```
#
# CS4395-001
# Professor Karen Mazidi
# April/15/2023

import nltk
import string
import random
import warnings
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from nltk.stem import WordNetLemmatizer

warnings.filterwarnings('ignore')

nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')

# Possible ways the user can interact with the chatbot
greetings_from_user = ("hi", "hello", "hey", "greetings", "hey there", "hola", "yo", "howdy")
user_likes = ['my favorite', 'i like', 'i do like', 'i did like',
              'my most favorite', 'i love', 'i do love', 'i did love',
              'i enjoy', 'i enjoyed', "I'm a fan of", 'I was a fan of', 'i liked',
              'i really liked', 'i really loved', 'i really enjoyed', 'i was really a fan of',
              'i was a fan of', 'i thought', "i didn't think", "i think", "i don't think", "i do think"]
user_dislikes = ['my least favorite', 'i dislike', "i don't like", "i didn't like",
                 'i hated', 'i hate', "i don't love", "i didn't love", "i didn't enjoy",
                 "i don't enjoy", "i'm not a fan of", "i wasn't a fan of", 'i really disliked',
                 'i really hated', "i really don't enjoy", "i wasn't really a fan of", "i don't care for",
                 "i didn't care for", "i'm not crazy about", "i disliked", 'i thought', "i didn't think",
                 "i think", "i don't think", "i do think"]

# Some responses the chatbot can give depending on user input
chatbot_greeting_responses = ["hi", "hello", "hey there", "hello world, I mean, hi there", "*nods*",
                              "hola (that's the only thing I can say in Spanish by the way)",
                              "hello! I'm happy to have someone to talk to!"]
chatbot_name_responses = ["oh, that's a wonderful name!", "eh, I've heard better names",
                          "that's nice! my name is zero", "that's a cool name!",
                          "that's an...interesting...name", "i love that name!",
                          "i hate that name", "that's a nice name, great to be speaking with you"]
chatbot_like_responses = ["i was a fan of that too!", "eh, not my favorite thing about the movie",
                          "that's my favorite thing about the movie too!", "i really enjoyed that too",
                          "i also liked it a lot", "i was a fan of that too", "that's cool! i liked that too",
                          "that's awesome!, i'm glad you liked that part", "coolio, but i don't care"]
chatbot_dislike_responses = ["i didn't really like that either", "really?! i liked that!",
                             "that was actually my least favorite part about the movie", "i hated that too!",
                             "really? i loved that!", "you sure have...opinions...", "that's unfortunate to hear",
                             "your opinion is not valid, i'm afraid", "interesting...opinion",
                             "i didn't care for it either, to be honest", "that sounds like an unpopular opinion",
                             "i hear you, i understand you, and now i condemn you", "coolio, but i don't care"]

# Read in the knowledge base with facts about 'The Nightmare Before Christmas'
with open('knowledge_base.txt', 'r', encoding='utf8', errors='ignore') as file:
    raw_text = file.read()

# Tokenize the raw token into sentences and individual words
raw_text = raw_text.lower()
word_tokens = nltk.word_tokenize(raw_text)
sentence_tokens = nltk.sent_tokenize(raw_text)

# A dictionary that contains punctuation
punctuation_dict = dict((punctuation, None) for punctuation in string.punctuation)
# To help lemmatize words
wnl = WordNetLemmatizer()


def preprocessing(user_input):
    """
    Performs some preliminary preprocessing on the user input (mainly lemmatizing tokens)
    :param user_input: The user's input/comment/question
    :return tokens_lemmatized: A list of lemmatized tokens
    """
    word_tokens = user_input.lower().translate(punctuation_dict)
    word_tokens = nltk.word_tokenize(user_input)
    tokens_lemmatized = [wnl.lemmatize(token) for token in word_tokens]
    return tokens_lemmatized


def chatbot_greetings(user_input):
    """
    Randomly picks a greeting if the user greetings the chatbot
    :param user_input: The user's input/comment/question (in this case, a greeting like 'hello' or 'hey')
    :return random_greeting: A randomly chosen gretting from a list of greetings
    """
    for word in user_input.split():
```

```
        if word.lower() in greetings_from_user:
            random_greeting = random.choice(chatbot_greeting_responses)
            return random_greeting


def chatbot_response(user_input):
    """
    Creates the response of the chatbot given user's input
    :param user_input: The user's input/comment/question
    :return response_from_chatbot: The appropriate response generated by the chatbot & found in the
    knowledge base. The chatbot should print some 'error' message if no appropriate response was found in the KB.
