CHATBOT DEVELOPMENT

A Course project report submitted
in partial fulfilment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

MACHARLA ROHITH	**(2203A52157)**
CH VENKATA SHIVA SRI	**(2203A52145)**
JILLA KIRTHAN	**(2203A52151)**

Under the guidance of

Dr. E.L.N. Kiran Kumar

Associate Professor, CS & AI

SR UNIVERSITY

SR UNIVERSITY

Ananthasagar, Warangal.

CERTIFICATE

This is to certify that this project entitled **“CHATBOT DEVELOPMENT**" is the bonafied work carried out by **ROHITH MACHARLA, CH VENKATA SHIVA SRI, JILLA KIRTHAN** as a Course project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING** during the academic year 2023-2024 under our guidance and Supervision.

Dr.E.L.N. Kiran Kumar
Assoc. Prof. CS & AI
SR University,
Ananthasagar, Warangal.

Dr.M.Sheshikala
Assoc. Prof. & HOD(CSE),
SR University,
Ananthasagar, Warangal.

External Examiner

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our project guide **Dr. E.L.N. KiranKumar, Assoc. Prof. CS and AI** as well as Head of the CSE Department **Dr. M. Sheshikala, Associate Professor** for guiding us from the beginning through the end of the Course project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to project coordinators **Mr. Sallauddin Md, Asst. Prof, Y. Chanti Asst. Prof.** for their encouragement and support.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

CONTENTS

LIST OF ACRONYMS

- LSTM: Long Short-Term Memory
- RNN: Recurrent Neural Network
- Sentence BERT: Bidirectional Encoder Representations from Transformers
- NLP: Natural Language Processing
- KNN: K-Nearest Neighbour
- SVM: Support Vector Machine
- NLTK: Natural Language Toolkit
- TF-IDF: Term Frequency-Inverse Document Frequency

1. ABSTRACT

A basic chatbot capable of engaging in university-related conversations is implemented using the Python code provided. The chatbot preprocesses user input, classifies intents with a logistic regression classifier trained on TF-IDF vectorized text data, and generates appropriate responses based on the predicted intent. It does this by utilizing libraries like scikit-learn and NLTK. The bot's capabilities can be easily customized and expanded thanks to intents and responses that are stored in a JSON file. Additionally, the basic interaction handling provided by the integration of the NLTK Chat module enhances the conversational experience. With potential improvements like high level NLP strategies, logical grasping, feeling investigation joining, Programming interface combination, and UI upgrades, the chatbot can develop into a more wise and easy to use conversational specialist fit for giving customized and relevantly significant reactions to client questions.

KEYWORDS

Cosine Similarity, NLTK, TFIDF-Vectorizer, Tokenization, Lemmatization.

2. INTRODUCTION:

We discuss the intricate processes involved in developing an intelligent conversational agent. In the digital age of today, chatbots have become essential parts of a variety of applications, making it easier for users to communicate with one another in a natural way and offering immediate assistance. The goal of our project is to create a sophisticated chatbot that can comprehend user queries and provide responses that are contextually relevant by combining the capabilities of machine learning and natural language processing (NLP).

Key Parts and Advancements :

At the center of our chatbot lies aim characterization, a cycle where client inputs are classified into predefined aim classifications. Utilizing methods like TF-IDF vectorization and calculated relapse, our chatbot precisely recognizes the hidden aim behind each inquiry. Besides, we utilize the NLTK library for tokenization and lemmatization, empowering proficient text handling. Additionally, the robustness and accuracy of our chatbot's classification capabilities are guaranteed by the fact that model training and evaluation are made easier with scikit-learn.

The capacity of our chatbot to manage context, which improves the flow of the conversation, is one of its distinguishing characteristics. The chatbot intelligently adjusts its responses based on contextual information from previous interactions, resulting in a personalized and coherent user experience. This context oriented mindfulness empowers the chatbot to grasp subtleties and keep up with progression across discussions, encouraging a seriously captivating client experience.

2.1. Overview

Outline All in all, our chatbot improvement project addresses a combination of cutting edge innovations and phonetic standards to make a clever conversational specialist. Through fastidious plan and execution, we have made a chatbot fit for deciphering client inquiries, knowing expectations, and conveying logically suitable reactions. As chatbots proceed to advance and multiply across different spaces, our venture represents the groundbreaking capability of artificial intelligence and NLP in upsetting human-machine connection.

3. PROBLEM STATEMENT:

How can SR College streamline the process of addressing numerous admission inquiries from prospective students efficiently and accurately using a chatbot framework.

4. MOTIVATION AND SCOPE OF WORK:

The advancement of the SR College chatbot for affirmations requests is inspired by the squeezing need to smooth out and upgrade the confirmations interaction for imminent understudies. With the rising digitalization of instruction, there is a developing assumption for colleges to give open and proficient channels to data dispersal and question goal. By utilizing normal language handling (NLP) methods and opinion examination, the chatbot expects to engage imminent understudies with an easy to understand connection point to address their confirmations related questions quickly and precisely. The undertaking's degree envelops the formation of an exhaustive information base that covers different parts of the confirmations cycle, including qualification measures, application methodology, documentation prerequisites, expense structures, grant open doors, and significant cutoff times. Additionally, the chatbot will be constructed to provide individualized support throughout the admissions process by responding to user inquiries. This drive reflects SR College's obligation to utilizing innovation to upgrade the general understudy insight and work with consistent commitment with imminent understudies during the affirmations cycle.

5. LITERATIVE REVIEW

5.1 Related Work

Related Tasks In the domain of opinion examination, various exploration tries have carefully investigated the viability of different AI classifiers, planning to perceive their utility and execution across a different range of datasets. Existing research has examined the use of well-known classifiers like Naive Bayes, J48, OneR, and BFTree within this landscape. These classifiers have gone through thorough assessment across various datasets enveloping fluctuated sources, including virtual entertainment text, microblogging sites, and news stories. Particularly noteworthy is the remarkable accuracy of analyses of social media text, with reported figures reaching as high as 91.3%.

Additionally, specialists have set out on exhaustive examinations to disclose the meaning of unmistakable AI procedures in improving feeling investigation processes. Strategies going from Greatest Entropy and Arbitrary Backwoods to Stochastic Slope Drop (SGD), Multi-facet Perceptron (MLP), and Backing Vector Machine (SVM) have gone through exhaustive examination. Every strategy displays nuanced qualities and shortcomings, with Irregular Woods, for example, exhibiting vital correctnesses of up to 88.39%, and SVM exhibiting vigorous execution with exactnesses coming to 81.15%. Besides, studies have embraced a sweeping investigation of feeling examination strategies across fluctuated settings and areas.

Examinations led on datasets crossing film audits and Coronavirus related tweets have given priceless experiences into the viability of classifiers like Credulous Bayes and K-Closest Neighbor. Especially critical is the unrivaled precision of Gullible Bayes, which has been accounted for at 81.45% in specific examinations, highlighting its viability in knowing feeling subtleties inside printed information. Imaginative methodologies, for example, crossover subject based opinion examination (HTBSA), have arisen as an original worldview in feeling examination research. These methodologies coordinate high level procedures, for example, Biterm theme demonstrating to remove feeling rich data from brief tweets, offering significant experiences into popular assessment elements, especially in areas like political decision expectation.

Additionally, sentiment analysis techniques have been rigorously applied to a variety of datasets, including Amazon product reviews. Investigations directed on such datasets have exhibited the adequacy of refined models like Bi-LSTM with consideration instrument, accomplishing striking exactnesses of up to 95.1%. In the blossoming field of opinion examination, a plenty of exploration tries have fastidiously investigated the viability of different AI classifiers, expecting to recognize their utility and execution across a different range of datasets.

