Alejo Vinluan (abv210001) **Thanh Vo (ttv170230)**

CHATBOT FINAL REPORT

Index

1. Overview

The goal of this assignment was to create a Chatbot while utilizing NLP techniques that were learned in class. The Chatbot must carry on a conversation within the domain while using knowledge from the user and knowledge from the web.

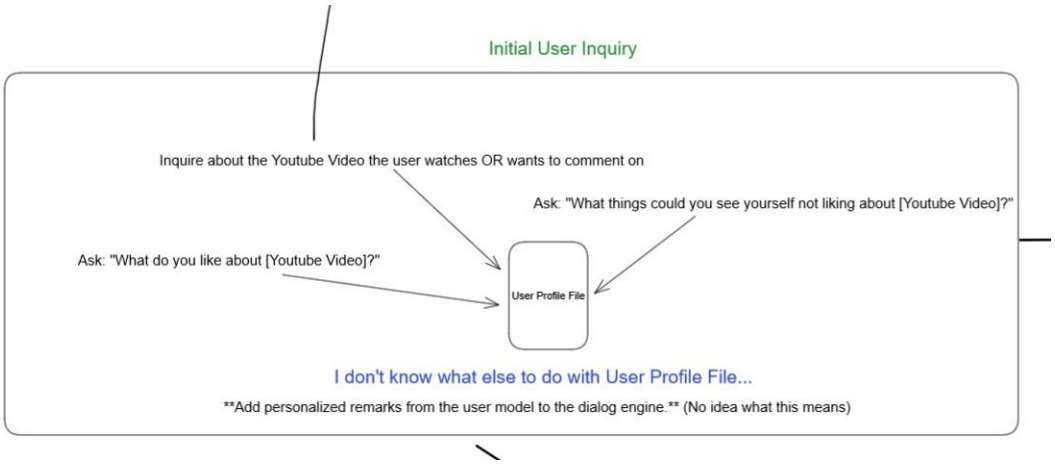
Initially our idea was to create a Chatbot that would scrape Basketball statistics from the website Basketball Reference. This Chatbot would give purely statistical facts from previous NBA games and about individual NBA players. However, this Chatbot would not be interactive and could not carry on limited conversation. We evolved this idea with a Minecraft Chatbot that would scrape data from the Minecraft wikipedia page. The same issue occurs where this bot would only give facts about Minecraft rather than have the components necessary to carry an engaging conversation with the user. In our next iterative development cycle, we came up with the idea of a more generalized Minecraft Youtube Chatbot, where the bot could comment on Minecraft Youtube videos. After contemplating this idea, it dawned upon us to widen the expanse of our Chatbot implementation and take into account any available Youtube video.

We decided to model our Chatbot to be a Youtube conversationalist. The user can ask about a particular topic or link a Youtube video. Then, the Chatbot could carry a conversation about the topic or video with the user, while having the capability to generate human-like insight and analysis.

