



Design and Development of An Automated Conversational Chatbot Using Natural Language Processing and NLTK

Migisha Jim

Department of Electrical, Telecommunication and Computer Engineering (ETC),
School of Engineering and Applied Sciences,
Kampala International University-Western Campus, Uganda.

**Corresponding Author: Migisha Jim; jimmigisha@kiu.ac.ug*

Abstract: The Emergence of chatbots represents a pivotal advancement in Artificial Intelligence (AI) and Machine Learning (ML), impacting various sectors like Healthcare, Sports, Automation, and Education. Specifically, a recent application in education involves crafting chatbots capable of engaging with students, aiming to enhance the learning experience. This research paper targets Bachelor's programs, particularly in Computer Science Engineering and allied programs, by introducing a software bot named AI Sciences chatbot. Leveraging NLTK-Corpus library and Machine Learning models, this bot focuses on aiding the learning process of fundamental AI Science concepts within B.Sc Computer Science. The proposed system is to develop an online software bot that enables direct interaction between users and the bot, empowering students to pose queries without human intervention by using NLTK features and the ML model. The chatterbot is capable and proficient in comprehending and responding to inquiries based on a pre-defined dictionary, supplementing its knowledge by autonomously seeking information. The primary objective of this research is to implement a software bot within an online learning platform, facilitating seamless access for students to seek guidance, ask questions, and receive support 24/7, fostering a more efficient approach to their studies without constant human intervention.

Keywords: Python. Artificial Intelligence, ML, AI sciences chatbot, NLTK-Corpus, Punkt, Tokenization, BS4.

1. Introduction

The integration of chatbots stands out as a groundbreaking technological advancement within Artificial Intelligence. These AI-powered chatbot systems have opened up new vistas across various Industries[1]. Chatbots serve a dual purpose in the educational sphere: honing students' communication skills while aiding teaching faculty in delivering automated lectures[1]. AI significantly shapes our daily engagements[2], with chatbot systems emerging as a favored AI technology supporting teaching and learning activities [3]. These chatbots embody a fusion of AI and Human-computer interface HCI models[4], driving research in online learning platforms geared toward leveraging chatbot technology for educational purposes. Incorporating this technology into educational learning

platforms signifies a pivotal approach toward augmenting and enriching the learning experience [5]. The education avenues for interaction include email communications, student-to-student engagements, teacher-student interactions, and self-learning experiences, all offering greater convenience for students. The application of AI through chatbot technology has the potential to furnish a more personalized and immersive learning environment[5][6]. Functioning as a conversational interactive agent, chatbots furnish responses to user queries [7], offering pertinent information to students ranging from course specifics to practice questions and answers [5][8]. The traditional education system grapples with several challenges, including overcrowded classrooms and a lack of personalized attention for students with varying learning paces and styles; coupled



with the rapid evolution of technology and traditional lecture-based teaching, these issues can hinder effective learning[9]. The entering of AI-powered chatbots is a promising solution to address these educational hurdles. In real-time education, AI-powered chatbots primarily assist students in their studies, enhance the learning experience and foster communication skills. This paper introduces an educational chatbot named 'Chatterbot,' explicitly designed for learning evaluation. The system employs a rule-based conversational approach and utilizes features from the Natural Language Tool Kit (NLTK) and Machine learning models. Chatterbots serve as software programs that facilitate user-bot communication, guiding users in asking questions through Automated interactions. Whether via text or voice, these conversational rule-based chatbots allow users to inquire about topics related to AI sciences. The Educational Chatterbot engages with students in Realtime, leveraging NLTK and Machine learning to provide information. Automating repetitive processes saves time and enhances the overall learning experience. Students can efficiently interact with this software bot across various study areas.

2. Research Problem and Motivation

2.1 Overview of Chatbot Technology

A chatbot is a computer program that interacts with users, answering their questions and delivering appropriate responses[10]. These programs mimic and process human communication, allowing people to engage with digital devices as if interacting with a natural person[11]. Chatbot systems automatically respond to human queries[12]. Chatbot technology became popular after Alan Turing proposed the Turing test[13], which explores whether machines can think like humans. Artificial intelligence & Machine Learning Techniques (AIML) and Pattern Matching techniques define chatbot operations[14]. The first recognized chatbot, Eliza, was created in 1966 to return user input as questions[15]. ALICE, another early chatbot, was considered a pioneer in human-computer interaction[16]. As technology advanced, modern chatbots like Smarter child emerged[17]. Notable intelligent chatbots include Apple Siri, Amazon Alexia, IBM Watson, Google Assistant, and Microsoft Cortana [18].