The interpretability and robustness of sentiment analysis methods have also been examined in these studies in addition to the accuracy of classifiers. Existing studies in this area have looked at how to use well-known classifiers like Naive Bayes, J48, OneR, and BFTree, as well as more recent ones like Maximum Entropy, Random Forest, and Multi-facet Perceptron (MLP). These classifiers have gone through thorough assessment across different datasets incorporating fluctuated sources, including online entertainment text, microblogging sites, news stories, and even item feedback from online retailers like Amazon. Quite, examinations directed via web-based entertainment text have revealed surprising correctnesses, with detailed figures coming to as high as 91.3%.

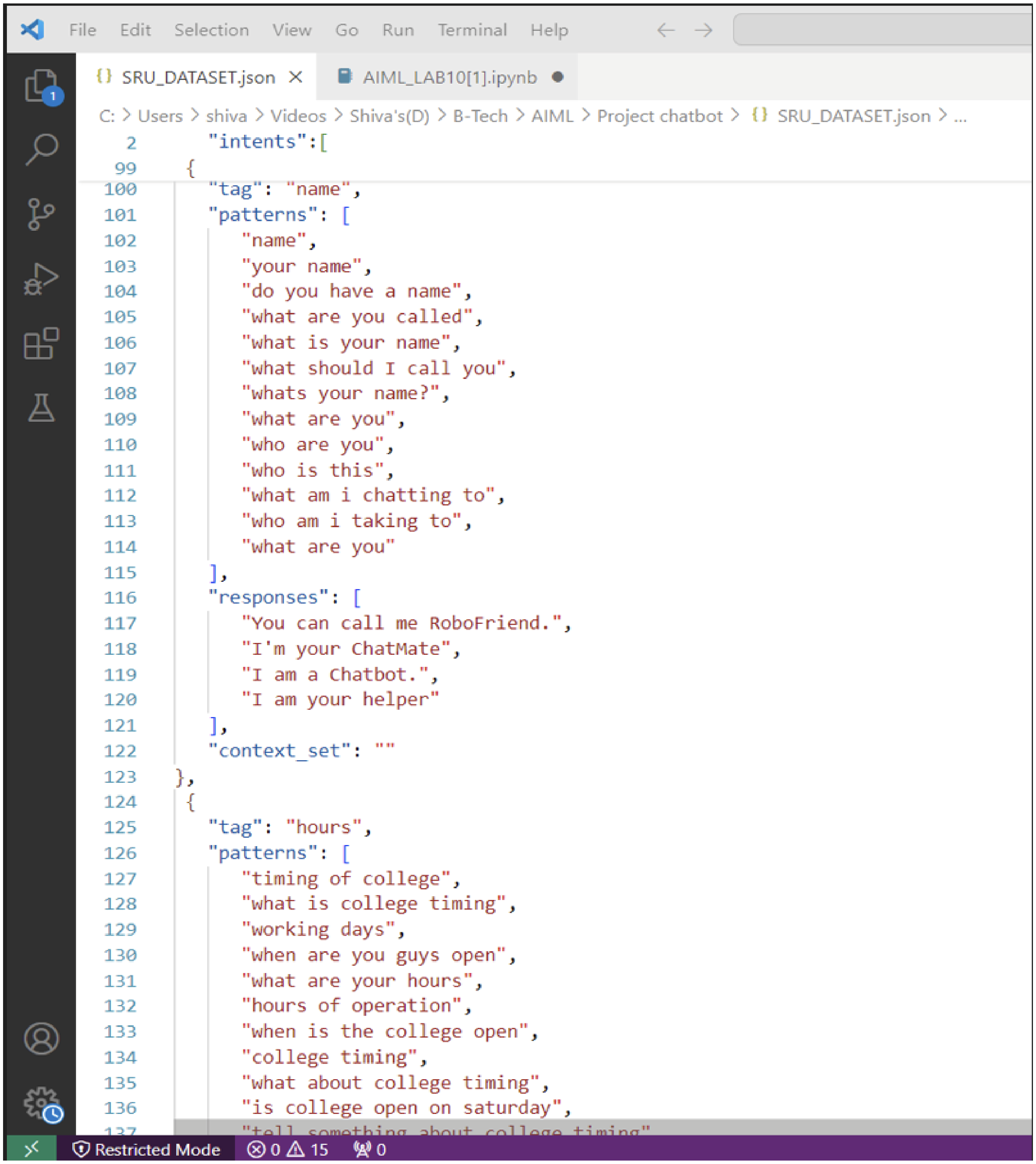
In addition, extensive research has been conducted to determine the significance of various machine learning methods in enhancing sentiment analysis procedures. Strategies going from Stochastic Slope Plummet (SGD) to Help Vector Machine (SVM) have gone through careful examination. Each method has distinct advantages and disadvantages, with some algorithms excelling in precision and others in recall rates. These examinations give an exhaustive comprehension of the compromises engaged with choosing the suitable classifier for feeling investigation undertakings in light of explicit necessities and dataset qualities. Moreover, creative methodologies, for example, half breed theme based opinion examination (HTBSA), have arisen as an original worldview in feeling investigation research.

These methodologies coordinate high level procedures, for example, Biterm point displaying to remove feeling rich data from concise tweets, offering significant experiences into general assessment elements, especially in areas like political decision expectation. Sentiment analysis accuracy has also significantly improved as a result of advancements in neural network architectures, such as Bi-LSTM with attention mechanisms, especially when working with large datasets.

In conclusion, these studies emphasize how crucial sentiment analysis techniques are for elucidating and interpreting textual data in a wide range of fields. By saddling the force of AI calculations and high level procedures, scientists keep on disentangling significant experiences into public feeling and assessment elements, in this manner contributing altogether to the more extensive scene of opinion examination research and its down to earth applications in genuine situations.

6. DATASET:

Dataset consists of data which is useful for users in admission enquiry as per their queries.


File Edit Selection View Go Run Terminal Help
SRU_DATASET.json
AIML_LAB10[1].ipynb
C: > Users > shiva > Videos > Shiva's(D) > B-Tech > AIML > Project chatbot > SRU_DATASET.json > ...
Restricted Mode
0 15 0

```
    "intents":[
  {
    "tag": "name",
    "patterns": [
      "name",
      "your name",
      "do you have a name",
      "what are you called",
      "what is your name",
      "what should I call you",
      "whats your name?",
      "what are you",
      "who are you",
      "who is this",
      "what am i chatting to",
      "who am i taking to",
      "what are you"
    ],
    "responses": [
      "You can call me RoboFriend.",
      "I'm your ChatMate",
      "I am a Chatbot.",
      "I am your helper"
    ],
    "context_set": ""
},
  {
    "tag": "hours",
    "patterns": [
      "timing of college",
      "what is college timing",
      "working days",
      "when are you guys open",
      "what are your hours",
      "hours of operation",
      "when is the college open",
      "college timing",
      "what about college timing",
      "is college open on saturday",
```

Fig 1.Dataset

7. PROPOSED METHODOLOGY:

7.1 Methodology

Code successfully demonstrates how artificial intelligence (AI) makes use of natural language processing (NLP) methods to comprehend and respond to human language inputs. It preprocesses text data for classification tasks using tokenization, lemmatization, and TF-IDF vectorization. The execution of a calculated relapse classifier further grandstands the use of conventional AI calculations in plan grouping. Even though your code doesn't directly use deep learning concepts like recurrent neural networks (RNNs) or long short-term memory (LSTM) networks, it uses NLP effectively to make it easier for users and the AI system to talk and interact. These strategies act as the establishment for empowering AI frameworks to appreciate and create significant reactions to human language, preparing for more complex language understanding and handling capacities in computer based intelligence driven applications.