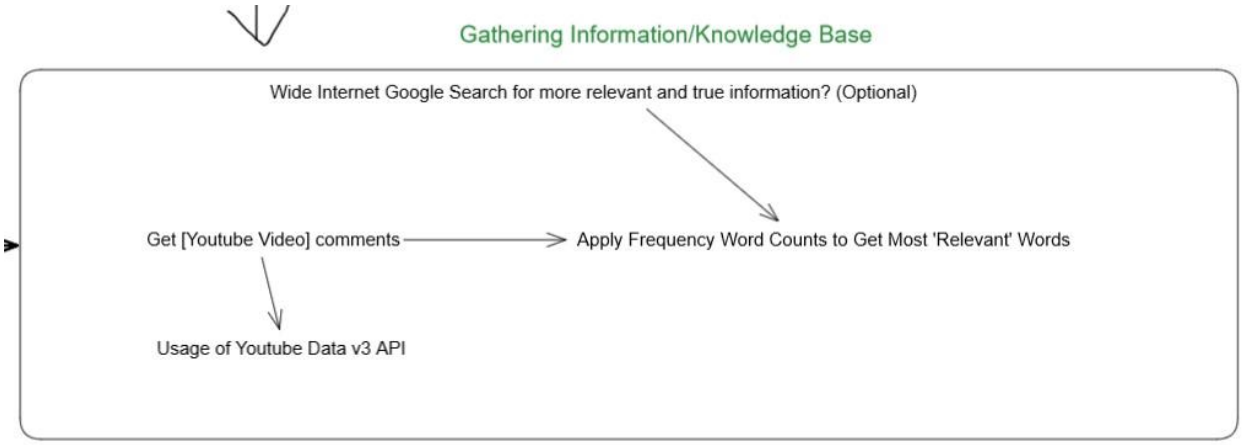
2. System Description

The system went through multiple iterations before we made our final Chatbot. In the most abstract sense, the system involves 3 main steps: Initial User Inquiry, Gathering Information, and Response Generation.

2a. Youtube Chatbot Version 1

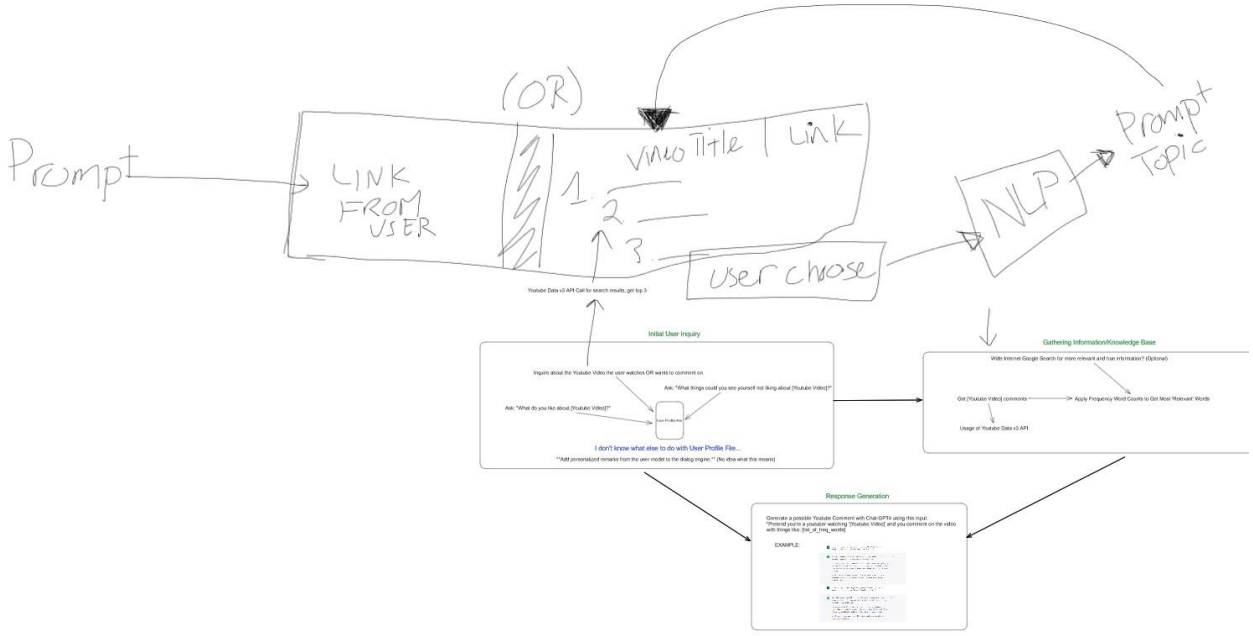


The Initial User Inquiry would have the user give their name, the topic, what they like about the topic, and what they dislike about the topic. This establishes a baseline to train the Chatbot about the user.

