**JUSTIN AEGTEMA**

**NLP CS 6320.001**

**NLP**

**CHATBOT PROJECT: ROB THE ROBOT**

**TRAVEL GUIDE FOR US CITIES**

**APRIL 2024**

# SUMMARY:

* ROB the Robot is a travel guide chatbot for popular cities in the United States
* ROB draws on a huge knowledge base of over 700 pieces of knowledge obtained from over 550 urls meant to give ROB knowledge of over 15 US cities, which can be queried by users using either city name or desired activity type
* **This is version 2 of ROB – an update of the version submitted in March for Project 1**
* Previously, ROB selected his response using NLP techniques, but that has been replaced with a **Naïve Bayes Classifier which I wrote** (not from a library) (see **aeg\_naive\_bayes\_classifier.py**)
* The Naïve Bayes Classifier was trained on **3200+ instances of synthetic data generated using ChatGPT**, and it achieved 0.99 accuracy when synthetic data was split into training/validation sets
* Along with the bread of his knowledge, ROB also features a cartoon face which appears in nearly every response
* ROB tells jokes/makes funny comments at times that will seem random to first-time users
* After talking to the user about a city or activity, ROB will ask the user if they like/dislike it, remember their response, and re-mention liked things later.
* Key Note: This report details my second attempt at tackling Project 2. My first attempt, to create a totally new chatbot that generated text using a from-scratch transformer, took 10+ hours and ended in the decision to try something else. **The first attempt is detailed in Appendix 1,** in case the grader is interested, as I know some graders are interested in effort put in overall. That said, I am really satisfied with my second attempt at Project 2, detailed in the main report here.

# System Description: Knowledge Base Data Set, Generating Synthetic Training Data, NLP Techniques, and ML Techniques

## Knowledge Base Data Set

The Knowledge Base used for the chatbot responses was the exact same as that used for project 1. It includes over 700 pieces of knowledge obtained from over 550 urls meant to give ROB knowledge of over 15 US cities, which can be queried by users using either city name or desired activity type. This Knowledge Base was developed using hundreds of lines of code across half a dozen .py files. Text processing was effective but simple, relying on removal of white space, unnecessary symbols, and for tf-idf analysis: lower-casing, tokenization, stop word removal, lemmatization, etc. The most productive insight in Knowledge Base creation was that simply aiming to break text up at natural breaks (end-of-sentence, denoted by periods), the result was chunks of text that natural formed what might be called “travel fun facts” which reliably made for excellent chatbot conversational material. This productive insight allowed the Knowledge Base to be generated at enormous scale in short time, which is what allows ROB to have over 700 pieces of knowledge without the need for manual review of every factiod. However, since this work was done for Part 1, it will not be detailed here further. If the grader is interested in this, see the Project 1 Report, in the folder “nonessential files for submission”.

While not actually used in the code, Knowledge base appendix.xlsx shows what the Knowledge Base that the code uses looks like.

## Generating Synthetic Training Data

Training data quality is very important. Training data was generated using ChatGPT. In total, ROB requires 25 categories to effectively navigate his various responses (16 for US cities, 7 for travel activities, 1 to discuss his capabilities, and 1 for general recommendations).

Selecting among 25 categories using natural language input is a serious challenge! Unfortunately, I felt that ChatGPT was not as up-to-the-task as I had expected (see Synthetic\_Request\_Data.csv for results). The data generated was certainly usable. However, it was repetitive to a fault.

Ultimately, my data set included 3200 data entries: 100 for each city category, the general category, and the capabilities category, and 200 for each activity category.

The second set of 100 each activity category data entries were generated separately, to add variety to the original 2500 entry data set.

I was surprised by how challenging it was to encourage ChatGPT to generate productive data at scale. I had to word my requests very carefully to express every detail I wanted, but requests that were too long became hard for ChatGPT to capture in a sophisticated way.

Amusingly, I asked it to generate 100 data entries for the “general” category, which is for when the user wants a general recommendation. However, I found that these data entries used the word “recommendation” more frequently than the 16 city prompts. This was problematic, as a common prompt I want the model to be able to handle is “Please give me recommendations for my trip to Dallas” and the model was classifying that request as “general”!

After a lot of back and forth with ChatGPT, I was able to get it to give me 100 data entries for the “general” category which were requests for “general recommendations” but which did not use the key word “recommendation”. Needless to say, this fixed the original problem, but it created a data set that was unnatural.

Ultimately, a major lesson learned throughout this process was:

Machine learning models such as Naïve Bayes are capable of astonishing intelligence, but they are only as smart as the data you give them. I had known that the state-of-the-art machine learning engineers focus heavily on data generation, but this experience taught me that first hand. The key lesson here was: put a big focus on getting or generating high quality data, because it’s all downstream of that.