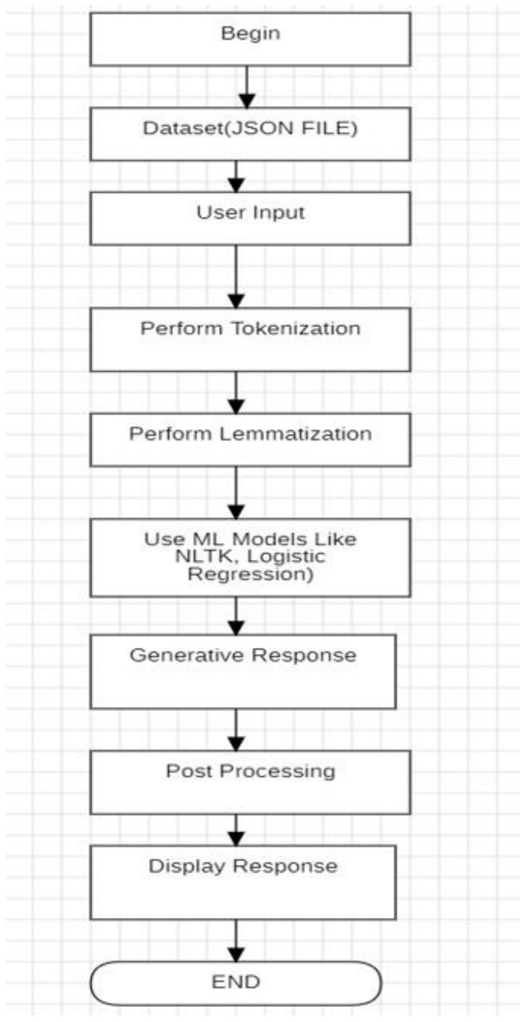


Fig 2. Flow chart

The chatbot's functionality is laid out in detail as shown in the above Fig 2: flowchart, which shows how inputs from users are processed and responses are generated. It starts with the client input stage, where the chatbot anticipates a message or question from the client. To ensure uniformity and consistency in handling textual data, this input is subjected to text preprocessing, which includes tasks like tokenization and lemmatization.

The chatbot moves on to intent classification after preprocessing, a crucial step in which user input is divided into predefined intents or categories. This task typically calls for the use of machine learning models like Logistic Regression or Naive Bayes. These models enable the chatbot to precisely determine the user's intention or question. Setting the executives, a discretionary step, permits the chatbot to keep up with discussion setting by putting away and referring to data from past communications.

When the goal is grouped, the chatbot recovers a fitting reaction from a data set or predefined set of reaction layouts. This reaction age stage is significant in guaranteeing that the chatbot conveys important and relevantly proper reactions to the client's question. Moreover, post-handling steps, for example, feeling examination or substance extraction might be applied to additionally refine the reaction before it is shown to the client.

The chatbot's interaction flow is organized in a way that the flowchart shows, guiding the development process and making sure that all necessary parts, from processing user input to producing responses, are included. By following this organized methodology, engineers can plan and execute a vigorous and proficient chatbot equipped for taking part in significant discussions with clients.

7.2 Data Flow Diagram (DFD)

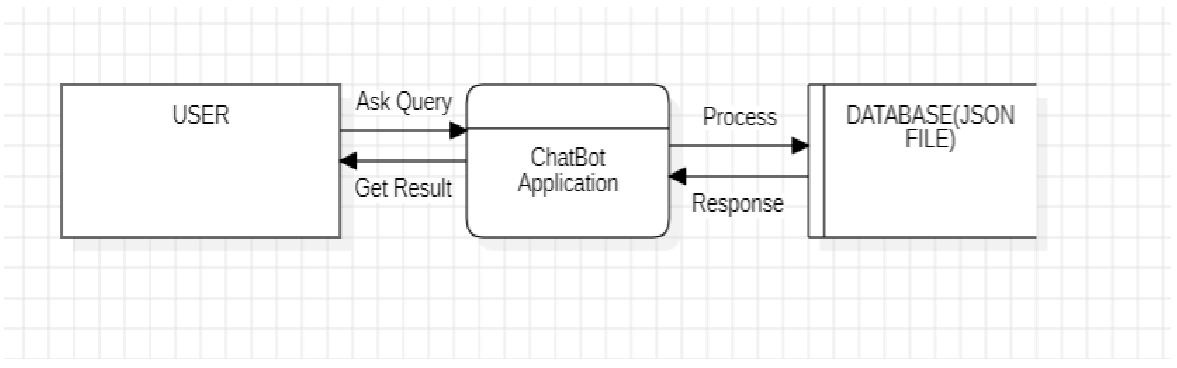


Fig. 3 Data Flow Diagram

Your chatbot development project's dataflow diagram depicts the system's data flow, showing how information is transmitted and processed during various interactions. It depicts the excursion of client input as it crosses through preprocessing, purpose order, reaction age, and eventually, the showcase of the user's response from the chatbot. This graph fills in as a visual portrayal of the chatbot's hidden design, featuring the interconnectedness of its parts and the consistent trade of information between them. By depicting the information stream inside the framework, the dataflow outline gives significant bits of knowledge into the chatbot's activity, working with cognizance and investigation of its usefulness.

Furthermore, the diagram serves as documentation for the chatbot project, enabling future developers to comprehend the system's design and make modifications or updates with ease. It encapsulates the essence of the chatbot's functionality in a concise and comprehensible format, ensuring the preservation of institutional knowledge over time.

7.3 DATA PRE-PROCESSING

We conduct a preliminary data analysis prior to employing any method for emotion or feeling analysis. Information handling is a significant stage in report extraction that produces top notch text discontinuity and limits computational intricacy. Tweet data processing steps are outlined in the following_points.

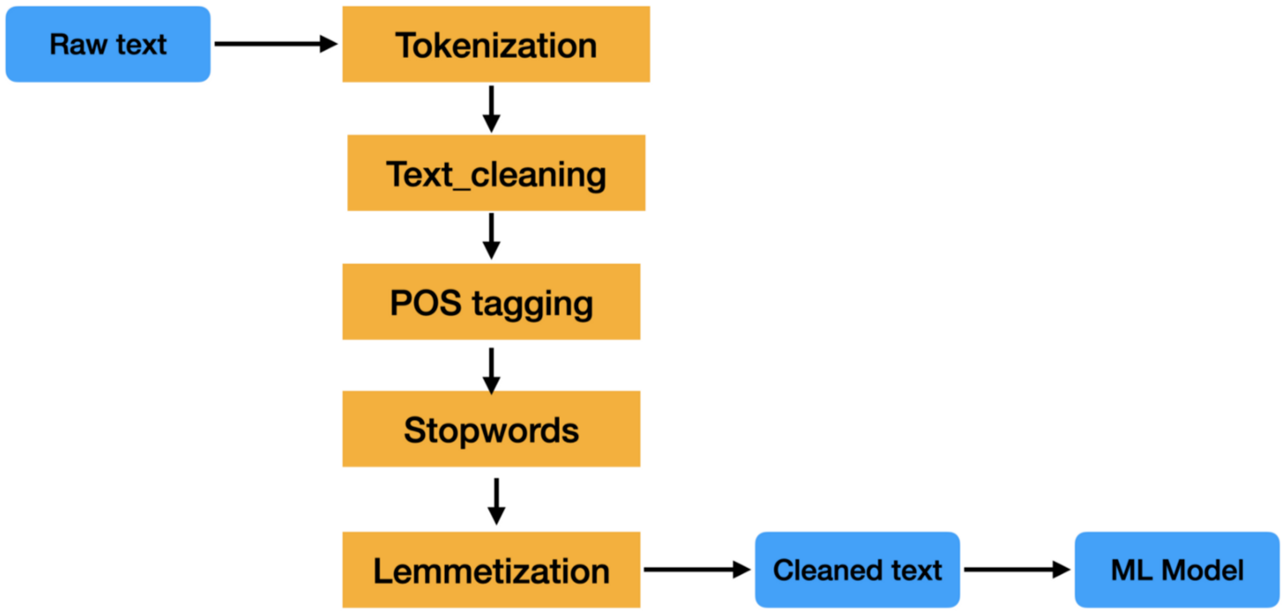


Fig. 4 Data Pre-Processing

7.3.1. Punctuation marks Removal: In this process, the special characters such as “’?!;: # $ % & ()
* + − / < > = [] n ˆ _ {} |~ are removed from the tweets.