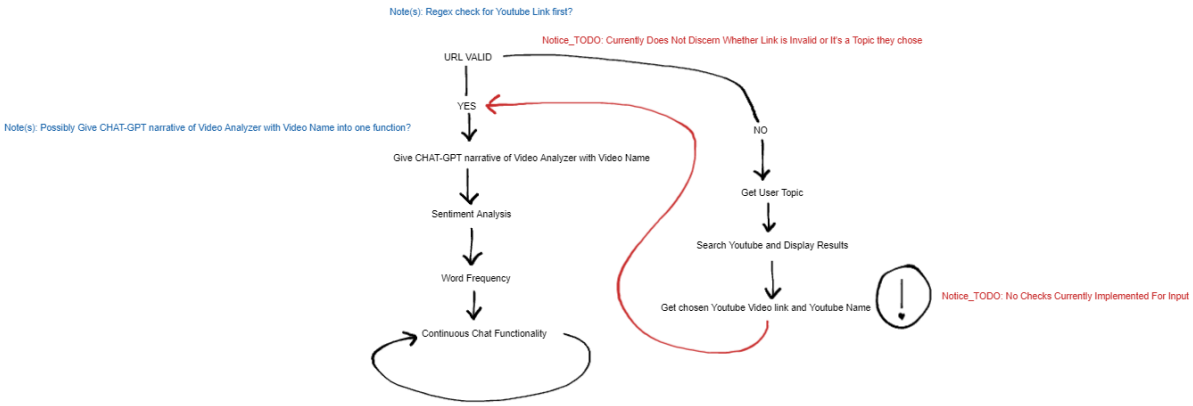


The Gathering Information and Knowledge Base portion of our Version 1 Chatbot involved the idea of manually scraping a Youtube video and grabbing the comments. We would be able to apply Word Frequency and find the most relevant words to that video.

Response Generation would finally train the model with the data given above and generate a response for the user. Our model could carry on the conversation, but there wouldn't be any way for the user to discuss other topics or give new Youtube links for the user.



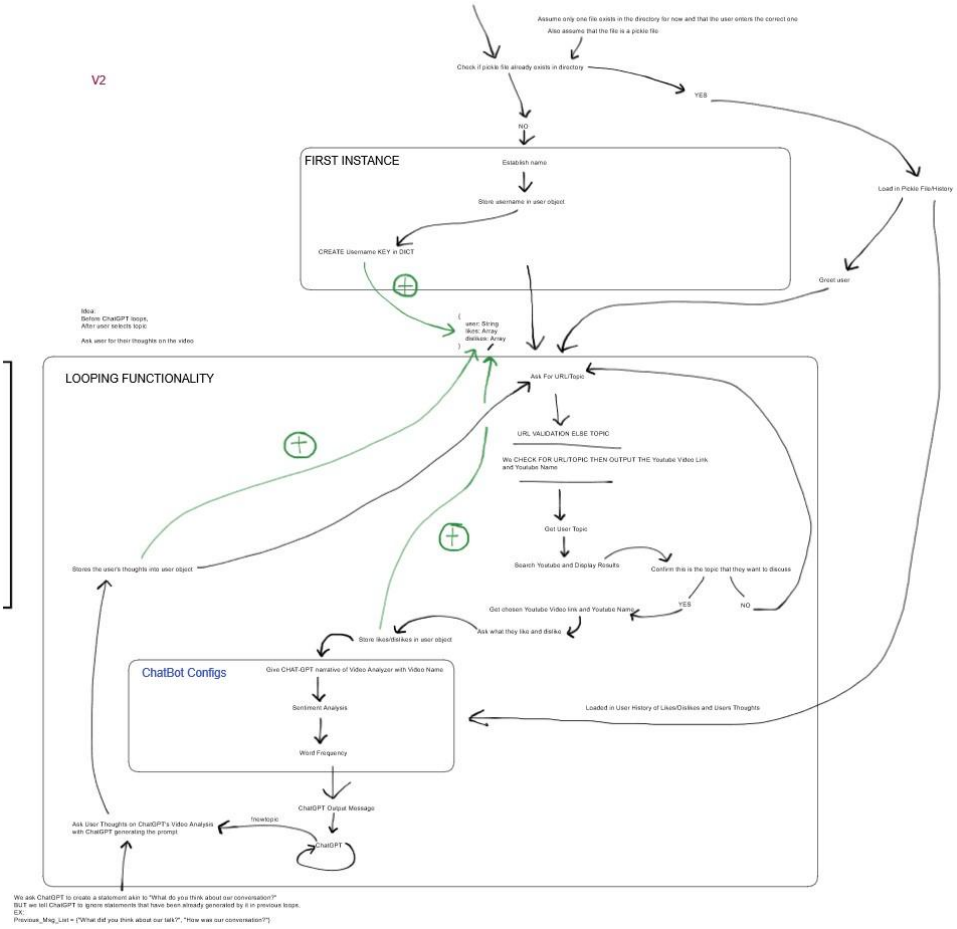
2b. Youtube Chatbot Version 2



Version 2 is a more defined and layed out implementation of our initial model -- Version 1. It expands to the control structures necessary to provide clarity to *developers* and also gives a clear pathway that any layman could intuitively. As you can see it still lacks a lot of features and implementation details necessary for the Chatbot but generally gives a more refined understanding of the various states the program would be at given initial conditions and appropriate context.