2.2 Chatbot for Education

In today's technological landscape, chatbots are increasingly utilized for learner interaction. As communication and various activities predominantly occur on online platforms, chatbots enhance student engagement and support automated teaching[19]. Their role in e-learning innovation is pivotal, creating an interactive learning environment akin to personalized one-on-one interactions. Chatbots play a crucial part in enhancing

individual students' skills. Numerous studies have highlighted the benefits of integrating chatbots into educational systems, spanning various applications.

2.2.1. Content Integration

Chatbots play a crucial role in education by facilitating easy access to essential information related to specific subject assignments and timetables. Authorized students can conveniently receive this information from online platforms. Whether studying subjects or checking schedules, chatbots assist anytime, anywhere [20].

2.2.2. Motivation and Interactivity

In today's educational landscape, students increasingly seek to acquire knowledge and access information about specific subjects through online platforms rather than relying solely on textbooks and notes[21]. Chatbots, as interactive systems, facilitate efficient learning in a comfortable environment. Students can acquire knowledge and enhance learning outcomes by engaging with conversational agents. In colleges and universities, teachers deliver lectures, and students interact with them in classrooms, fostering positive relationships [22]. The integration of chatbots within educational systems supports increased student interaction levels and is pivotal in motivating students to maintain regular engagement and expedite the learning process[23].

2.2.3. Teaching & Learning Tool

Researchers have developed numerous e-learning tools to enhance the education system. Chatbots serve as conversational agents among these tools, fostering student interaction and collaboration [24]. When engaging with a chatbot, the teachers can observe the types of questions students ask, identify subject problem areas, and assess students' learning abilities [25]. Chatbots play a pivotal role in the education system by providing an interactive learning platform as a form of social learning[26]. These chatbots offer personalized advice, conduct online assessments, respond to student queries, and track their progress. Chatbots provide an easy and cost-effective way to deliver educational content, streamlining the teaching-learning process.

2.2.4. Quick-Access Chatbots

Quick-access chatbots enhance students' learning efficiency and facilitate quick access to educational information. Additionally, chatbots are utilized as social learning platforms, allowing individuals from diverse backgrounds to share their views and address specific issues by adapting to individual learning needs[27].

2.2.5. Language Learning Concepts

Pham et al. [28] developed an English learner chatbot deployed on mobile devices for user communication. This chatbot automatically sends queries to

students and proposes solutions to multiple-choice questions. Additionally, students can learn new language concepts through a learning tool called API Helper, which assists novice Android programmers in understanding API usage[29]. Zaw et al. [30] Created a Java programming tool for learning various exercise problems across different study topics. Computer science technology programming courses can be challenging to grasp in higher education due to complex concepts.

A Python-Bot was developed to simplify the teaching and learning methods. This chatbot provides an easy-to-use platform for students to enhance their Python programming skills[3]. Furthermore, AI technology in education holds immense potential. It improves student learning outcomes and enables flexible learning anytime,

anywhere. Innovative teaching and learning processes benefit from integrating AI technology [31].

2.2.6. Development of learning platforms

A Chatbot development platform is an application tool used to create a chatbot. Several platforms are employed for building chatbots, including Microsoft Azure Bot, Google's Dialogflow, Amazon's Alexa, Facebook Bots for Facebook Messenger, and Snatchbot[3][28]. Among these, the Python Bot developed using Snatchbot supports software programmers by integrating machine learning capabilities through APIs. This enables the addition of advanced features to the chatbot. The Python-Bot serves educational purposes and assists students in learning Python programming concepts by providing a structured framework, as shown in Figure.1



Figure.1 Existed methodology of Teaching Python Bot[3].

In the Existing system, the development of Python-Bot involved utilizing the SnatchBot API, which leverages a range of pre-defined tools driven by various NLP algorithms. An intriguing aspect of SnatchBot is a web platform designed for creating chatbots using an Application programming interface(API)[3][28]. The unique feature enables the deployment of chatbots on websites and social networks. This configuration allows students to access Python-Bot easily using smartphones or laptops[3]. This accessibility benefits novice programmers seeking to learn Python programming as they interact with Python-Bot. These models play a crucial role in generating suitable dates and times for meetings by considering pre-defined available options. Furthermore, an automatic email notification is triggered to alert the tutor/lecturer

about the scheduled meeting. Upon confirmation by the tutor, the students can engage in a face-to-face discussion. In orchestrating meeting appointments, the python-Bot

emphasizes its role in facilitating in-person support for learners [3]. The process of teaching Python-Bot is developed using the algorithm format[3].