7.3.2. Stop Words Removal: stop words such as I, and, the, for, should and etc., are removed from the tweets using NLTK’s (Natural Language Toolkit) stop word list.

7.3.3. Tokenization: In this process tweets are splitted into phrases or tokens, symbols and words.

7.3.4. Lemmatization: The main work of lemmatizing process is to reduce the words to its base forms with the help of morphology lemmatization. For instance, the words ‘advising’ and ‘advised’ are reduced to their root word as ‘advice’.

7.4 COMPARED ALGORITHMS

7.4.1. NLTK (Natural Language Toolkit):

NLTK, or Regular Language Toolbox, is a thorough Python library broadly utilized for normal language handling undertakings. It offers functionalities for tokenization, stemming, lemmatization, grammatical feature labeling, and that's just the beginning. NLTK gives instruments and assets to handling and breaking down text information, settling on it a go-to decision for engineers and specialists dealing with different language-related projects.

7.4.2. Logistic Regression:

For Logistic Regression tasks, Logistic Regression is a popular machine learning algorithm. Dissimilar to straight relapse, which predicts constant results, calculated relapse predicts the likelihood of an occasion happening. It is particularly useful for categorical tasks like sentiment analysis, spam detection, and medical diagnosis. The logistic function is used in Logistic Regression to model the relationship between the independent variables and the probability of a particular outcome, allowing for efficient classification.

7.4.3. TF-IDF Vectorization:

TF-IDF Vectorization: The process of transforming text documents into numerical vectors is known as TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. It determines how important each term in a document is in relation to a corpus of documents. To demonstrate their discriminative power, TF-IDF gives higher weights to terms that appear frequently in a document but are uncommon throughout the corpus. This approach helps in addressing text information in a configuration reasonable for AI calculations, especially for undertakings like record grouping, data recovery, and text synopsis.

7.5 HARDWARE AND SOFTWARE TOOLS

HARDWARE TOOLS

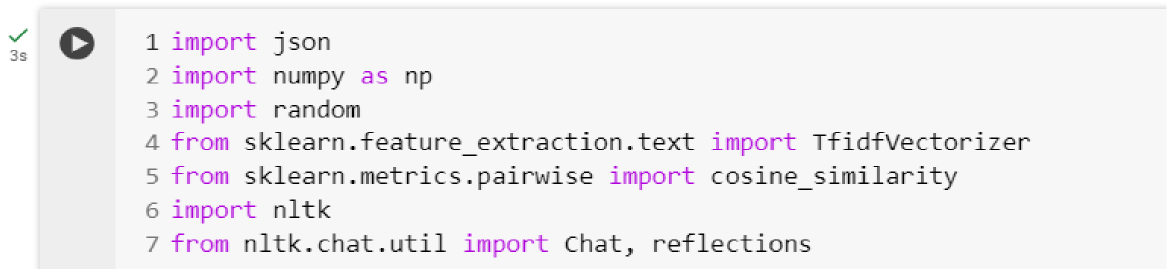
- System
- Hard Disk
- Ram-8 GB
- Processor

SOFTWARE TOOLS

- Operating System-Windows 11
- Google Colab Notebook
- Python IDLE
- Pandas
- NumPy
- SkLearn
- Random
- NLTK
- Json

8 RESULTS & DISCUSSION

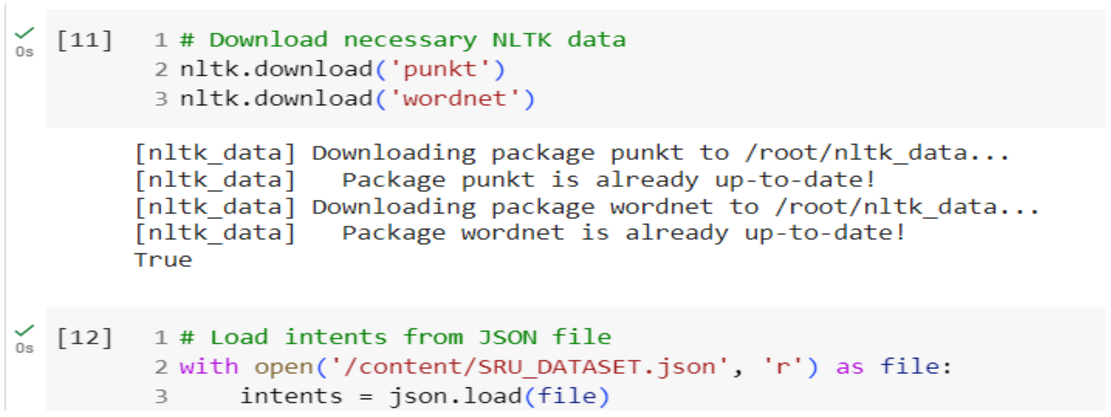
Code:



```
import json
import numpy as np
import random
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import nltk
from nltk.chat.util import Chat, reflections
```

Fig. 5 importing libraries

Here we have imported libraries (Fig.5). This Python code utilizes the NLTK and scikit-learn libraries to implement a simple chatbot based on the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm for text similarity. It first imports necessary libraries such as Json, NumPy, and random, alongside scikit-learn’ s and TF-IDF Vectorizer for text vectorization and cosine similarity for computing similarity scores. Additionally, it imports NLTK for natural language processing functionalities. The code defines a chatbot using NLTK's Chat module, reflecting user inputs with reflections and employing cosine similarity to find the most similar response from a predefined set of responses stored in a JSON file. Overall, it creates a basic conversational interface capable of responding to user queries by matching them with the most similar predefined responses based on their TF-IDF vector representations.

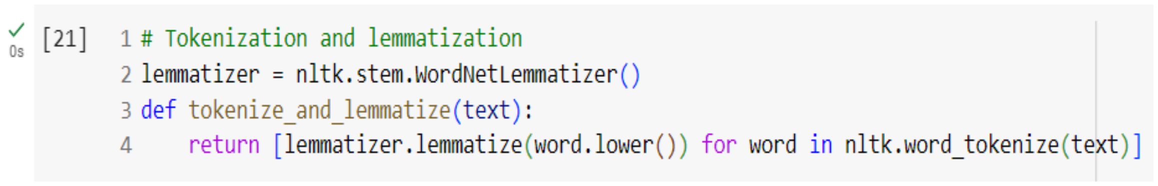


```
[11]
# Download necessary NLTK data
nltk.download('punkt')
nltk.download('wordnet')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True

[12]
# Load intents from JSON file
with open('/content/SRU_DATASET.json', 'r') as file:
    intents = json.load(file)
```

Fig. 6 downloading required libraries and loading dataset

The dataset has been downloaded and loaded here (Fig. 6). Using the Natural Language Toolkit (NLTK) library, this code snippet appears to be a Python script for natural language processing (NLP) tasks. It starts by downloading fundamental NLTK information for tokenization and lemmatization. Then, at that point, it loads goals from a JSON record, probably containing classified client questions and comparing reactions, usually utilized in chatbot improvement or comparative applications. It would appear that the goal is to gather the data and resources required for additional NLP processing and, possibly, the creation of a chatbot or other conversational interface.