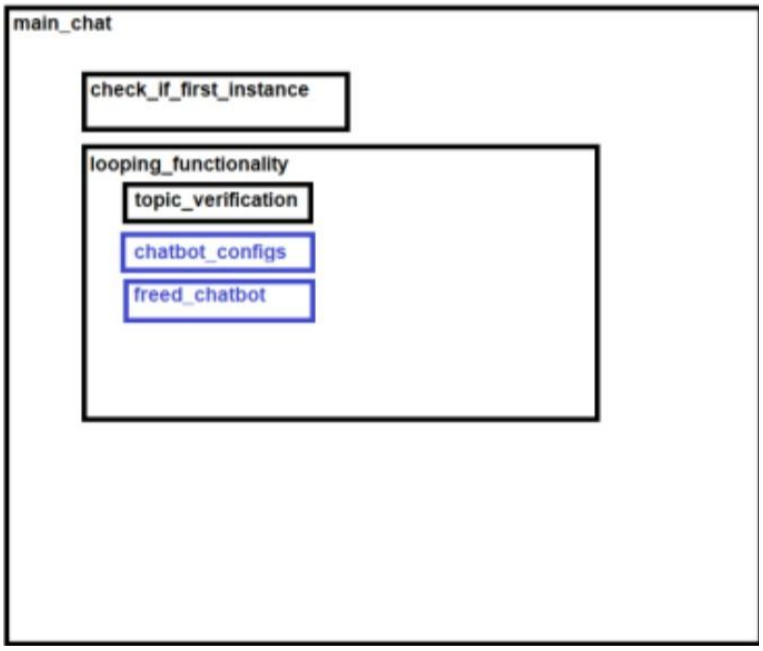
2c. Youtube Chatbot Version 3

We expanded upon Version 2 by hashing out the details for certain features such as names, likes, and dislikes. At an even lower level of abstraction, we made sure to detail the execution of several minute features that have incredibly powerful implications in the overall functionality of our Chatbot. For example, the user would have a saved profile so that the Chatbot would recognize your name, likes, and dislikes the next time you open the program. Furthermore, the user could change the topic after they finish the conversation with the Chatbot. Lastly, the user would also be able to specify any topic and choose amongst a selection of videos rather than having the Chatbot exclusively look at youtube links. This enables the chatbot to give more choice and freedom to the user and lets the finality of a conversation topic be up to the user entirely.



Notice that here we are starting to get even less into the abstraction and more into the details by having various loops defined that will enhance and define the behavior of the Chatbot. (IGNORE THE TEXT LABELED V2)

3. Explanation on Subdivision of Components



This is meant to show the initial functional structure of our actual program. Although the structure is still technically maintained, there are additional pieces that have since been added on to ensure the quality of our Chatbot.

3a. main_chat

main_chat would encompass the entire program so that the user can run “main.py”, which would import all of the other necessary files from the same directory.

3b. check_if_first_instance

check_if_first_instance checks whether or not a User File exists. If it doesn’t, a user file is created in the same directory for the user. If it does, then the user file is loaded and used.

```
Youtube Bot: What's your name?
User: Alejo
Youtube Bot: Hello Alejo!
```

3c. looping_functionality

looping_functionality is the main chat functionality of the Chatbot. This is subdivided into three separate functions: topic_verification, chatbot_configs, and freed_chatbot.