Algorithm-1: Python-Bot Information Decision

```

Input: Topic (this is the topic deduced from the text
Enter edby the user);
If Topic Matches Message Bank
then
    return predefined response
  
```

```

(:TopicResponse)//returns pre-defined
responsessuch that the response is in
the collection of responses for the
topic;
else
return default and select a topic);
end

```

Existing Algorithm of Teaching a
Python-Bot [3]

3. Methodology

The study involves a comprehensive literature review of various chatbots for diverse educational activities. Our proposed system introduces AI Sciences Chatbot, a software bot developed explicitly for learning evaluation. Students can directly pose questions or express doubts to this chatbot. The system employs a rule-based conversational approach, utilizing the NLTK-Corpus library and machine learning models. The chatbot is an intermediary that facilitates communication between users and the bot. It responds to automated queries, guiding students in effectively framing their questions. Using text-based interactions, the conversational rule-based chatbot allows students to engage with it naturally. Leveraging NLTK's WordNet Lemmatization, PUNKT tokenizer, and Machine learning models, the AI Sciences chatbot provides real-time information and creates an interactive learning environment. This approach saves time through automated pre-processes and enhances communication efficiency across various study areas.

3.1. System Architecture

The shown in above Figure 2 Proposed System architecture is implemented in this paper. Users pose questions or express doubts directly by interacting with the chatbot through speech or text. After that, natural language understanding was used, and the NLTK_corpus library with a machine learning model was used to create a chatterbot. And core code of the program through which the chatbot runs on the system. To import Tokenization data to the model creator, create sentence tokenization and word tokenization, and use Sentiment Analysis application by using Natural Language Understanding(NLU), which makes it easier for machines to understand the context. And perform processed_text for using NLTK. Stem.

After importing greeting inputs to the bot, WordNetLemmatizer uploads input data(Collection of Questions from AI Sciences) to the chatterbot using some instructions and commands to generate the bot using dialogue management and form a response. After forming a response, a machine learning model from the sci-kit learn library will be used to create an offering response from input data using Natural Language Generation(NLG). The software technology automatically transforms data into simple English language after the final Bot response. "Hello, I am from AI Sciences chatbot "can you ask me any questions regarding Artificial Intelligence."

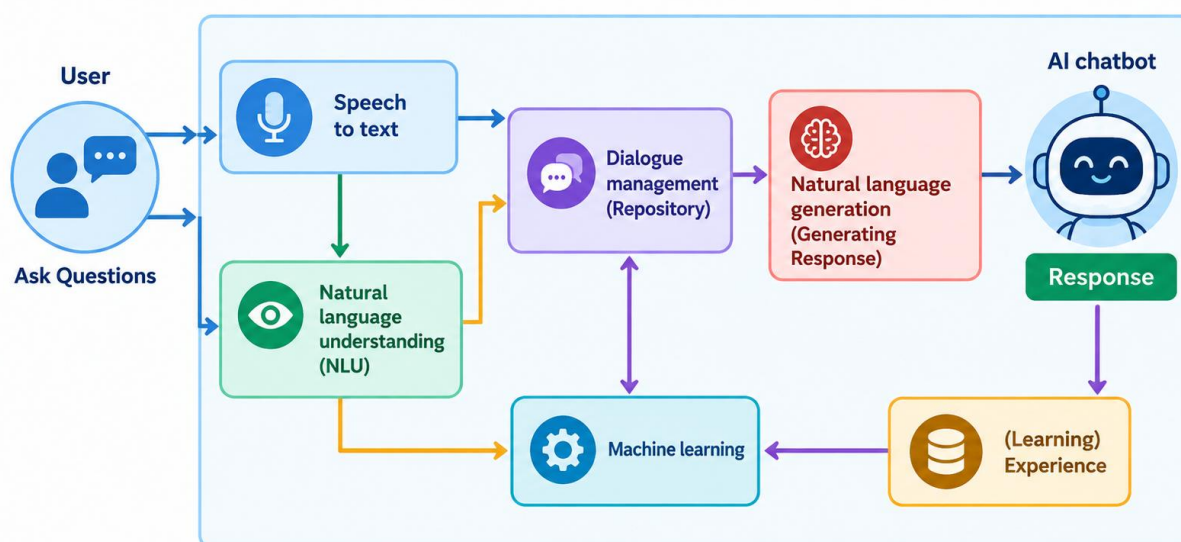


Figure. 2 Proposed System Architecture

3.2 Algorithm-2: (Build aAI Sciences ChatBot)

Step-1: To create the chatbot using NLTKCorpus.
Step-2: The core code of the program through which the chatbot runs on the system
Step-3: To import urlopen&bs4 library functions allow the