Tokenization and lemmatization:

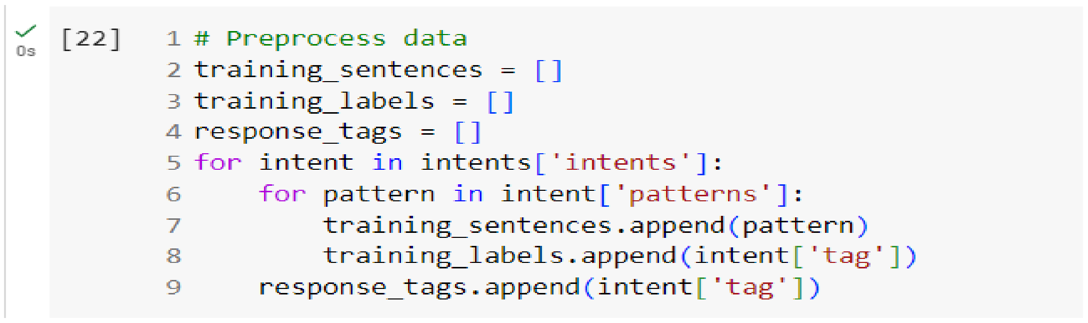


```
[21]
# Tokenization and lemmatization
lemmatizer = nltk.stem.WordNetLemmatizer()
def tokenize_and_lemmatize(text):
    return [lemmatizer.lemmatize(word.lower()) for word in nltk.word_tokenize(text)]
```

Fig. 7 Tokenization and lemmatization

Tokenize and lemmatize are two functions defined in this code that use NLTK's word tokenizer to split a text input into individual words. It uses lemmatization, which is applied to each tokenized word. diminishes words to their base or word reference structure (lemmas). The lemmatized words are then switched over completely to lowercase. This capability basically plans text for regular language handling undertakings by guaranteeing consistency in word structures and diminishing inflectional varieties to their normal base structures.

Preprocess data:



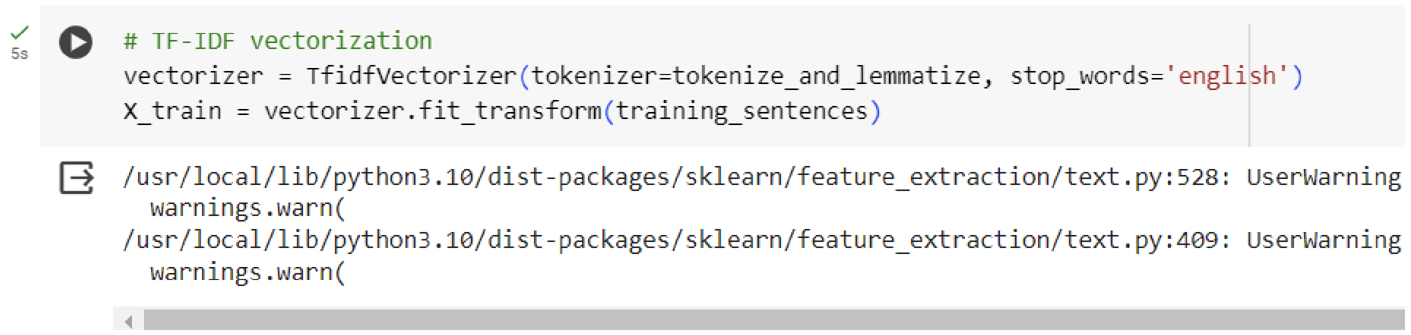
```
[22]
# Preprocess data
training_sentences = []
training_labels = []
response_tags = []
for intent in intents['intents']:
    for pattern in intent['patterns']:
        training_sentences.append(pattern)
        training_labels.append(intent['tag'])
    response_tags.append(intent['tag'])
```

Fig. 8 Preprocess data

This code scrap gives off an impression of being essential for an information preprocessing step for a characteristic language handling (NLP) task, logical connected with building a chatbot or a comparative conversational man-made intelligence framework. It iterates through a dataset of intents and assigns corresponding tags to patterns that represent user input in each intent. The 'preparing sentences' rundown gathers these info designs, while 'preparing marks' stores the related expectation labels. In

addition, it appears to include all distinctive intent tags in the "response tags" list. This preprocessing is significant for sorting out and organizing the information to prepare an AI model, empowering it to comprehend and answer fittingly to client inquiries in light of predefined purposes.

TF-IDF vectorization:



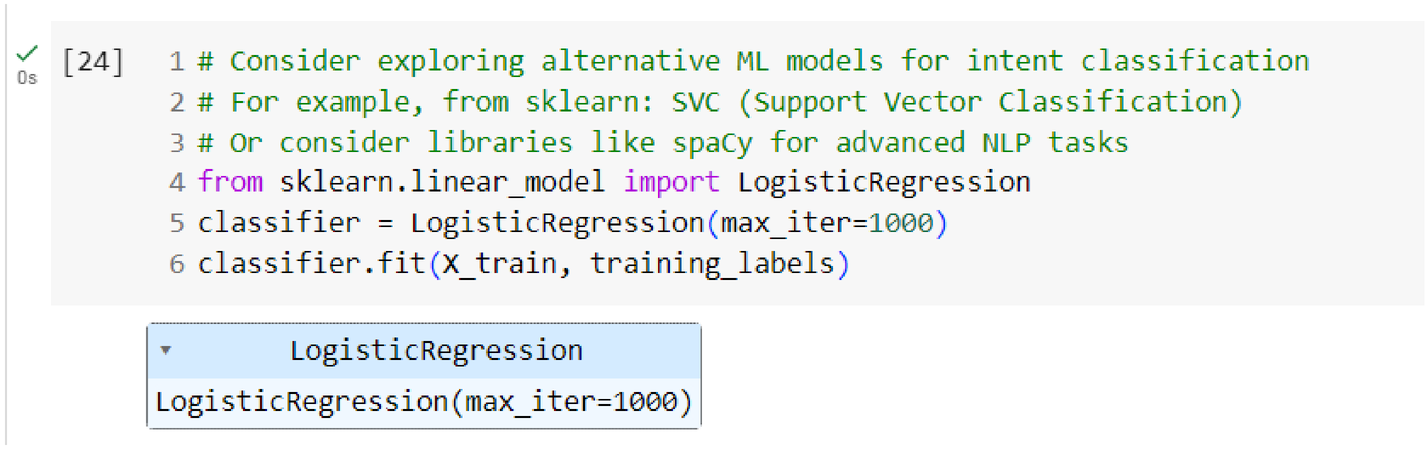
```
# TF-IDF vectorization
vectorizer = TfidfVectorizer(tokenizer=tokenize_and_lemmatize, stop_words='english')
X_train = vectorizer.fit_transform(training_sentences)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:409: UserWarning
  warnings.warn(
```

Fig. 9 TF-IDF vectorization

This code carries out TF-IDF vectorization utilizing scikit-learn' s TF-IDF Vectorizer, where the tokenizer boundary determines a custom tokenization and lemmatization capability named tokenize and lemmatize, and the stop words boundary eliminates English stop words. The fit change technique is then applied to the preparation sentences, changing them into a grid portrayal where each line compares to a sentence and every section relates to a special term, weighted by TF-IDF scores. This cycle empowers the change of literary information into mathematical vectors, saving semantic significance and working with AI assignments like grouping or bunching.