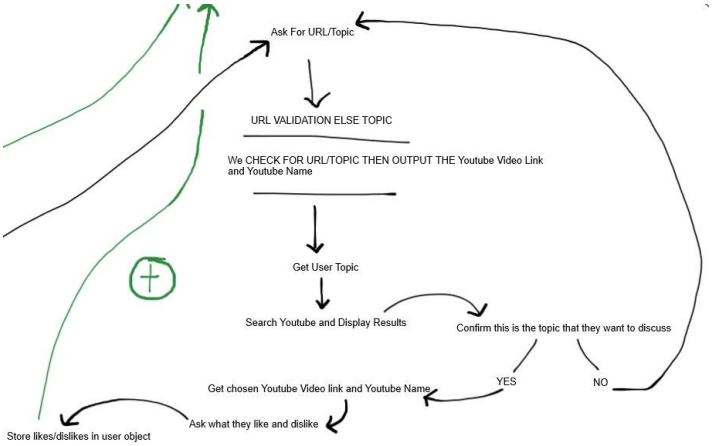
3d. topic_verification

topic_verification is the true beginning of the chat functionality. In this instance, the Chatbot will ask the user for a topic or a Youtube link. If a topic is provided, a user will have a choice in 5 videos. The user must choose one of the 5 videos to discuss. If a Youtube link is provided, this step is skipped and the Chatbot will go directly to asking about the user's likes and dislikes.

```
Youtube Bot: What do you want to discuss?
Youtube Bot: Please enter JUST the topic or link.
Alejo: Natural Language Processing
Youtube Bot: Please Pick a Video From The Following Listed Below.
1. Natural Language Processing In 5 Minutes | What Is NLP And How Does It Work? | Simplilearn | https://www.youtube.com/watch?v=CMrHM8a3hqw
2. Natural Language Processing: Crash Course Computer Science #36 | https://www.youtube.com/watch?v=fOvTtapxa9c
3. What is NLP (Natural Language Processing)? | https://www.youtube.com/watch?v=fLvJ8VdHLA0
4. Natural Language Processing with spaCy & Python - Course for Beginners | https://www.youtube.com/watch?v=dIUTsFT2MeQ
5. Natural Language Processing: Crash Course AI #7 | https://www.youtube.com/watch?v=oi0JXuL19TA
Youtube Bot: Please enter the video integer number.
```

The bot will then ask about the user's likes and dislikes. These likes and dislikes will be used to feed into the Chatbot model.

```
Youtube Bot: What do you like about: Natural Language Processing In 5 Minutes | What I
s NLP And How Does It Work? | Simplilearn?
Alejo: The science behind languages
Youtube Bot: What do you dislike about Natural Language Processing In 5 Minutes | What
 Is NLP And How Does It Work? | Simplilearn?
Alejo: It seems difficult
```