chatbot to send HTTP requests to retrieve data from webpages.

Step-4: To read the content from the source link 'urlopen' function after doing the content parsing.

Step-5: To import Tokenization data to the model, the creator.

Step-6: To create sentence tokenization and word tokenization

Step-7: To perform processed_textfor using nltk.Stem.WordNetLemmatizer

Step-8: Import greeting inputs to the bot

Step-9: Import sci-kit-learn machine learning libraries to feature_extraction' text module and 'metrics. A pairwise module is used for chatbot development.

Step-10: From clean-feature extraction.text import Tfidf vectorizer function is used in chatbot development for text vectorization and feature extraction methods. Language-specific pre-processing is used to improve text quality and remove stopwords in the text to reduce noise data.

Step-11: From sklearn. metrics. pairwise import cosine_similarity function is used in chatbot development to measure the similarity between text inputs, enabling semantic similarity chatbot to assess how similar the current input is to past interactions. This helps the chatbot understand the context of the conversation and identify relevant responses based on previous user interactions.

3.2 Functional working of a chatbot using NLTK

The chatbot identifies the most similar previous interaction to the current user input based on the cosine similarity scores calculated on the value function the chatbot uses 'flat.sort()' is a flattened array or arranged list of values sorted in ascending order to determine the index of the previous interaction with the highest similarity to the current user input. The index of the interaction is stored in 'IDX.' The chatbot allows an organized and systematic arrangement of values to make information effective. The index('IDX') of the most similar previous interactions is identified, and the chatbot can access the corresponding previous information or responses from its memory.

The TF-IDF scores represent the importance of terms in distinguishing documents or pieces of text; by sorting these scores in descending order, the chatbot can identify the most relevant keywords in the text data. After sorting TF-IDF scores, the chatbot may want to select the top keywords that represent the context of the conversation the most.

The 'req_tfidf = flat[2]' function retrieves the second-highest value from a sorted array of TF-IDF scores. These scores are used for keyword identification in text data. The chatbot can extract import topics from the conversion, allowing the chatbot to understand users' input.

Chatbots dynamically create human-like responses to user inputs the chatbot checks the condition 'if (req_tfidf == 0)' to check whether the second highest TF-IDF score ('req_tfidf') is equal to zero if they are filtered out as

stopwords don't appear in the document in a chatbot allows for the detection and handling zero scores the chatbot to identify cases where they are no significant terms to consider further processing ensuring robustness and efficiency the chatbot is ready to response generation process is Natural language response generation(NLG).

Chatbot' robo_response = robo_response + sent_tokens[idx]' is used to append a sentence from a list of sentences('sent_tokens') to the response generated by the chatbot ('robo_response'). In conversational flow systems, maintaining a history of users' past interactions in['idx'] is essential for effective dialogue management in the list of sentences ('sent_tokens').

It can help the chatbot better understand the user's query and the user's response to 'sent_tokens.' The chatbot can keep previous interactions turned in the conversation by using NLG and this information to guide its response. Adding sentences from the list of sentences('sent_tokens'), the selected sentence is likely to be related to a particular topic, and the content of the previous interaction is retrieved using 'IDX' to inform the natural language generation of the chatbot to provide a more contextually relevant response to the user queries.

Upload input data to the AI sciences chatbot and use instructions and commands to generate the forming response.

Chatbot is generated from a sci-kit learn model to learn the collection of questions from AI Sciences.

Import Bot Response from NLG

Collection of Questions from AI sciences

Bot -Response: ("Can ask me any question regarding Artificial_intelligence")

END.

