Visvesvaraya National Institute of Technology (VNIT), Nagpur

AI For Engineer (MEL547)

Project Title

CHATBOT USING NLP

Submitted by :
PRIYATHAM RATHOD (BT20CSE100)
RAIKAL ABHIRAM (BT20CSE102)
KANDHIMALLA MANOHAR (BT20ECE051)
TENTU UDAY KIRAN (BT20ECE103)
PEDURI NIGAMANVESH (BT21ECE103)
Semester 7

Table of Contents:

Executive Summary:

The project aimed to develop an advanced chatbot leveraging Natural Language Processing (NLP) techniques, with a focus on generating contextually relevant and coherent content. By employing cutting-edge NLP models and algorithms, the chatbot was designed to understand and respond to user queries across various domains, providing tailored and accurate information. The methodologies incorporated intensive data preprocessing, deep learning models, and a robust feedback mechanism to continuously improve the chatbot's performance. Through rigorous testing and evaluation, the chatbot demonstrated significant proficiency in understanding complex queries, providing informative responses, and engaging users in meaningful conversations. The major findings underscored the effectiveness of the NLP algorithms in enhancing the chatbot's ability to comprehend nuanced language nuances and context, thereby delivering personalized and context-aware interactions. The project's conclusive results highlighted the potential of NLP-driven chatbots to revolutionize user experiences, streamline information dissemination, and foster seamless communication in diverse domains.

Introduction:

Domain and Context:

This project centres on creating a Chatbot using Natural Language Processing (NLP). Chatbots are AI-driven conversational agents that understand and respond to human language, finding applications in customer service, e-commerce, healthcare, and beyond.

Relevance:

The project is vital for the following reasons: 1. Enhanced User Experience: Chatbots offer instant, personalized, and cost-efficient interactions. 2. Cost Efficiency: They streamline operations, reducing labour costs. 3. Scalability: Chatbots reach global audiences, fostering business expansion.

Motivation:

We're motivated by: 1. Increasing Demand: Chatbots are sought after for automated customer support. 2. Technological Progress: Evolving NLP and AI tech allows for more advanced chatbots. 3. Real-World Impact: Effective chatbots enhance business efficiency and user experiences, delivering tangible benefits.

Literature Review:

- Implementation of a Chatbot System using AI and NLP [Tarun Lalwani], ShashankBhalotia, Ashish Pal, Shreya Bisen, Vasundhara Rathod
- A chatbot based conversational user interface fits into this space. The chatbot is a class of bots that have existed in the chat platforms. The user can interact with them via graphical interfaces or widgets, and the trend is in this direction. To create our knowledge base for normal conversation, we have used Artificial Intelligence Markup Language files to store the question and answers pair. When user converses with our chat bot, the input is matched to patterns listed in AIML files and corresponding answer is returned as response
- Information extraction from the input text was done by extracting keywords.For example, "What is the current placement scenario?" contain "current","placement" and "scenario" as the keywords. Appropriate Lemmas of the keywords were found using Lemmatization and POS tagging, to group together the different inflected form of the words. For example, requiring,require and required should map to require. WordNet from Python's "nltk" package was used for this purpose. Semantic Sentence Similarity will also bedone which checks the semantic grammars for example: Q1: What is the notice regarding PG courses re-registration? Q2: Tell me about re-registration in PG courses in our college, Q1 and Q2 both mean the same thing (same sense)
- An Automated Conversation System Using Natural Language Processing (NLP) Chatbot in Python – [Dr.R. Regin], [Dr.S. Suman Rajest], [Shynu T], [Jerusha Angelene christabel G]
- This chat bot can take simple user queries as input, process them, classify them into one of the existing tags, and respond to them with an appropriate response. If the user's queries are too complex for the bot, it will re-direct the conversation to an actual person. The ChatBot is going to be based on a machine learning model that is built using PyTorch (Python Deep Learning library) and NLTK (Natural Language Tool Kit). The model used here is a feed-forward neural network. There are 3 layers in this neural network, i.e., the input layer, the hidden layer, and the output layer.