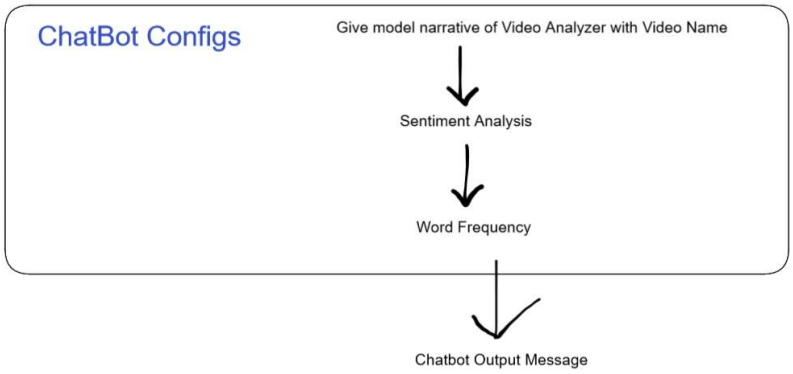


Here we show a piece of the topic verification from our larger scale model from the top of the page.

```
def topic_verification(self):
    """
    The user either enters a URL or topic. If it is not a working URL then we force them to enter a URL or topic.
    Otherwise, we immediately assume it's a topic.
    """
    print("Youtube Bot: Please enter JUST the topic or link.\n")
    # FOR TESTING PURPOSES:
    # self.user_name = 'Tester'
    user_input = input("{}: ".format(self.user_name))
    print()
    if user_input == '!exit':
        print("Thanks for talking to Youtube Bot :)\n")
        sys.exit(0)
    while True:
        if re.search("youtu.be\/|youtube.com\/|https", user_input):  # Check if the user's input has a link
            if self.yt.verify_url(user_input):  # Verify the URL works
                if not self.yt.verify_comments_enabled(self.yt.get_video_id(urlparse(user_input))):
                    print("Youtube Bot: It seems like the video you've chosen has comment scraping disabled. Please pick another.\n")
                    user_input = input("{}: ".format(self.user_name))
                    print()
                    continue
                return user_input, self.yt.get_video_name(user_input)
            else:
                print("Youtube Bot: WHOOPS. Your URL does not seem to work. Please try again or enter a topic.\n")
                print("Youtube Bot: Please ensure the link does not contain timestamps or anything after the Video ID.\n")
                user_input = input("{}: ".format(self.user_name))
                print()
        else:  # VIDEO SEARCH FROM TOPIC
            videos_link_arr, videos_title_arr = self.yt.get_topic_list(user_input)

            print("Youtube Bot: Please Pick a Video From The Following Listed Below.\n")
            for num in range(len(videos_title_arr)):
                print(str(num + 1) + '. ' + html.unescape(videos_title_arr[num]) + ' | ' + videos_link_arr[num])

            print("Youtube Bot: Please enter the video integer number.\n")

            while True:  # Continuous checks to see if the input is correct
                selected_vid_index = input("{}: ".format(self.user_name))
                print()
                if selected_vid_index == '!exit':
                    print("Thanks for talking to Youtube Bot :)\n")
                    sys.exit(0)
                if selected_vid_index.isdigit():
                    selected_vid_index = int(selected_vid_index)
                    if 0 < selected_vid_index < 6:  # Checks if in the range of 1, 5 inclusive
                        selected_vid_index = selected_vid_index - 1
                        if not self.yt.verify_comments_enabled(self.yt.get_video_id(urlparse(videos_link_arr[selected_vid_index]))):
                            print("Youtube Bot: It seems like the video you've chosen has comment scraping disabled. Please pick another topic.\n")
                            break
                        return videos_link_arr[selected_vid_index], videos_title_arr[selected_vid_index]
                    else:
                        print("Youtube Bot: Please enter a listed video number.\n")
                else:
                    print("Youtube Bot: Please enter an INTEGER number.\n")
```

3e. chatbot_configs

chatbot_configs() would take data from multiple sources and feed them into the model for training. This data includes the video name, a list of the most common words commented under the video, and a sentiment analysis of the comments under the video. The Chatbot would then produce a Chatbot given this information.




```
def chatbot_configs(self, link):
    # Get the comments from the video
    comments = self.yt.comment_finder(link)
    # clean_comments type list
    clean_comments = self.nlp.clean_text_array(comments)
    comment_freq = self.nlp.word_frequency(clean_comments)

    # Setup chatbot to discuss current video
    # Give the chatbot the title of the video
    self.message_log.append(
        {"role": "system", "content": "You are a bot that pretends to give analysis on Youtube videos."})
    self.message_log.append({"role": "system",
                             "content": "You do not need to have knowledge on the videos, just give an analysis based on sentiment score and title."})
    self.message_log.append({"role": "system",
                             "content": "Also, try to utilize keywords that relate to the video in your response."})
    title_message = "The video you are currently discussing is called " + self.yt.get_video_name(link)
    self.message_log.append({"role": "system", "content": title_message})

    # Give the user feedback on the video
    user_likes = "The user likes this about the video " + self.user_data.likes[-1]
    user_dislikes = "The user does not like this about the video " + self.user_data.dislikes[-1]
    self.message_log.append(
        {"role": "system", "content": user_likes})
    self.message_log.append(
        {"role": "system", "content": user_dislikes})

    # Give the chatbot the sentiment analysis of the comments
    sentiment_score = self.nlp.sentiment_analysis(" ".join(comments))
    sentiment_message = "The video you are currently discussing has a sentiment score of " + str(
        sentiment_score[0]) + " percent negative, " + str(sentiment_score[1]) + " percent neutral, and " + str(
        sentiment_score[2]) + " percent positive."
    self.message_log.append({"role": "system", "content": sentiment_message})

    # Give the chatbot the most frequent words in order
    common_words = []
    for key, value in sorted(comment_freq.items(), key=lambda x: x[1], reverse=True):
        common_words.append(key)

    # Send the bot the 100 most common words
    common_words_message = "The most common words commented under the video are "
    for i in range(0, min(len(common_words), 100)):
        common_words_message += common_words[i] + " "
    self.message_log.append({"role": "system", "content": common_words_message})
    self.message_log.append(
        {"role": "system", "content": "Generate your first response, given the information above."})
    bot_message = self.chat.generate_message(self.message_log)
    self.message_log.append({
        "role": "assistant",
        "content": bot_message
    })
    print("YouTube Bot:", textwrap.fill(bot_message, 60), "\n")
    return
```