3.3. Working Model and Implementation

By installing all the necessary modules and required libraries. This research involves developing a software-based AI sciences chatbot using Python libraries NLTK_Corpus and a machine learning model within a desktop application function with Jupyter Notebook. The chatbot serves as an intriguing means of interaction, allowing users to engage with a software bot without human intervention. The chatbot development process entails several steps: First, we import the NLTK libraries and upload the link-relevant data from urllibrequests. After importing the necessary components, we create a model for tokenizing data_sentences and word_sentences from NLTK WordNet.Next, we processed text data using lemmatization. It generated a bot-forming response after using the sci-kit-learn library. As a

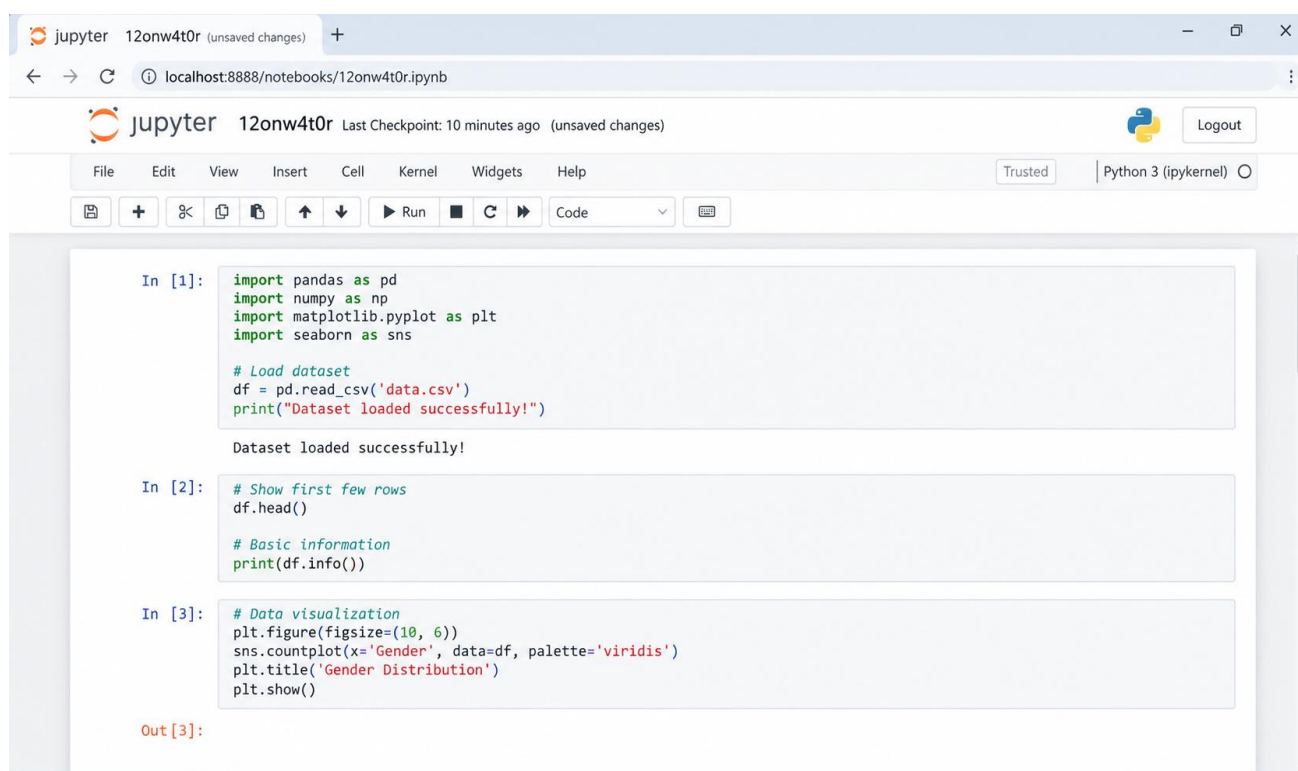
result, the chatbot responded to greetings and asked questions about AI science concepts.