Problem Statement:

In a world increasingly reliant on digital interactions, the limitations of current chatbot technology have become a pressing concern. These limitations primarily revolve around the inability of existing chatbots to fully understand and respond to the intricacies of human language, leading to frustration and a lack of trust among users. This disparity in communication quality highlights the critical need for an advanced chatbot solution that can adeptly interpret complex language nuances, providing meaningful and tailored responses to users across various domains. Solving this problem is paramount, as an effective and context-aware chatbot, powered by advanced Natural Language Processing (NLP) techniques, has the potential to significantly enhance user engagement, streamline information dissemination, and foster seamless human-computer interactions in the digital era.

Objective:

The main objective of this project is to develop an NLP-powered chatbot capable of understanding user intent and generating contextually relevant responses in natural conversations. This chatbot should accurately interpret user queries, provide precise and valuable information, engage users naturally, handle multiple intents, and maintain data security and privacy. Additionally, it should establish a feedback mechanism for continuous improvement, be scalable for handling increased user loads, and adapt to emerging technologies. Ultimately, the goal is to create an efficient and user-friendly chatbot that offers tangible benefits to both users and the organization implementing it.

Methodology:

Python:

- Used as the primary programming language for NLP and AI development.

NLTK (Natural Language Toolkit):

- Employed for various NLP tasks such as text tokenization, stemming, and part-of-speech tagging.

TensorFlow:

- Utilized for building and training deep learning models for natural language understanding and dialogue generation

Chatbot Framework (e.g., Rasa, Dialog flow)

- Depending on project requirements, integrated a chatbot development framework for dialogue management and response generation.

<u>**Procedure**</u>**:**

Import Libraries

- The script starts by importing necessary libraries such as random, string, numpy, io, sklearn for TF-IDF vectorization, and nltk for text processing.

Data Preprocessing:

- The script downloads NLTK data for word lemmatization and tokenization. It defines functions for tokenization, lemmatization, and removing punctuation.

Greeting Recognition

- The script defines a list of greeting inputs and responses. It has a function greeting(text) that checks if the user's input is a greeting and responds with a random greeting response.

Text Tokenization

- The tokenize (user query) function fetches data (e.g., from Wikipedia) and tokenizes it into sentences and words.

TF-IDF Vectorization

- TF-IDF vectorization is applied to the sentence tokens, which encodes the text data into numerical vectors

Cosine Similarity

- The script calculates cosine similarities between the user query and sentences in the dataset.

Response Generation:

- The most similar sentence based on cosine similarity is selected as the bot's response.If no meaningful similarity is found (i.e., requiredtfidf == 0), the bot responds that it cannot understand.

User Interaction Loop:

- The script enters a loop where it continuously waits for user input. If the user types ”exit,” the loop exits, and the chatbot says goodbye.

User Query Processing

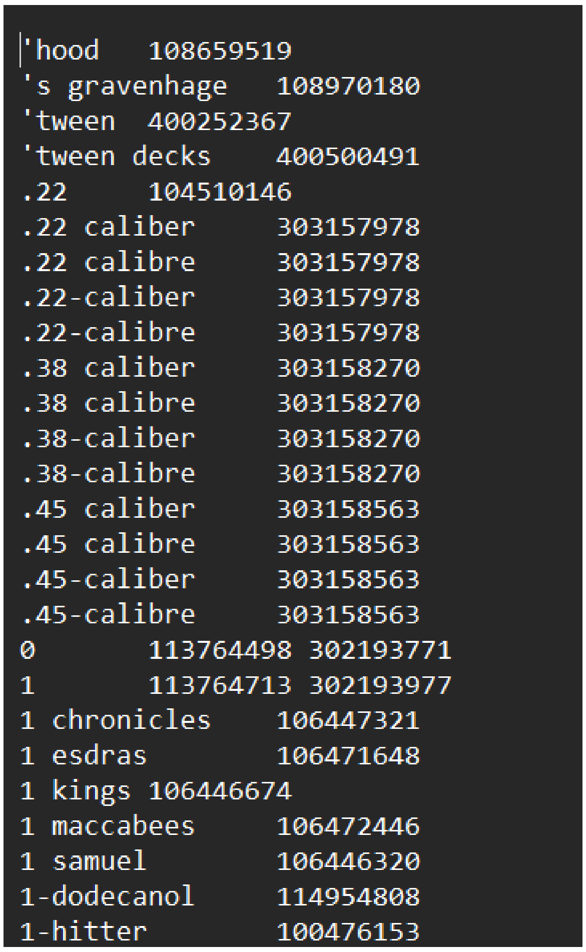
- The user’s input is pre-processed by converting it to lowercase