3f. freed_chatbot

freed_chatbot() takes in the textual output that ChatGPT was given in the ChatBot Configs file. As previously mentioned the ChatBot Configs function, we provide sentiment analysis from various NLP techniques we learned in class and then ChatGPT spits out a narrative in which the user can see. From there, ChatGPT loops in which the user can freely interact with the Chatbot and ChatGPT was provided with all of the external data extracted from each of the knowledge bases.

4. NLP Techniques

4a. Clean Text

The first NLP technique utilized was cleaning the provided text from the Youtube comments. The following steps are taken in order to clean the text:

1. Remove all newline and tab characters
2. Remove non-alphanumeric characters from the text
3. Lower the text
4. Use NLTK's word tokenizer to split the words
5. Lemmatize the individual words to get their base word
6. Remove non-English words from the text
7. Remove stopwords using NLTK's English stop words

The text is properly filtered for processing to use within the other NLP techniques.

4b. Word Frequency

Word Frequency is utilized in order to train the model on the most frequent words that appear within the comment section of the Youtube video. Word Frequency enables the Chatbot to use some of the common words as well as give the Chatbot the general topic and idea of the video.

```
# Input: list of comments
# Output: dictionary of word counter
def word_frequency(self, comment_list):
    word_dict = {}
    for comment in comment_list:
        for word in comment:
            if word not in word_dict:
                word_dict[word] = 1
            else:
                word_dict[word] += 1
    return word_dict
```

4c. Sentiment Analysis

Sentiment Analysis takes in an input of the cleaned Youtube comments and outputs an overall sentiment score of the comments. It splits each of these words into 3 categories: negative, neutral, and positive. This function can give the Chatbot a reference on how other people view the video.

For example, the statement “Hello! I hate people. I like puppies.” returns the following sentiment score:

```
# Input: String of clean words
# Output: Sentiment score
def sentiment_analysis(self, words):
    sa = SentimentIntensityAnalyzer()
    score = sa.polarity_scores(words)
    return [score['neg'] * 100, score['neu'] * 100, score['pos'] * 100]
```

```
a = nlp.sentiment_analysis("Hello! I hate people. I like puppies.")
print(a)
```

```
[42.1, 31.6, 26.3]
```

The overall statement is 42.1% negative, 31.6% neutral, and 26.3% positive. This can give the Chatbot an overall context on whether or not commenters enjoyed the video.

Combining all 3 of the NLP Techniques listed above can be used to train the Chatbot by allowing the model to assume correlation. If the most common word of a comment section is “cats” and there is an overwhelmingly positive sentiment analysis in the comments, the model can assume people generally like cats.

5. Live Lookup of the Data (Knowledge Base)

We utilize Youtube Data API v3 to quickly lookup the data from Youtube. By using an API, we were able to avoid expensive scraping techniques that would take a long time to process data. The *YoutubeToolkit.py* file displays the techniques in which we were able to get data from the API.

We first have to utilize the Knowledge Base if a user wants to talk about a topic. Data is retrieved using the topic as a search term on Youtube. The first 5 results of the search term, along with their link, is then displayed for the user to select one video.A

```
Youtube Bot: Please enter JUST the topic or link.

Alejo: cats

Youtube Bot: Please Pick a Video From The Following Listed Below.

1. Funny Dog And Cat  Funniest Animals #223 | https://www.youtube.com/watch?v=4P
FjWTusngo
2. Cat TV Videos ~ Birds and Squirrels Picnic for Cats to Watch ⋆ 8 HOURS ⋆ | https:
//www.youtube.com/watch?v=X405vBJS2Q0
3. FUNNY CAT MEMES COMPILATION OF 2023 V14 | https://www.youtube.com/watch?v=ma-h5t_UY
W0
4. Music Video - Cats | https://www.youtube.com/watch?v=_yqSbnbUsj4
5. 1 HOUR FUNNY CATS COMPILATION 2022| Cute And Lovely Cat Videos 2022 | https://w
ww.youtube.com/watch?v=rqS2vFuU6SE
```

We then utilize that the user-provided URL or the chosen video has the API enabled and that we are able to retrieve all of the comments.

