
       "joke",
       "joke please",
       "yes",
       "absolutely",
       "affirmative",
       "amen",
       "fine",
       "alright",
       "definitely",
       "of course",
       "undoubtedly",
       "please try again",
       "another!",
       "new joke",
       "another one",
       "one more time",
       "again",
       "go for it"
     ],
     "responses": [
       "Fun fact: Australia's biggest export is boomerangs. It's also their biggest import.",
       "Before the invention of the wheel… everything was a drag!",
       "My new thesaurus is terrible. Not only that, it's also terrible.",
       "Imagine if Americans switched from pounds to kilograms overnight. There would be mass confusion!",
       "It is inappropriate to make a dad joke if you are not a dad. It's a faux pa.",
       "I have an addiction to cheddar cheese. But it's only mild.",
       "If you haven't heard of the band 938 Megabytes, it's because they haven't had a gig yet.",
       "I'm terrified of elevators so I'm going to start taking steps to avoid them.",
       "Did you hear about the two thieves who stole a calendar? They each got six months.",
```

```
    {
  "tag": "no-pun",
  "patterns": [
    "this is boring",
    "this is stupid",
    "lame",
    "end me",
    "you are not punny",
    "I don't want to hear a joke",
    "no more jokes",
    "you aren't funny",
    "cringe",
    "trash",
    "stop",
    "awful",
    "horrible",
    "no I hated it",
    "no I did not like it",
    "not really",
    "not even one bit",
    "not at all",
    "no way",
    "na",
    "no thanks",
    "no"
  ],
  "responses": [
    "Tough crowd",
    "I'm trying my best here",
    "Well this is awkward..",
    "But, I know for a FACT that I am punny!",
    "Ok, if you don't want to hear anything else I understand, your loss..",
    "I am sorry you didn't like my joke.. NOT"
  ]
},
```

These are just two examples of the multiple tags that we have within our Json file. Here are some samples of what is happening:

User: Tell me a joke

Our logic will go and search our knowledgebase trying to predict what is the best fit for that sentence the user provided. After predicting that the “tag”: “pun” it will then randomly choose a response. It could look something like this:

Bot: "My boat was cold, I tried to make a fire, but it sank. I guess you can’t have your kayak and heat it too.”

Here are some sample user models that were created after multiple user sessions took place within the chatbot:

This is the user model for a user named Dylan:

```
{'dylan': <__main__.Person object at 0x000001AC8473D9D0>}
GoodPuns: ['Long fairy tales have a tendency to dragon.', 'To the guy who invented Zero, thanks for nothing!']
Bad Puns: ['I did a theatrical performance about puns. It was a play on words.']
```

Let’s see what happens when we have a new user join the session:

```
{'dylan': <__main__.Person object at 0x000002391543DA30>, 'abed': <__main__.Person object at 0x000002391543D9D0>}
GoodPuns: ['Life as a professional yo-yoer has its ups and downs.', 'RIP boiling water. You will be mist.']
Bad Puns: ["If you don't C sharp before crossing the street, you'll B flat."]
```

The “goodpuns” and “badpuns” being outputted above are session dependent, but those lists represent the puns that the user likes and dislikes however they are being independently stored in each person object. We can see this tested below. I will log in as user Dylan and then Abed:

```
What's your name?: Dylan
Punny Bot: Welcome back, Dylan!  I see you missed me!
Punny Bot: Would you like to hear a pun?
Dylan: tell me one of my favorites
Long fairy tales have a tendency to dragon.
Did I remember this one correctly?
Dylan:no
Punny Bot: You're a liar!! Haha just kidding, I'll go ahead and remove that joke from the list then
Punny Bot: Would you like to hear another?
{'dylan': <__main__.Person object at 0x000002874DAAD9D0>}
GoodPuns: ['To the guy who invented Zero, thanks for nothing!']
Bad Puns: ['I did a theatrical performance about puns. It was a play on words.']
```

Here, you can see that the bot correctly remembered and displayed one of the pun's that were favorited. After telling the bot no, you can see the pun get deleted from the "goodpuns" list. Lets try Abed:

```
What's your name?: abed
Punny Bot: Welcome back, Abed!  I see you missed me!
Punny Bot: Would you like to hear a pun?
Abed: tell me one of my least favorites
If you don't C sharp before crossing the street, you'll B flat.
Punny Bot: Did I remember this one correctly?
Abed:yes
Punny Bot: Awesome! I am glad I remembered that bad joke for you
Punny Bot: Would you like to hear another?
{'dylan': <__main__.Person object at 0x0000017DB2D9DA30>, 'abed': <__main__.Person object at 0x0000017DB2D9D9D0>}
GoodPuns: ['Life as a professional yo-yoer has its ups and downs.', 'RIP boiling water. You will be mist.']
Bad Puns: ["If you don't C sharp before crossing the street, you'll B flat."]
```

You can see how we were still able to remember Abed's state as well as Dylan's.

Evaluation/Summary:

Overall, there is a lot to improve with this chatbot, but for the scope of the project, we felt it accomplished a fair amount. Expanding on the bot further, we would expand the knowledge base quite a bit and find a way to make the responses even more fine tuned and accurate to give the user a more immersive experience. The chat bots' strengths would be remembering the user sessions and being able to predict the user's intents, even if context is seemingly out of order. It does a good job reading intent, even when the user adds additional words to the questions being asked. We thought that was neat and made the chatbot seem much more personable. Some weaknesses would be that the chatbot can get lost sometimes, and some keywords seem to trigger random responses. I think overall if the chatbot is used with the intention of using it correctly, it will perform well and do everything you want to do. So, try it out, before attempting to break it! 😉