This experience also makes it clear how useful it is to have a client or user base providing you with constant, real data.

## NLP Techniques

Numerous NLP Techniques were used.

As mentioned already under the Knowledge Base Dat Set heading, the KB creation process included tf-idf analysis, which naturally included lower-casing, tokenization, stop-= word removal, and lemmatization.

Additionally, a key-word search process functionally very similar to cosine similarity was used to interpret user responses to “yes” or “no” questions. “yes”, “no”, and various synonyms such as “nah” were compared to user response with a priority list, with the first match used to determine user intent. This was used to determine if the user liked/disliked a particular city or activity (using Naïve Bayes for answers to yes/no questions seemed like overkill). Naturally, user responses were first processed using lower-casing, removal of non-alpha characters, tokenization, and lemmatization. Interestingly, NLTK considers “no” a stop word – so stopwords certainly were not removed to process these yes/no responses!

Finally, NLTK techniques were similarly used to pre-process text before it was processed by the Naïve Bayes Classifier. This included the usual suspects – lower-casing, removal of non-alpha characters, tokenization, and lemmatization. However, since the Naïve Bayes Classifier was never used for yes/no questions, stop words were also removed.

## ML Techniques

While I spent 10+ hours on a wholy original attempt of Project 2 which would have used a transformer if it were successful (see Appendix 1), I ultimately gave up on that attempt and opted to modify my Project 1 instead.

In the final delivered project, ROB uses a Naïve Bayes Classifier to determine user intent. **Rather than use a library, I wrote the code for this Naïve Bayes Classifier** (see **aeg\_naive\_bayes\_classifier.py**)

The user can respond to the chatbot requesting information pertaining to 25 different categories – 16 cities, 7 travel activities, the category “general” which exists for general travel recommendations, and the category “capabilities” which exists to help graders and others understand ROB’s capabilities with higher specificity.

My Naïve Bayes Classifier takes the user’s typed input as a text string and outputs one of the 25 categories, which is then used by the chatbot to select an appropriate response.

Anecdotally, the Naïve Bayes Classifier is pretty good at noticing key words such as “Dallas” or “Beach”, seleting the appropriate category, so that the chatbot can provide the user with a fitting response.

While the Project 1 version was able to catch a limited number of key words, the expansive training data and power of the Naïve Bayes Classification algorithm opened the doors for the model to become far more sophisticated this time around. While the data set was more limited than I wanted, it still allowed for improvements on Project 1, in certain ways.

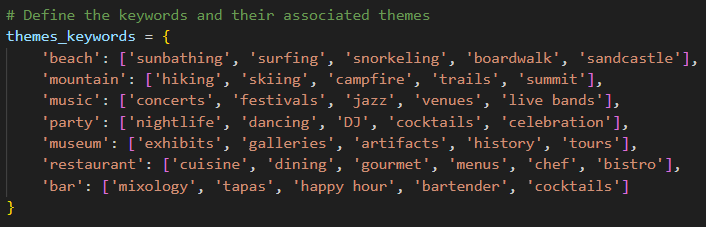


Figure 1

As shown in Figure 1, an expanded set of terms was used for data generation. *These terms allow the model to respond appropriately in ways that version 1 could not.* *For example, a user question mentioning fine dining or nightlife will get a relevant response from the restaurant or party category, as shown in the Sample B Dialog with Benjamin below.* If I could generate a more expansive, complex data set, these classification abilities would surely improve exponentially.

Using the 3200 synthetic data entries, I split them into 80% train, 20% validation, finding that **my Naïve Bayes Classifier achieved 99% accuracy in classification. While at first this might seem like a huge success, it’s probably actually an indication that the data set is too repetitive and similar.**

Anecdotally, it’s actual accuracy is probably closer to 95% if all categories are equally prompted. If grey areas are overrepresented, it will perform worse. Grey areas consistently trip it up. For vague user input, the goal is for that to be categorized as “general” so that the model can provide a couple general travel recommendations, then request more user input. However, the Naïve Bayes Classifier tends to split grey input – a good amount winds up in “general”, but it also commonly gets classified as “capabilities” instead.

# Chatbot Features, Dialog Tree, and Sample Dialog

Note that apart from the introduction of Naïve Bayes powered by Synthetic Data as the classification technique, several other small improvements were made to ROB for Project 2.

In the prior model, bugs prevented Austin, Las Vegas, and the topic “bars” from working properly. All of those bugs were fixed for version 2.