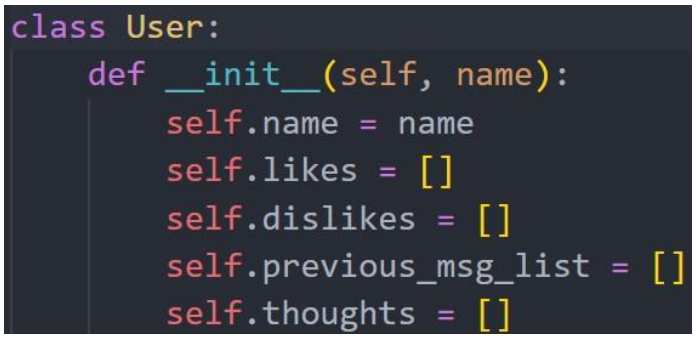
```
def verify_comments_enabled(self, video_id):
    try:
        comment_response = self.youtube.commentThreads().list(
            part='snippet',
            videoId=video_id,
            maxResults=100
        ).execute()
        return True
    except:
        return False
```

```
# Verifies the link provided is from YouTube
def verify_url(self, url):
    if re.search('(https:\/\/)?(www\.)?youtube\.com\/watch\?v=[a-zA-Z0-9\-\_]{11}$|(https:\/\/)?(www\.)?youtu\.be\/[a-zA-Z0-9\-\_]{11}', url):
        return True
    return False
```

After verifications are done, we fetch data from the video the user has chosen including, get_video_name(), get_video_id(), get_topic_list(), and comment_finder(). Each of these functions reflect their names by utilizing the API to fetch this data. comment_finder() will return a list of comments under the video that need to be processed and cleaned before utilizing NLP techniques on them.

6. User Models

The User Model created by the program is used to save the user's name, likes, and dislikes that is extracted from their input. This allows the Chatbot to load previous conversations it's had with the user anytime the user returns to the program.



```
class User:
    def __init__(self, name):
        self.name = name
        self.likes = []
        self.dislikes = []
        self.previous_msg_list = []
        self.thoughts = []
```

previous_msg_list is a chat history that is stored within the User object. This gives the Chatbot context about its previous conversations with the user. thoughts() are thoughts that the Chatbot has given the user. This is stored so that there is variation in how the Chatbot can ask the user to give their thoughts.

7. Analysis of the Chatbot

7a. Strengths

- It is able to continue the conversation in a way that does not make it seem too robotic in its mannerisms. By implementing ChatGPT into the codebase for dialogue, we are able to extract more human-like statements for different contexts and purposes seamlessly.

- The ChatBot can manage various forms of input through input checking such that the user can easily understand the logical pathing and has several choices that they can go back and forth between i.e. topic choosing, topic verification, new topic command, and exit command.

- The ChatBot is also written with a structured model so that if any code changes are deemed necessary to the codebase, it would be very efficient to manipulate the code.

- The ChatBot can take in live data from the Youtube API and is able to handle dynamic prompts at certain instances in the chatbot model such as freed_chatbot().

7b. Weaknesses

- The conversations can feel repetitive at times. The functionality of the Chatbot is purely to provide feedback about Youtube videos. However, the Chatbot cannot watch the videos and so cannot give context from within the video. The data fed into the Chatbot is purely from the comments section.

- The chatbot is restricted to the comment section. If a video does not have a comment section or the API is disabled by the video creator, the Chatbot cannot discuss the video.

7c. Overall Analysis

Overall, the Chatbot performs its functions well. The Chatbot can give a general idea of how the video was received by viewers. There is a lack of depth from the analysis given by the Chatbot since the analysis is generated by only the comments of the video. However, the Chatbot can carry on a general conversation with the user and interpret the user's opinion given the context.