Cady Baltz
CS 4395.001 Human Language Technologies
Portfolio Assignment: Finding or Building a Corpus
3/11/2023

### Introduction:

In this project, I designed a web crawler to find websites related to tourism in Japan. The goal of this program was to create a knowledge base of information that could be used to create a travel agent chatbot that can help answer questions about visiting Japan.

# **Knowledge Base Creation:**

#### 1) Finding 15 Relevant URLs

First, I found a website related to traveling in Japan that covered a wide range of information, from Japan's history and culture to its most popular tourist sites. Most importantly, this website contained external links that would allow me to further traverse the Internet for more information about this topic. The URL of this website is: <a href="https://en.m.wikivoyage.org/wiki/Japan">https://en.m.wikivoyage.org/wiki/Japan</a>.

Then, I created a web\_crawler() function to find fourteen additional relevant links branching off of my initial starter URL. Writing this function required a lot of trial and error in order to ensure that the websites I found contained useful information. To find links, I created a 'link\_queue' that initially only contained my starter URL. Then, I scraped the text off of this URL using the Python Beautiful Soup library, including all of its links. I added all of these links to the queue to continue searching indefinitely (although I added a cap of 1000 iterations to prevent infinite loops).

In order for a link to be added to my final 'related\_links' list, it had to satisfy the following requirements:

- 1. The URL could not be in my list of "blocked\_hosts".
  - a. After observing the output of my function, I noticed some websites that would not allow me to scrape their text. For example, certain websites included a "Prove you are not a robot" message. To handle these edge cases, I added the website's domain name to a manual list in my program each time I encountered one.
- It had to have a different hostname than all of the existing links in the array
  - a. This requirement helped ensure diversity in my webpage selection.
- 3. It had to be successfully scraped by Beautiful Soup.

- a. I used a try/except block to try scraping the website and check whether it was successful.
- 4. It had to contain at least 500 words of paragraph text (text within tags).
  - a. I specifically searched within the tags using Beautiful Soup to exclude webpage text that was not useful, like text from navigation bars and footers.
  - b. I included the 500 words requirement to ensure that I was scraping enough text to build a sizable knowledge base.
- 5. The text had to use ASCII characters.
  - a. I added this requirement after I found that a lot of the webpages being returned were written in Japanese.

I continued checking links from my queue using these requirements until I had fifteen valid links.

## 2) Cleaning the Raw Webpage Text

After extracting the text within the tags using Beautiful Soup, I saved this raw text to its own file in my 'raw\_page\_texts' directory. Then, I iterated through all of these raw text files to clean them in my clean\_text() function. In this function, I clean the text by removing extra whitespace, including new lines and tabs. Then, I used NLTK's sent\_tokenize() method to separate the raw text into individual sentences. Then, I iterated through each tokenized sentence and printed it on a line in a new file in my 'clean\_page\_texts' directory.

#### 3) Determining the Top 25 Terms

To determine the top 25 terms, I applied simple term frequency when analyzing my documents. In my get\_top\_25\_terms() method, I iterated through each cleaned text file from the previous step. Then, I tokenized each individual sentence using the NLTK word\_tokenize() method. Finally, I iterated through each token and incremented its count in a dictionary I created.

To ensure my top 25 terms were useful, I applied the following requirements to them:

- 1. They are alphabetic
- 2. They are not in the English stop words list from NLTK
- 3. They have a length greater than five characters

Then, I simply sorted my count dictionary and returned the 25 terms with the greatest counts across all of my documents.

#### 4) Selecting the Top 10 Terms

After finding my top 25 terms, I printed them all to console and manually observed them. I selected ten terms that I thought would be useful for a chatbot travel agent to recognize, and that I thought would result in substantial information in my knowledge base.

#### Selected terms:

- 1. japanese
- 2. country
- 3. english
- 4. popular
- cities
- 6. hotels
- 7. restaurants
- 8. travel
- 9. international
- 10. stores

#### 5) Creating the Knowledge Base

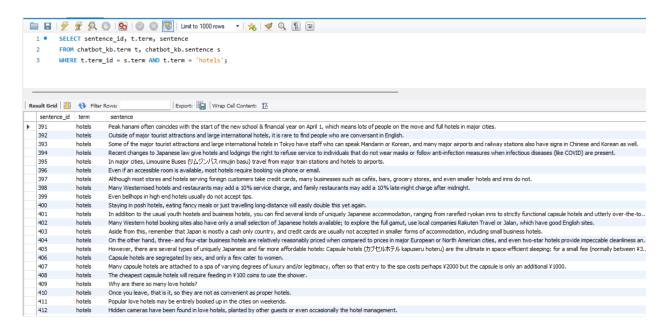
Once I had my top 10 terms, I searched for information related to these terms in my cleaned text files. I created a new dictionary with 10 keys – one for each term listed above. Then, I iterated through each sentence of each cleaned text file. If the sentence contained one of the terms listed above, I included it in the corresponding entry of the dictionary. Then, I dumped this new dictionary into a pickle file named "chatbot kb.p".