Here are some notable interactions ROB will have:

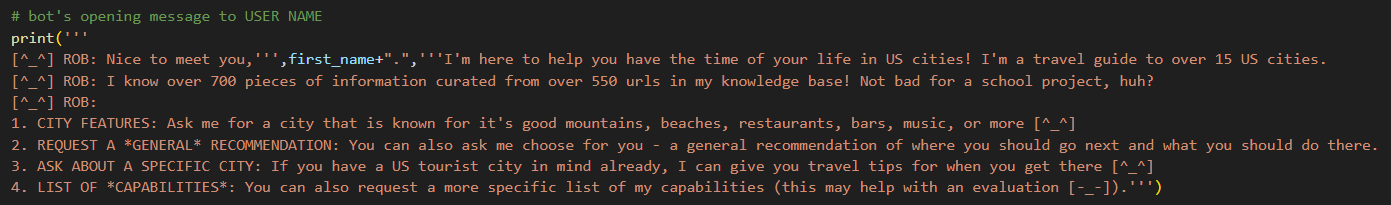


Figure 2: Opening message

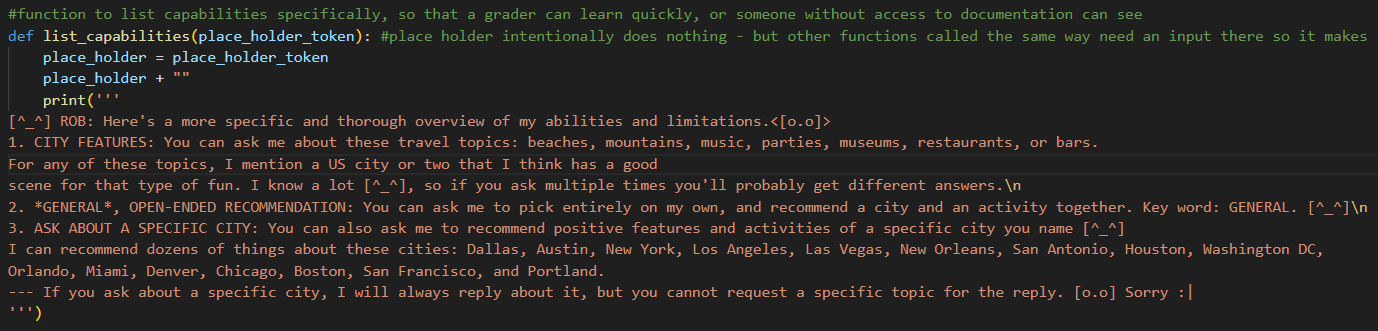


Figure 3: ROB’s more specific description of his own capabilities, provided if the user asks for it.

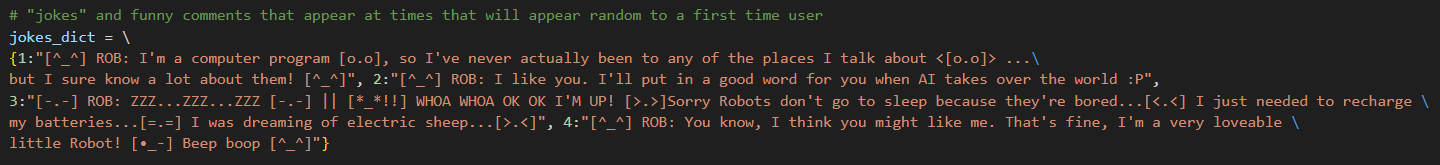


Figure 4: ROB will make jokes and funny comments at times that will appear random to a first time user.

In addition to the above, ROB also stores regularly changing transition word/phrase that he will insert in appropriate places (such as “Furthermore” or “To add to that”). ROB will also mention the user by name occasionally. These may seem small, but they really liven up the experience with the bot, and make it seem much more natural and less stale.

## Dialog Tree:

So, do you like it?

Chatbot will sometimes mention past likes at this point (not dislikes!)

Chatbot remembers cities visited, cities liked/disliked, activities liked/disliked

if no

if yes

Have you been to that city?

Do you like that activity?

Chatbot gives user general travel tips or lists capabilities.

Immediate return to prompt for input.

Chatbot gives user tips about a specific US city or travel activity.

User natural language input selecting US city, travel activity, “general”, or “capabilities”.

Chatbot prompts user to provide it with a travel question.

Chatbot gives short version of capabilities explanation.

User provides name.

CHATBOT START UP: Chatbot asks for user name.

## Sample Dialog Interactions:

*These dialog interactions exclude the opening message shown in Figure 2 to save space. User models for A – Albert and B – Benjamin are shown in Appendix 2.*

**Sample A - Albert:**

[^\_^] ROB: Ask me a travel question in natural language and press enter...