4. Results & Discussion

4.1. Installation modules from NLTK

python libraries are a core element of the Python programming language. AI chatbots rely on Natural language processing(NLP) and machine learning techniques to understand and respond to user input. In Python, sklearn refers to sci-kit-learn, a popular machine learning library of the feature_extraction.text module within sci-kit-learn utilities for working with text data for

pre-processing using TF-IDF Vectorizer. In the context of chatbots 'Cosine_similarity' from sklearn.metrics Pairwise could be used to evaluate the similarity between different text inputs. Installing libraries such as the Natural Language Tool Kit (NLTK) allows developers to incorporate AI capabilities into their chatbots. Installing NLTK along with the 'punkt' tokenizer and Wordnetcorpus is a suite of tools and resources for NLP tasks. These components enable developers to perform various text processing and analysis tasks and develop an application such as an AI chatbot. The code is shown in Figure 3.



```

In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv('data.csv')
print("Dataset loaded successfully!")

Dataset loaded successfully!

In [2]: # Show first few rows
df.head()

# Basic information
print(df.info())

In [3]: # Data visualization
plt.figure(figsize=(10, 6))
sns.countplot(x='Gender', data=df, palette='viridis')
plt.title('Gender Distribution')
plt.show()

Out [3]:

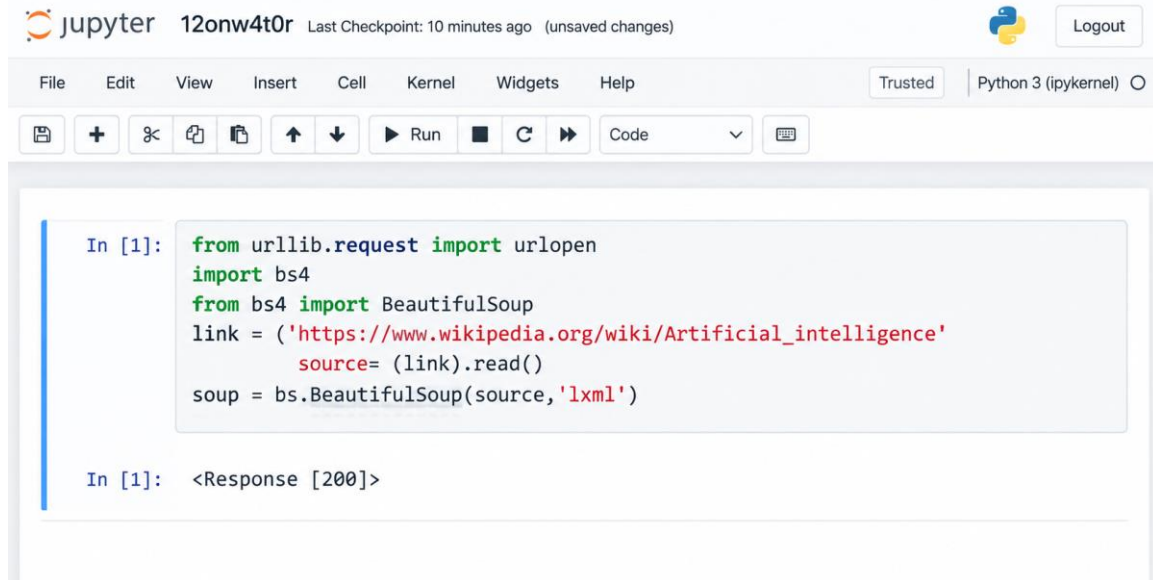
```

Figure.3. Installation required modules and libraries from NLTK

AI chatbots rely on Natural language processing(NLP) and machine learning techniques to understand and respond to user input. In Python, sklearn refers to sci-kit-learn, a popular machine learning library of the feature_extraction.text module within sci-kit-learn utilities for working with text data for pre-processing using TF-IDF Vectorizer. In the context of chatbots 'Cosine_similarity' from sklearn.metrics Pairwise could be used to evaluate the similarity between different text inputs. Installing libraries such as the Natural Language Tool Kit (NLTK) allows developers to incorporate AI capabilities into their chatbots. Installing NLTK along with the 'punkt' tokenizer and Wordnetcorpus is a suite of tools and resources for NLP tasks. These components enable developers to perform various text processing and analysis tasks and

develop an application such as an AI chatbot. The code is shown in Figure 3.

4.2 Import urlopen&Bs4: In the second step process in chatbots, the 'urlopen' function is from 'urllib. Request' is a versatile tool for accessing data from external sources in chatbot development. The 'Beautifulsoup' function allows the chatbot to make HTTP requests to retrieve data from external resources. The chatbot can send HTTP requests to these APIs or webpages and receive the corresponding responses. Installing and importing 'Beautifulsoup' from the 'bs4' library is useful for parsing and extracting information from HTML or XML documents, making it easier for the chatbot to extract specific data elements from webpages to read the content from the source link of 'urlopen' function. The code is shown in Figure 4.



```

In [1]: from urllib.request import urlopen
import bs4
from bs4 import BeautifulSoup
link = ('https://www.wikipedia.org/wiki/Artificial_intelligence'
        source= (link).read())
soup = bs.BeautifulSoup(source,'lxml')

In [1]: <Response [200]>