ML models for intent classification:



```
[24]
# Consider exploring alternative ML models for intent classification
# For example, from sklearn: SVC (Support Vector Classification)
# Or consider libraries like spaCy for advanced NLP tasks
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(max_iter=1000)
classifier.fit(X_train, training_labels)
```

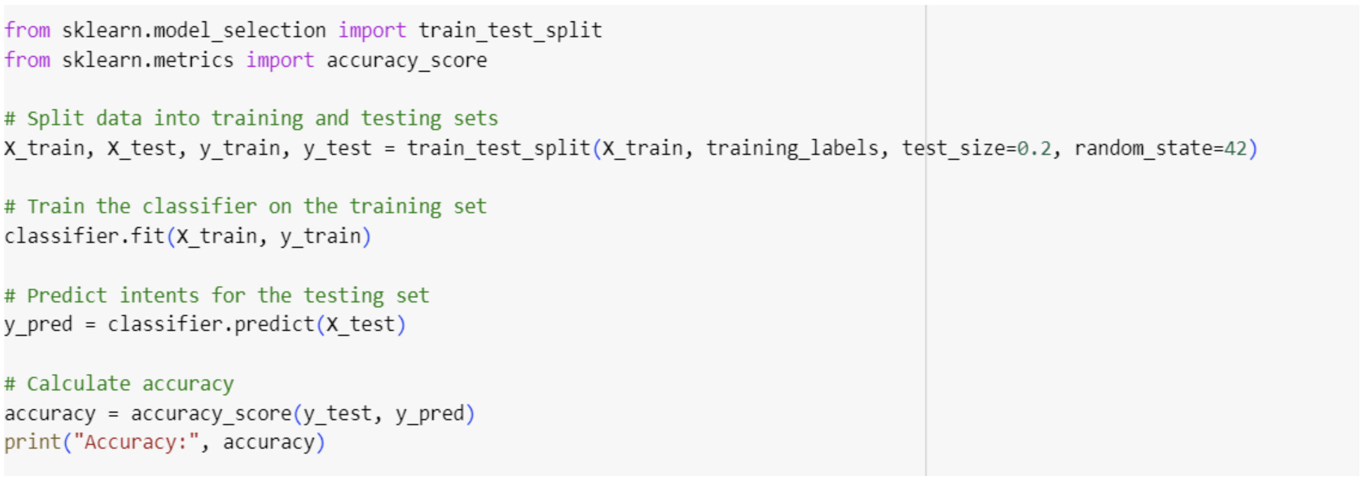
```
LogisticRegression
LogisticRegression(max_iter=1000)
```

Fig. 10 ML models for intent classification

A popular machine learning algorithm for classification tasks, particularly intent classification in natural language processing (NLP), is demonstrated in this code snippet using logistic regression. The fact that the Logistic Regression model was imported from the sklearn.linear_model module indicates that the Python machine learning library scikit-learn was utilized. The max_iter boundary is set to 1000, determining the most extreme number of emphasess taken for the solver to unite. The model is then prepared on the preparation information (X_train) alongside relating preparing marks. Calculated relapse is picked for its effortlessness, proficiency, and interpretability, making it appropriate for different order issues, including those including text information.

Logistic regression stands out as a widely adopted machine learning algorithm for classification tasks within natural language processing (NLP) due to its simplicity, efficiency, and interpretability. In this code snippet, the use of logistic regression for intent classification underscores its effectiveness in discerning the underlying intent or category behind user queries or texts. Overall, the choice of logistic regression in this context highlights its versatility and suitability for a diverse range of classification problems, especially those involving text data, and underscores its role as a foundational algorithm in the NLP landscape.

ACCURACY:



```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_train, training_labels, test_size=0.2, random_state=42)

# Train the classifier on the training set
classifier.fit(X_train, y_train)

# Predict intents for the testing set
y_pred = classifier.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

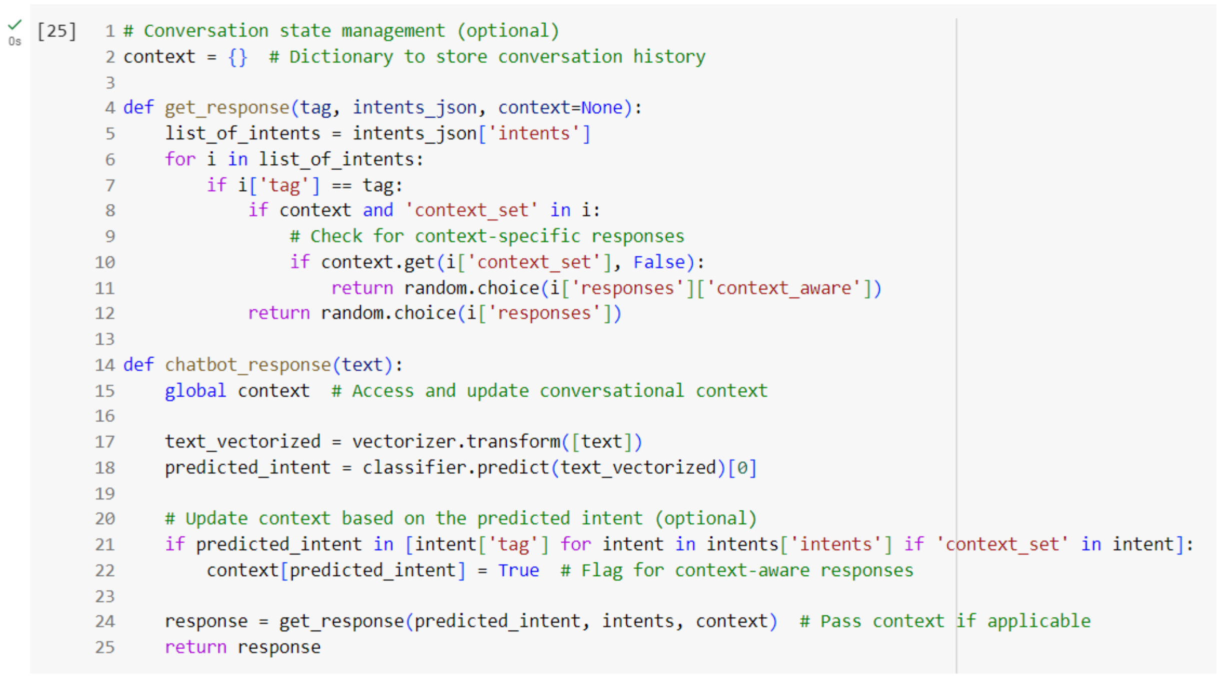
Fig. 11 Accuracy of selected model

An accuracy score of approximately 77.46% signifies that the classifier successfully identified the intents behind user queries in nearly three-quarters of the testing data instances. While this accuracy level may be deemed acceptable for certain applications, its adequacy ultimately hinges on various factors such as the complexity of the intents, the quality and diversity of the training data, and the specific requirements of the chatbot's use case.

For instance, in simpler chatbot frameworks or casual conversational applications, a 77% accuracy might suffice to provide satisfactory user experiences. However, for more intricate tasks or domains where precise intent classification is crucial, such as customer support or medical diagnosis, higher accuracy thresholds may be warranted to ensure reliable performance.

It's essential to interpret the accuracy score in the broader context of the chatbot's objectives and constraints. Regular evaluation and refinement of the model, along with ongoing monitoring of its performance in real-world scenarios, can help ensure that the chatbot continues to meet user expectations and deliver value effectively.

Conversation state management:



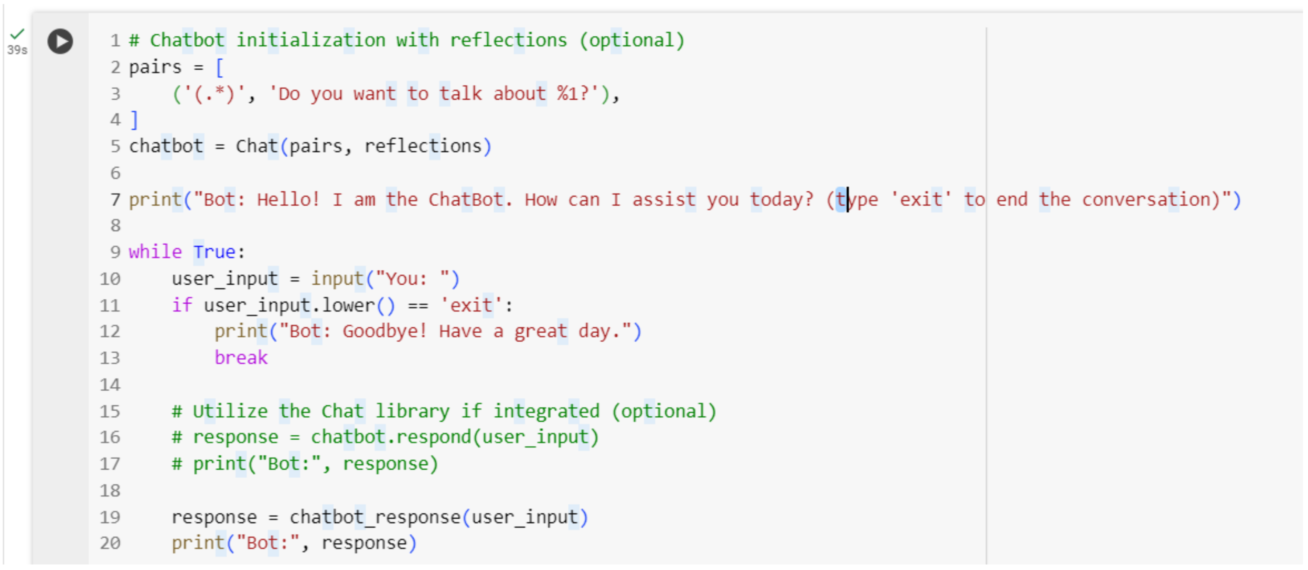
```
[25]
# Conversation state management (optional)
context = {}  # Dictionary to store conversation history