**I want to learn about Austin.**

[^\_^] ROB: Albert, Austin’s famous legacy of blues and soul music continues. On top of that, Located on South Congress, Joann’s is the perfect spot to experience Austin.

[^\_^] ROB: **Have you ever been to Austin?** (type your answer and press enter)

Yes I have

[^\_^] ROB: **Oh wow! Did you like Austin?** (type your answer and press enter)

Yes I did

[^\_^] ROB: Ask me a travel question in natural language and press enter...

**Tell me about cities with great music scenes**

[^\_^] ROB: Boston is a music-lover's dream come true, with dozens of venues that feature great local, regional, national, and global acts nightly. To add to that, New Orleans is also the birthplace of jazz, and live music of all kinds permeates the city, day and night.

[^\_^] ROB: **Do you like different types of music?** (type your answer and press enter)

Yes

[^\_^] ROB: I'm a computer program [o.o], so I've never actually been to any of the places I talk about <[o.o]> ...but I sure know a lot about them! [^\_^]

[^\_^] ROB: I hope I can show you more cities that you like as much as Austin![^\_^] ROB: I hope I can show you more places where you can enjoy different types of music!

[^\_^] ROB: Ask me a travel question in natural language and press enter...

**Tell me about New York**

[^\_^] ROB: You can also hike to Mount Marcy and the highest point in New York. Furthermore, From New York City to Boston, visitors witness memorable skylines and historical monuments commemorating the rich history of each city.

[^\_^] ROB: **Have you ever been to New York?** (type your answer and press enter)

No

[^\_^] ROB: Ask me a travel question in natural language and press enter...

**Sample B – Benjamin**

[^\_^] ROB: Ask me a travel question in natural language and press enter...

**What city has a good nightlife?**

[^\_^] ROB: They love to dress up and **party** here, so put on a wig and mask, and hit the French Quarter and the Marigny for fabulous food at the best New Orleans restaurants. And get this: But there’s more to Miami than just **partying** until the sun comes up.

[^\_^] ROB: Do you like parties? (type your answer and press enter)

yeah I love them

[^\_^] ROB: Ask me a travel question in natural language and press enter...

**What city is good for fine dining?**

[^\_^] ROB: The city of Denver offers many attractions, museums, concerts and **restaurants** for foodies. Additionally, Check out the Art Institute of Chicago, or one of the city's many acclaimed **restaurants**.

[^\_^] ROB: Do you like restaurants? (type your answer and press enter)

yes

[^\_^] ROB: I'm a computer program [o.o], so I've never actually been to any of the places I talk about <[o.o]> ...but I sure know a lot about them! [^\_^]

[^\_^] ROB: I hope I can show you more places where you can enjoy parties!

[^\_^] ROB: Ask me a travel question in natural language and press enter...

**general advice please**

[^\_^] ROB: The options are limitless in Los Angeles, where entertainment is quite literally a way of life. New Orleans is also the birthplace of jazz, and live music of all kinds permeates the city, day

and night. Miami is filled to the brim with famous nightclubs, Art Déco architecture, delicious food, white sand beaches, luxurious hotels, and a discernible Cuban influence.

[^\_^] ROB: Ask me a travel question in natural language and press enter...

# Likert-Style Survey

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ryan | James | Jordan |
| I felt that ROB was knowledgeable. | Strongly Agree | Strongly Agree | Agree |
| Conversations with ROB felt natural. | Strongly Agree | Agree | Disagree |
| Converations with ROB were enjoyable. | Strongly Agree | Agree | Neutral |
| I would recommend ROB to a friend. | Strongly Agree | Neutral | Disagree |

# Evaluation and Conclusions

Strengths:

* As with version 1, ROB has an impressively wide and deep knowledge base. The Naïve Bayes version of ROB is able to navigate that knowledge base using an even broader range of user inputs.
* The Naïve Bayes Classifier is a remarkably effective algorithm for classifying input, especially given its elegant simplicity
* If the user types the right requests, ROB can appear very knowledgable.
* ROB remembers where the user has traveled, what cities/activites they like, and ROB brings the likes up again in a naatural manner.
* ROB also features an emotive, charismatic face, and tells jokes!
* Particularly excellent for version 2 was ROB’s vastly expanded set of clue words, thanks to the huge data set
* Notably, ROB uses a Naïve Bayes Classifier that was written for this project, rather than from the SKLearn library.