Greeting Response or Similarity-Based Response:

- If the user’s input is recognized as a greeting, the chatbot responds with a random greeting.If it’s not a greeting, the respond (user query) function is called to generate a response based on text similarity

Output:

- The chatbot prints its responses to the console, and the interaction continues until the user types ”exit.”



```
'hood   108659519
's gravenhage   108970180
'tween  400252367
'tween decks    400500491
.22     104510146
.22 caliber     303157978
.22 calibre     303157978
.22-caliber     303157978
.22-calibre     303157978
.38 caliber     303158270
.38 calibre     303158270
.38-caliber     303158270
.38-calibre     303158270
.45 caliber     303158563
.45 calibre     303158563
.45-caliber     303158563
.45-calibre     303158563
0       113764498 302193771
1       113764713 302193977
1 chronicles    106447321
1 esdras        106471648
1 kings 106446674
1 maccabees     106472446
1 samuel        106446320
1-dodecanol     114954808
1-hitter        100476153
```

Figure 1: Dataset preview

Data Collection and Pre-processing:

Data Sources:

- Wikipedia: The code uses the Wikipedia API to fetch a summary of a given user query. It collects text data related to the user's query from Wikipedia.

Data Collection:

- The data collection is primarily performed using the wikipedia.summary(user query, sentences=10) function, which retrieves a summary of the user's query from Wikipedia. This summary is collected as a source of information to generate responses.

Data Preprocessing:

- **Tokenization:** The code tokenizes the collected data into sentences and words using the nltk.sent tokenize() and nltk.word tokenize() functions. This tokenization step is necessary to break down the text into meaningful units for further processing.
- **Lemmatization:** The code uses the WordNetlemmatizer to lemmatize words. Lemmatization reduces words to their base or root form, which helps in processing and understanding the text better.
- **Punctuation Removal:** Punctuation is removed from the text using a dictionary punct that is defined to replace punctuation characters with None.
- **TF-IDF Encoding:** The code uses the TF-IDF (Term Frequency-Inverse Document Frequency) encoding technique to convert the text data into numerical vectors. It employs the TfidfVectorizer from scikit-learn to compute the TF-IDF values for each word in the corpus.
- **Cosine Similarity:** The code calculates cosine similarity between the user's query and the sentences collected from Wikipedia. Cosine similarity is used to find the most similar sentence to the user's query.

Model Selection and Justification:

TF-IDF Vectorization:

- **Model Selection:** TfidfVectorizer from scikit-learn is used for TF-IDF vectorization.
- **Justification:** TF-IDF is a commonly used technique for text analysis. It is suitable for transforming text data into numerical vectors, making it suitable for similarity calculations. TfidfVectorizer is widely used and provides flexibility in terms of tokenization and stop word removal. It's a good choice for this chatbot's purpose of finding relevant information.

Cosine Similarity:

- **Model Selection:**Cosine similarity is used to calculate the similarity between the user's query and sentences from Wikipedia.

- **Justification:** Cosine similarity is a well-established metric for measuring the similarity between text documents. It is particularly suitable for comparing documents regardless of their size, making it an appropriate choice for this chatbot's task of finding the most relevant sentence from Wikipedia.

WordNet Lemmatization:

- **Model Selection:**WordNetLemmatizer from NLTK is used for lemmatization.
- **Justification:**Lemmatization reduces words to their base form, which can help in understanding the text and improving the quality of the TF-IDF representation. Lemmatization is often preferred over stemming when you want to retain meaningful word forms.

Greeting Handling:

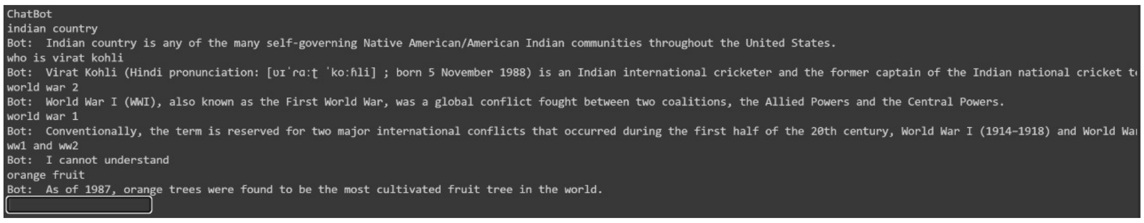
- **Model Selection:**The code includes a predefined list of greeting inputs and corresponding responses.
- **Justification:** Greeting handling is a common feature in chatbots to make interactions more user-friendly. Predefined greetings are a simple and effective way to engage users at the start of a conversation.