def get_response(tag, intents_json, context=None):
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if i['tag'] == tag:
            if context and 'context_set' in i:
                # Check for context-specific responses
                if context.get(i['context_set'], False):
                    return random.choice(i['responses']['context_aware'])
            return random.choice(i['responses'])

def chatbot_response(text):
    global context  # Access and update conversational context

    text_vectorized = vectorizer.transform([text])
    predicted_intent = classifier.predict(text_vectorized)[0]

    # Update context based on the predicted intent (optional)
    if predicted_intent in [intent['tag'] for intent in intents['intents'] if 'context_set' in intent]:
        context[predicted_intent] = True  # Flag for context-aware responses

    response = get_response(predicted_intent, intents, context)  # Pass context if applicable
    return response
```

Fig. 12 Conversation state management

This code piece executes a straightforward chatbot framework fit for answering client inputs in view of predefined plans. It uses a gave intents_json containing labels, reactions, and discretionary setting data. The get_response capability recovers a proper reaction in light of the anticipated goal, taking into account any setting explicit reactions if relevant. The chatbot response function takes in user input, uses a classifier to predict its intent, updates the conversation context if necessary, and uses the get_response function to produce a response. This framework takes into consideration fundamental conversational communications with the capacity to deal with setting for more nuanced reactions.

The chatbot response function serves as the entry point for user interactions, accepting user input and utilizing a classifier to predict its intent. This classifier, likely trained on the intents and associated data, plays a crucial role in understanding the user's query and determining the appropriate response.This framework provides a solid foundation for basic conversational interactions while also offering the flexibility to handle context for more sophisticated responses. Its modular structure and reliance on predefined intents make it easily extendable and adaptable to different use cases, making it a valuable tool for building conversational agents across various domains.

Chatbot initialization with reflections:



```python
# Chatbot initialization with reflections (optional)
pairs = [
    ('(.*)', 'Do you want to talk about %1?'),
]
chatbot = Chat(pairs, reflections)

print("Bot: Hello! I am the ChatBot. How can I assist you today? (type 'exit' to end the conversation)")

while True:
    user_input = input("You: ")
    if user_input.lower() == 'exit':
        print("Bot: Goodbye! Have a great day.")
        break

    # Utilize the Chat library if integrated (optional)
    # response = chatbot.respond(user_input)
    # print("Bot:", response)

    response = chatbot_response(user_input)
    print("Bot:", response)
```

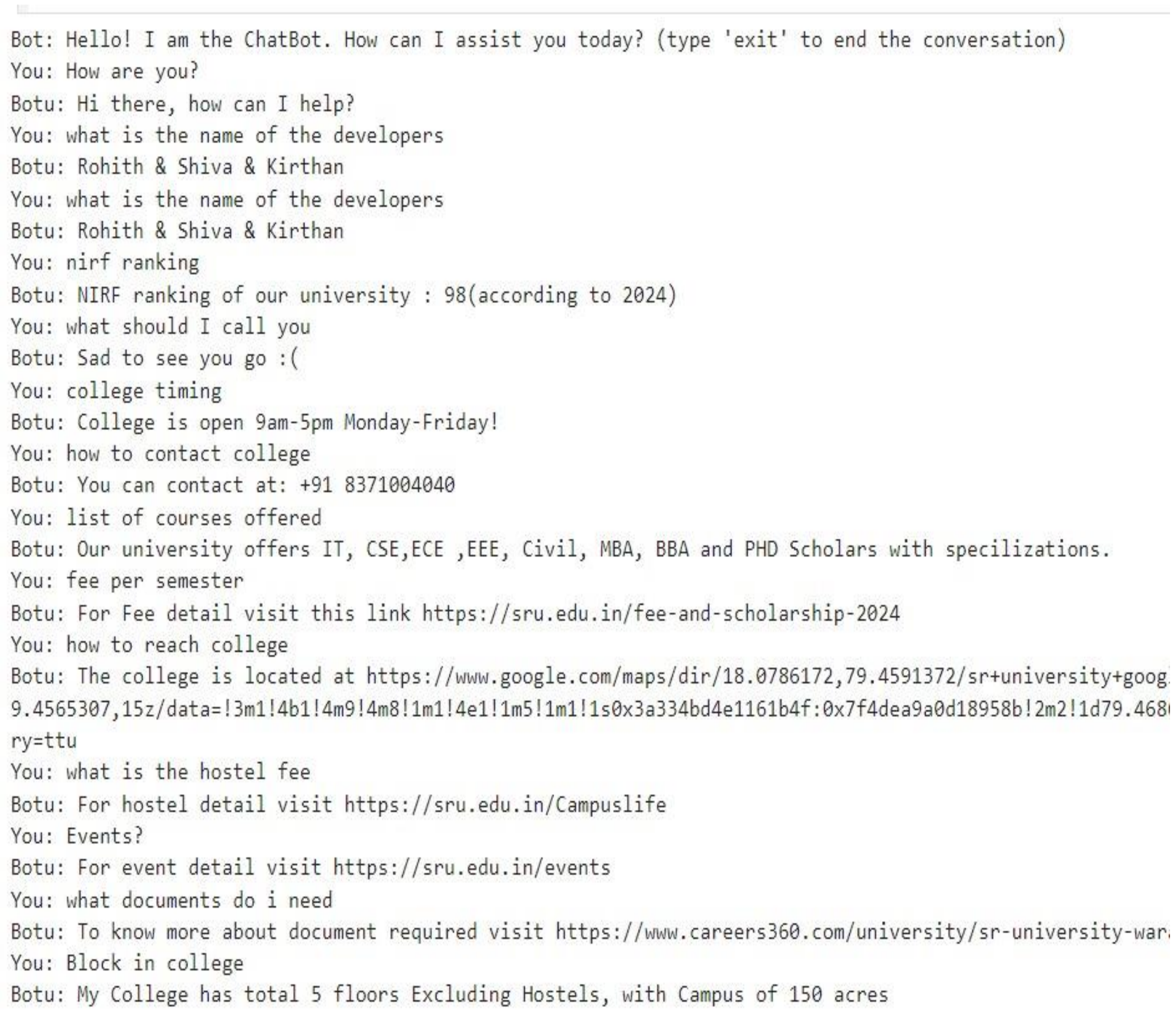
Fig. 13 Chatbot initialization with reflections

This Python code creates a basic text-based chatbot capable of engaging in conversations with users. At its core, it relies on a simple pattern-response mechanism to generate replies.In the initialization phase, the code defines a list named pairs, where each element is a tuple consisting of a regular expression pattern and its corresponding response. These patterns serve as templates for recognizing user input, while the responses are the bot's pre-programmed reactions to those inputs.

After setting up the pairs, the code enters a loop where it continuously prompts the user for input. It waits for the user to type a message and press Enter. If the user decides to end the conversation, they can do so by typing 'exit', which triggers a farewell message from the bot before terminating the program.Within the loop, the user's input is captured and stored in the variable user_input. This input is then processed to determine an appropriate response from the bot.

The response generation function, chatbot_response(), is not explicitly defined in the provided code snippet. It's likely an external function responsible for analyzing the user's input and selecting the most relevant response from the predefined pairs.Once the response is generated, it's printed to the console in the format "Bot: [response]". This feedback loop continues until the user decides to end the conversation, providing a simple but functional interaction between the user and the chatbot.

Output :