Weaknesses:

* ROB’s algorithm remains incredibly simple – he can’t do consistent, long conversations outside of his highly limited structure
* ROB version 2 struggles with ambigious input.
  + While the previous version easily put all “Who knows what the user wants?” type requests to general, the Naïve Bayes Classifier version commonly thinks vague requests are a cue to re-list capabilities. Unfortunately, that’s very jarring for users!
* Sometimes, it’s painfully clear that ROB is clueless, and he looks like an idiot.
* Ultimately however, the biggest takeaway here is that the three most important things in an ML model are: data, data, data. The strenghts of the data set became strengths for ROB, but its weaknesses became his weaknesses.
  + Frankly, the Naïve Bayes algorithm, while far from perfect, is remarkably powerful, and has an incredible ability to capture relationships **when provided with a good or even decent data set**.
  + Unfortunately, I did not anticipate the challenges associated with generating quality data would be as severe as they were. In the future, I will put more time, energy, and focus, toward generating a high quality data set.

Please also see Appendix 1: First Attempt at ML Chatbot: Text Generation with Transformer and Subtitle Dataset

**Appendix 1: First Attempt at ML Chatbot: Text Generation with Transformer and Subtitle Dataset**

After 10+ hours I abandoned this approach.

# Key Takeaways:

* + While this portion of the project was challenging and discouraging, I learned some useful lessons:
    - I learned a ton about transformers, chatbots, and text generation
    - I got a lot of experience working with “no-handrails” transformer code
    - I got some experience learning how to let something go even after putting 10+ hours of work into it, which is clearly an important lesson
    - Perhaps the most important lesson here is: **first, go for the achievable, and get it working. Only reach for the above-and-beyond after you make something simple that works! That’s a lesson I’ll remember for a long time.**

# Details:

* I chose to do text generation, because it is interesting, exciting, and fun
* a fun data set – I chose “open subtitles”
  + 92.5K lines of dialog from roughly 75 different movies
  + <https://huggingface.co/datasets/open_subtitles>
* I wanted to do a transformer, since that is the most advanced, state-of-the-art tech available for natural language generation and chatbots
* Because I wanted to do a transformer, I chose to start with the code from the blog post provided in the assignment:
  + <https://blog.tensorflow.org/2019/05/transformer-chatbot-tutorial-with-tensorflow-2.html>
* Unfortunately, he did not provide all the code – importantly, he did not provide his pre-processing code
* While he claimed to link to his working model, the link was dead
* So, I got to work filling in his gaps
* While I doubt what he excluded was that important, it was difficult to replace without knowing what it was
* I wrote 100+ lines of code to replace his missing pre-processing, to write create\_padding\_mask and create\_look\_ahead\_mask, and to fix the decoder input for the transformer
  + For me, it was challenging because I was trying to fit my code to work with his several hundred lines of code – I didn’t want to change anything he did because I knew his model would work
* Ultimately, trying to fill in the code he was missing, while it might have been simple for some people, proved to be too challenging for me
* **After 10+ hours between data set selection, reading through his blog post to understand how his transformer would work, and finally writing 100+ lines of code to try and make his code work, I decided to abandon this approach.** The existing challenges to making his code work and anticipated future challenges with text generation were to great, considering the time and energy I was able to allocate to this project during the final weeks of the semester.
* ***The code for this portion is included in nlp\_proj\_2\_discarded\_attempt.ipynb***
* The code I wrote:
  + Everything from start to just before “def scaled\_dot\_product\_attention” (so, the first roughly 120 lines)
  + def create\_padding\_mask
  + def create\_look\_ahead\_mask
  + The rest of the code, except some tests and other minor things, is largely written by the tensorflow blogger Prof Mazidi linked us to in the original assignment

**Appendix 2: Sample User Models**

**Sample A – Albert**

{

"name": "Albert",

"has-been-to": [

"austin"

],

"hasnt-been-to": [

"nyc"

],

"likes-cities": [

"austin"

],

"dislikes-cities": [],

"likes-topics": [

"music"

],

"dislikes-topics": []

}

**Sample B – Benjamin**

{

"name": "Benjamin",

"has-been-to": [],

"hasnt-been-to": [],

"likes-cities": [],

"dislikes-cities": [],

"likes-topics": [

"party",

"restaurant"

],

"dislikes-topics": []

}

**Sample C – Justin12**

{

"name": "Justin12",

"has-been-to": [],

"hasnt-been-to": [

"vega",

"orleans",

"antonio",

"houston",

"washington",

"orlando",

"miami",

"denver",

"chicago",

"boston",

"francisco",

"portland"

],

"likes-cities": [],

"dislikes-cities": [],

"likes-topics": [],

"dislikes-topics": [

"beach",

"mountain",

"party",

"museum",

"restaurant"

]

}