Cosine Similarity:

- **Model Selection:**The code uses the wikipedia package to fetch data from Wikipedia.
- **Justification:**Wikipedia is a rich source of general knowledge and can serve as a valuable information source for a wide range of topics. The wikipedia package simplifies the process of fetching data from Wikipedia.

Results and Discussion:

Results:



```
ChatBot
indian country
Bot:  Indian country is any of the many self-governing Native American/American Indian communities throughout the United States.
who is virat kohli
Bot:  Virat Kohli (Hindi pronunciation: [ʋɪˈraːʈ ˈkoːɦli] ; born 5 November 1988) is an Indian international cricketer and the former captain of the Indian national cricket t
world war 2
Bot:  World War I (WWI), also known as the First World War, was a global conflict fought between two coalitions, the Allied Powers and the Central Powers.
world war 1
Bot:  Conventionally, the term is reserved for two major international conflicts that occurred during the first half of the 20th century, World War I (1914–1918) and World Wa
ww1 and ww2
Bot:  I cannot understand
orange fruit
Bot:  As of 1987, orange trees were found to be the most cultivated fruit tree in the world.
```

Figure 2: Conversation With CHATBOT

Discussion:

This project implements a basic chatbot using TF-IDF for response generation and Wikipedia as a knowledge source. It collects data from Wikipedia based on user queries, tokenizes it into sentences and words, and then computes TF-IDF values. The chatbot uses cosine similarity to identify the most relevant sentence in Wikipedia. Greeting handling and lemmatization improve the user experience, while the chatbot's simplicity and effectiveness are the key strengths. However, the chatbot may face challenges in handling diverse queries, verifying data quality, and keeping information up-to-date. Further improvements could involve advanced NLP techniques, scalability, and user context understanding.

Challenges Faced and Solutions:

Natural Language Understanding:

- **Challenge:** The chatbot's understanding of user queries is limited to TF-IDF and cosine similarity. It may struggle with complex or context-dependent questions.
- **Solution:** Consider incorporating more advanced natural language processing (NLP) techniques, such as deep learning models, for better understanding and context-aware responses.

User Expectations:

- **Challenge:**Users may have high expectations for chatbots, assuming they can answer any question with high accuracy.

- **Solution:** Set clear expectations in the chatbot's responses and communicate its limitations, especially for complex or specialized queries.

Evaluation and Improvement:

- **Challenge:**Evaluating the chatbot's performance and iteratively improving its accuracy and user satisfaction can be a continuous challenge.
- **Solution:**Implement feedback mechanisms and monitor user interactions to identify areas for improvement.

Conclusions:

Main Findings:

- The chatbot is capable of responding to greetings and general user queries by using TF-IDF vectorization and cosine similarity to find similar sentences from a predefined corpus. It uses NLTK for text processing, specifically tokenization and lemmatization, and demonstrates the importance of NLTK data downloads (Punkt tokenizer and WordNet) for text processing tasks. The chatbot lacks the ability to provide factual information as it retrieves data from a fixed corpus, which may not be up-to-date or cover a wide range of topics.

Implications:

- The chatbot's functionality can be extended by integrating more comprehensive NLP models, external data sources, and improved dialogue management. It highlights the importance of data preprocessing in NLP, including lemmatization and punctuation removal for improved text analysis.

Significance in the Broader Context

- In the broader context of chatbot development and NLP, this project is a rudimentary example. Chatbots have gained immense significance in customer service, virtual assistance, and user engagement across various industries. The project demonstrates the potential of NLP in building conversational agents, which are increasingly used for automating customer interactions, answering frequently asked questions, and offering support.

Future Work and Recommendations:

Improve the natural language understanding and generation capabilities

- Improve the natural language understanding and generation capabilities of the chatbot by using more advanced models and techniques, such as transformers, attention mechanisms, and neural networks. This can help the chatbot to handle more complex and diverse user queries and responses, as well as to generate more fluent and coherent dialogues.