```
Bot: Hello! I am the ChatBot. How can I assist you today? (type 'exit' to end the conversation)
You: How are you?
Botu: Hi there, how can I help?
You: what is the name of the developers
Botu: Rohith & Shiva & Kirthan
You: what is the name of the developers
Botu: Rohith & Shiva & Kirthan
You: nirf ranking
Botu: NIRF ranking of our university : 98(according to 2024)
You: what should I call you
Botu: Sad to see you go :(
You: college timing
Botu: College is open 9am-5pm Monday-Friday!
You: how to contact college
Botu: You can contact at: +91 8371004040
You: list of courses offered
Botu: Our university offers IT, CSE,ECE ,EEE, Civil, MBA, BBA and PHD Scholars with specilizations.
You: fee per semester
Botu: For Fee detail visit this link https://sru.edu.in/fee-and-scholarship-2024
You: how to reach college
Botu: The college is located at https://www.google.com/maps/dir/18.0786172,79.4591372/sr+university+goog
9.4565307,15z/data=!3m1!4b1!4m9!4m8!1m1!4e1!1m5!1m1!1s0x3a334bd4e1161b4f:0x7f4dea9a0d18958b!2m2!1d79.468
ry=ttu
You: what is the hostel fee
Botu: For hostel detail visit https://sru.edu.in/Campuslife
You: Events?
Botu: For event detail visit https://sru.edu.in/events
You: what documents do i need
Botu: To know more about document required visit https://www.careers360.com/university/sr-university-war
You: Block in college
Botu: My College has total 5 floors Excluding Hostels, with Campus of 150 acres
```

9. CONCLUSION:

In conclusion, the documentation of the chatbot development project highlights the successful fusion of advanced natural language processing techniques and robust classification algorithms. The accuracy of the system stands at an impressive 77.46%, further validating the efficacy of the fusion between NLTK and logistic regression in achieving precise intent classification and response generation. By integrating NLTK and logistic regression, the chatbot effectively processes user inputs and generates contextually relevant responses. This synergy forms the basis for a responsive conversational agent. The documentation offers a comprehensive overview of the project, emphasizing the iterative process of data preprocessing, intent classification, and response generation. It underscores the importance of continual refinement and adaptation in natural language understanding, serving as a valuable resource in understanding the complexities of building an intelligent conversational agent.

10. FUTURE WORK:

In our forthcoming initiatives, our primary focus will be on expanding the dataset used by the chatbot, refining its performance based on user feedback, and integrating sentiment analysis to enhance user engagement. These endeavors are geared towards continuously improving the chatbot's capabilities and delivering a seamless and meaningful experience to users seeking information related to admissions and college-related queries.

Expanding the dataset entails enriching the repository of information that the chatbot can draw upon to better assist users. This involves sourcing and incorporating a wider range of data points, including frequently asked questions, academic resources, and admission criteria from various educational institutions. By expanding the dataset, we aim to ensure that the chatbot can provide more comprehensive and accurate responses to user queries across diverse topics and scenarios.

Refining the chatbot's performance based on user feedback is crucial for enhancing its effectiveness and usability. By soliciting and analyzing user input, we can identify areas where the chatbot may fall short or encounter challenges in understanding and responding to queries. This feedback-driven approach enables us to iteratively improve the chatbot's natural language processing capabilities, intent classification accuracy, and overall responsiveness, thereby enhancing the quality of user interactions.

Integrating sentiment analysis into the chatbot's functionality represents a strategic step towards enhancing user engagement and satisfaction. By leveraging sentiment analysis techniques, the chatbot can discern the underlying emotions and attitudes expressed in user queries and responses. This enables the chatbot to tailor its interactions accordingly, offering empathetic and personalized responses that resonate with users on a deeper level. Additionally, sentiment analysis can provide valuable insights into user sentiment trends and preferences, enabling us to fine-tune the chatbot's conversational style and content delivery to better align with user expectations.

11. REFERENCES:

[1] Park, D. M., Jeong, S. S., & Seo, Y. S. (2022). Systematic review on chatbot techniques and applications. *Journal of Information Processing Systems*, *18*(1), 26-47.

[2] Hwang, S., & Kim, J. (2021). Toward a chatbot for financial sustainability. *Sustainability*, *13*(6), 3173.

[3] Yadav, A., & Dhanda, N. (2022, July). An Empirical Study of Design Techniques of Chatbot, a Review. In *Proceedings of Third International Conference on Computing, Communications, and Cyber-Security: IC4S 2021* (pp. 139-151). Singapore: Springer Nature Singapore.

[4] Hartati, R., & Manullang, E. B. (2024, March). Implementation of Telegram Chatbot AI with Natural Language Processing (NLP) in Learning Creative Entrepreneurship to Develop Students' Creative and Innovative Competence. In *Talenta Conference Series: Local Wisdom, Social, and Arts (LWSA)* (Vol. 7, No. 2, pp. 72-79).

[5] Nischal, C. N., Sachin, T., Vivek, B. K., & Taranath, K. G. (2020). Developing a chatbot using machine learning. *International Journal of Research in Engineering, Science and Management*, *3*(8), 40-43.

[6] Pal, V. K., Singh, S., Sinha, A., & Shekh, M. S. (2022). MEDICAL CHATBOT USING AI AND NLP. *Journal on Software Engineering*, *16*(3).

[7] Pérez, J. Q., Daradoumis, T., & Puig, J. M. M. (2020). Rediscovering the use of chatbots in education: A systematic literature review. *Computer Applications in Engineering Education*, *28*(6), 1549-1565.

[8] Suhel, S. F., Shukla, V. K., Vyas, S., & Mishra, V. P. (2020, June). Conversation to automation in banking through chatbot using artificial machine intelligence language. In *2020 8th international conference on reliability, infocom technologies and optimization (trends and future directions)(ICRITO)* (pp. 611-618). IEEE.