    """
    # Append user input to sentence tokens
    sentence_tokens.append(user_input)

    # Set up a Tfidf vectorizer and fit to the sentence tokens
    # (which includes the user input)
    tfidf_vectorizer = TfidfVectorizer(stop_words='english', tokenizer=preprocessing)
    tfidf = tfidf_vectorizer.fit_transform(sentence_tokens)

    cosine_value = cosine_similarity(tfidf[-1], tfidf)  # Get the cosine similarity value
    flattened_cosine_value = cosine_value.flatten()  # Flatten cosine values to be in one dimension
    flattened_cosine_value.sort()  # Sort cosine values
    highest_tfidf_value = flattened_cosine_value[-2]  # Get the highest cosine value (aka the highest tfidf value)

    # Index value of the response from knowledge base
    index_in_knowledge_base = cosine_value.argsort()[0][-2]

    response_from_chatbot = ""

    # If the tfidf value != 0, then there's an appropriate response found in the knowledge base.
    # In that case, respond with that response.
    if highest_tfidf_value != 0:
        response_from_chatbot = response_from_chatbot + sentence_tokens[index_in_knowledge_base]
        return response_from_chatbot
    # If the tfidf value == 0, then there wasn't an appropriate response found in the knowledge base.
    # In that case, respond with an 'error' message.
    elif highest_tfidf_value == 0:
        response_from_chatbot = response_from_chatbot + "Sorry, I didn't understand you. Please try again."
        return response_from_chatbot


def get_fun_fact():
    """
    Returns a fun fact if the user prompts one.
    :return random_fact: A random fun fact from the knowledge base.
    """
    # Read in the knowledge base
    with open('knowledge_base.txt', 'r', encoding='utf8', errors='ignore') as file:
        raw_text = file.read()
    file.close()

    # Get sentences from knowledge base
    sentences_from_kb = nltk.sent_tokenize(raw_text)

    # Create a list of fun facts and fill it with fun facts from knowledge base
    list_of_fun_facts = []
    for sentence in sentences_from_kb:
        if "Fun fact:" in sentence:
            list_of_fun_facts.append(sentence)

    # Randomly choose a fun fact and return it
    random_fact = random.choice(list_of_fun_facts)

    return random_fact


if __name__ == "__main__":

    print("Hi, my name is Zero and I'm a chatbot that has knowledge about"
          " the movie 'The Nightmare Before Christmas'!\nAsk me a question about the movie"
          " and I can answer it for you! If you want a fun fact about the movie, just type in 'fun fact'!\n"
          "To end conversation with me, just type 'bye' or 'goodbye'.")

    chatbot_running = True

    while chatbot_running == True:

        # Get user input
        input_from_user = input()
        input_from_user = input_from_user.lower()

        # If the user doesn't say bye to the chatbot, keep talking with them
        if 'bye' not in input_from_user and 'goodbye' not in input_from_user:
            # Response if user thanks chatbot
            if 'thank you' in input_from_user or 'thanks' in input_from_user:
                print("Zero: You're welcome!")
```

```
    else:
        # Respond to the user's input with appropriate response
        if chatbot_greetings(input_from_user) is None:
            # Response if the user wants a fun fact
            if 'fun fact' in input_from_user:
                random_fact = get_fun_fact()
                print("Zero: " + random_fact)
            # Response if user tells chatbot their name
            if 'my name is' in input_from_user:
                with open('user_models.txt', 'a') as file:
                    file.write(input_from_user + '\n')
                file.close()
                print("Zero: " + random.choice(chatbot_name_responses))
            # Response if the user tells chatbot their likes
            if any(substring in input_from_user for substring in user_likes):
                with open('user_models.txt', 'a') as file:
                    file.write(input_from_user + '\n')
                file.close()
                print("Zero: " + random.choice(chatbot_like_responses))
            # Response if the user tells chatbot their dislikes
            if any(substring in input_from_user for substring in user_dislikes):
                with open('user_models.txt', 'a') as file:
                    file.write(input_from_user + '\n')
                file.close()
                print("Zero: " + random.choice(chatbot_dislike_responses))
            # General response to user
            if all(substring not in input_from_user for substring in user_likes)\
            and all(substring not in input_from_user for substring in user_dislikes)\
            and ('fun fact' and 'my name is' not in input_from_user):
                print("Zero: " + chatbot_response(input_from_user))
                sentence_tokens.remove(input_from_user)
        # Response if the user greets the chatbot
        elif chatbot_greetings(input_from_user) is not None:
            print("Zero: " + chatbot_greetings(input_from_user))
# Response if the user says goodbye to the chatbot
else:
    chatbot_running = False

    # Put marker to indicate a new user model
    with open('user_models.txt', 'a') as file:
        file.write("----------------------------------------------------------------" + '\n')
    file.close()

    print("Thanks for talking to me.")
```