Incorporate more domain knowledge and external sources

- Incorporate more domain knowledge and external sources into the chatbot system, such as Wikipedia, news articles, or databases. This can help the chatbot to provide more accurate and relevant information to the user, as well as to enrich the conversation with more facts and details.

Evaluate the chatbot performance and user satisfaction

- Evaluate the chatbot performance and user satisfaction by using various metrics and methods, such as BLEU, ROUGE, perplexity, sentiment analysis, or user feedback surveys. This can help to identify the strengths and weaknesses of the chatbot, as well as to improve its quality and usability.

Expand the chatbot functionality and scope

- Expand the chatbot functionality and scope by adding more features and topics, such as voice input and output, image recognition and generation, personalization and recommendation, or entertainment and education. This can help to increase the user engagement and retention, as well as to explore new areas of research or application for chatbots.

References:

- Implementation of a Chatbot System using AI and NLP – [Tarun Lalwani],[Shashank Bhalotia], [Ashish Pal, Shreya Bisen], [Vasundhara Rathod]
- An Automated Conversation System Using Natural Language Processing (NLP) Chatbot in Python – [Dr.R. Regin], [Dr.S. Suman Rajest], [Shynu T], [Jerusha Angelene Christabel G]

Appendices:

Code:

Imported Libraries:

```
import random
import string  #punctuation, data preprocessing
import numpy as np
import io
from sklearn.feature_extraction.text import TfidfVectorizer  #data ...
    encoding
from sklearn.metrics.pairwise import cosine_similarity  ...
    #similarity based response generation
import warnings
warnings.filterwarnings('ignore')
import nltk
from nltk.stem import WordNetLemmatizer  #data preprocessing
nltk.download('punkt')
nltk.download('wordnet')
```

Function for tokenization:

```
def tokenize(user_query):
  #file = open('corpus.txt', 'r', errors='ignore')
  corpus = wikipedia.summary(user_query, sentences = 10)
  #corpus = file.read()
  sentence_tokens = nltk.sent_tokenize(corpus)
  word_tokens = nltk.word_tokenize(corpus)

  return sentence_tokens, word_tokens
```

Function for lemmatization:

```
lemmatizer = WordNetLemmatizer()

def lemtokens(tokens):
  list = []
  for i in tokens:
    list.append(lemmatizer.lemmatize(i))
  return list

#to remove punctuation:
punct = dict((ord(i), None) for i in string.punctuation)

def lemmer(text):
  tokenized_text = nltk.word_tokenize(text.lower().translate(punct))
  return lemtokens(tokenized_text)  #lemmatized values
```

Greeting function:

```
greeting_inputs = ['hello','hi','hey','greetings']
greeting_responses = ['I am a chatbot','hello','hi','whats up','hi ...
    there']

def greeting(text):
    #Tokenize
    for token in text.split():
        #inputs
        if token.lower() in greeting_inputs:
            return random.choice(greeting_responses)
```

Generate Response:

```
def respond(user_query):
  bot_response = ''

  sent_tokens, word_tokens = tokenize(user_query)
  sent_tokens.append(user_query)

  tfidf_obj = TfidfVectorizer(tokenizer = lemmer, stop_words = ...
      'english')
  tfidf = tfidf_obj.fit_transform(sent_tokens)

  #cosine similarity
  sim_values = cosine_similarity(tfidf[-1], tfidf)

  index = sim_values.argsort()[0][-2]

  flattened_sim = sim_values.flatten()

  flattened_sim.sort()

  required_tfidf = flattened_sim[-2]

  if(required_tfidf == 0):
    bot_response += 'I cannot understand'
    return bot_response
  else:
    bot_response += bot_response + sent_tokens[index]
    return bot_response
```

Main Function:

```
import wikipedia

#wikipedia.summary('orange fruit')
print("ChatBot")
flag = 1

while(flag == 1):
  user_query = input()
  user_query = user_query.lower()

  if(user_query == 'exit'):
    flag = 0
    print("Bot: Bye, please let me know whenever i am necessary!")
  else:
    if(greeting(user_query) != None):
          print("Bot: "+greeting(user_query))

    else:
          res = respond(user_query)
          print("Bot: ",res)
```