### 5) Creating a SQL Database

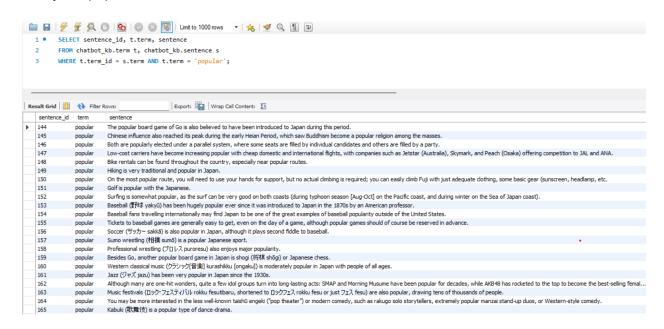
To further improve the usability of my knowledge base, I created a SQL database for it. The database consists of two tables: term and sentence. The 'term' table contains the 10 terms listed above, as well as a unique ID for each. The 'sentence' table contains pieces of knowledge that map to these terms. Each entry of the 'sentence' table includes a unique ID, a sentence of information (with a maximum length of 1000 characters for handling storage concerns), and a foreign key that maps to the 'term' table.

By arranging my data in a SQL database, I can easily query it and view the information. In total, my database contains 871 entries. On the next page, you can find screenshots of my SQL knowledge base showing the execution of a few different queries.

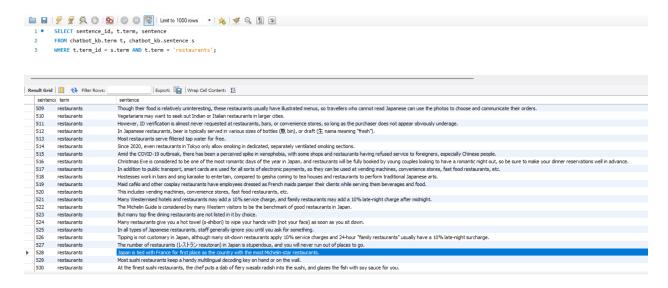
#### Query for 'hotels':



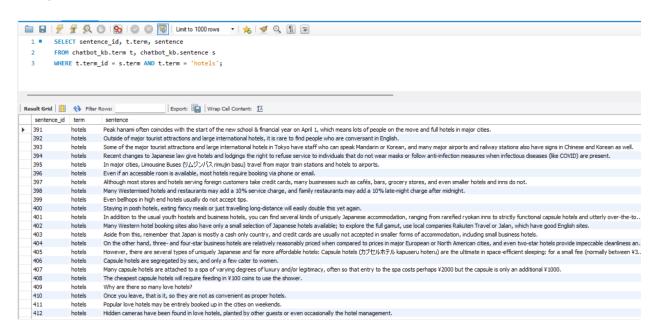
#### Query for 'popular':



#### Query for 'restaurants':



#### Query for 'hotels':



#### 6) Sample Dialog

Using pieces of knowledge I found in my database, here is what I envision could be a chatbot conversation. All of the Chatbot responses come directly from my knowledge base with no modification of the text. I also highlighted the keywords that the user said that would help the chatbot form its answer using my knowledge base.

Chatbot: Hello, how can I help you today?

User: I want to plan a trip to Japan! What is the best way to **travel** there?

Chatbot: Delta Air Lines, United Airlines, and American Airlines also operate flights from

numerous U.S. cities into Narita or Haneda, as does Air Canada from several

Canadian cities.

User: What is the best way to **travel** around Japan?

Chatbot: A tourist who plans to travel a great deal around the country should consider

investing in a Japan Rail Pass, which offers — with a few exceptions — unlimited travel on all Japan Railway (JR) services including bullet trains, limited express

and regular commuter trains.

User: What are the most **popular** things to do in Japan?

Chatbot: Hiking is very traditional and popular in Japan.

User: Hiking sounds fun! Where should I hike?

Chatbot: On the most popular route, you will need to use your hands for support, but no

actual climbing is required; you can easily climb Fuji with just adequate clothing,

some basic gear (sunscreen, headlamp, etc.

User: What about food? Does Japan have good **restaurants**?

Chatbot: Japan is tied with France for first place as the country with the most Michelin-star

restaurants.

User: That sounds great! I want to go to Japan now. Can you help me book a **hotel**?

Chatbot: Some major chains of cheaper business hotels include Tokyu REI Hotels, known

for its generously sized rooms, Sunroute Hotels and Toyoko Inn. Even if an accessible room is available, most hotels require booking via phone or email.