```

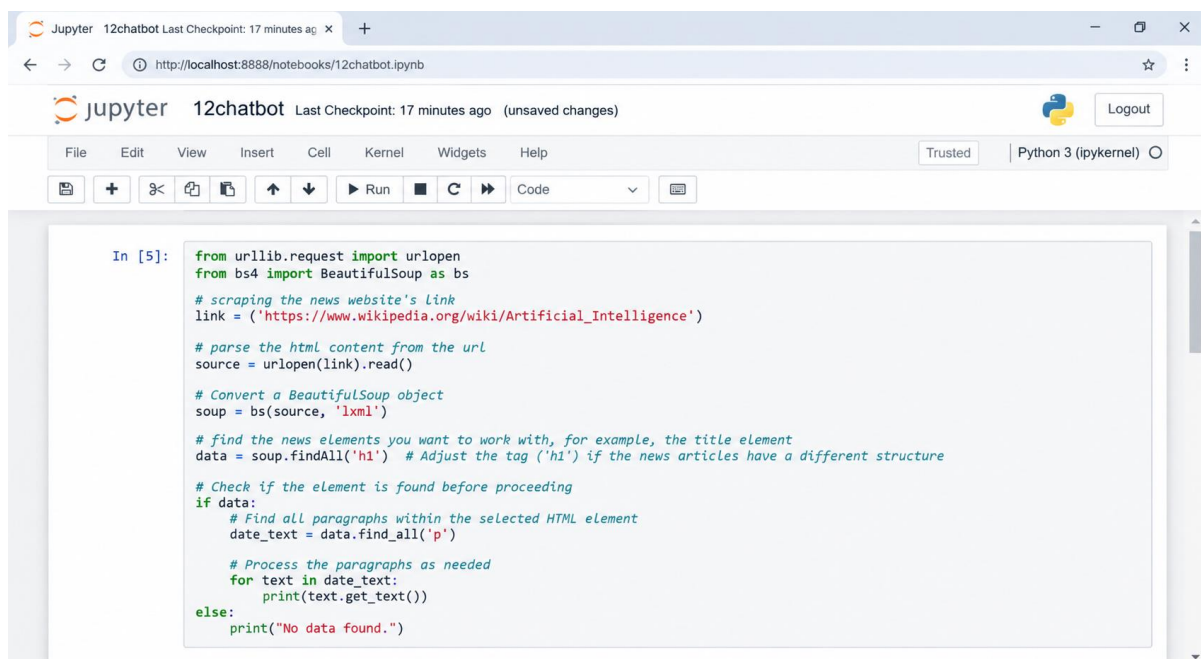
Figure. 4. Read the link for input (AI sciences) from urllib. Request import urlopen function and import BeautifulSoup from bs4.

4.3 Content parsing: The third step is importing the 'urlopen' function from urllib: request and import BeautifulSoup from bs4. The valid link through the 'European' function enables the chatbot to send HTTP requests to this webpage. After that, fetch the HTML content from the URL to read the content of 'urlopen' (link). After that, creating a BeautifulSoup object can be incredibly useful for extracting relevant information from HTML documents when the chatbot needs to interact with web-based content in web content parsing.

4.3.1 Web Content Parsing: Chatbots must scrape webpages from specific information to respond to user

queries. 'Facilitates the parsing of HTML or XML documents, making extracting relevant data such as text, links, or other structured content easier.

4.3.2 Web scraping: 'Beautifulsoup is a Python web scraping library that allows us to parse XML or HTML pages.' BeautifulSoup integrates well with other Python libraries used for web scraping, such as requests to send HTTP requests to access all paragraphs selected within the HTML element and process the data_text needed from the target web pages to get the data. The code is shown in Figure 5.



```

In [5]: from urllib.request import urlopen
from bs4 import BeautifulSoup as bs
# scraping the news website's link
link = ('https://www.wikipedia.org/wiki/Artificial_Intelligence')
# parse the html content from the url
source = urlopen(link).read()
# Convert a BeautifulSoup object
soup = bs(source, 'lxml')
# find the news elements you want to work with, for example, the title element
data = soup.findAll('h1') # Adjust the tag ('h1') if the news articles have a different structure
# Check if the element is found before proceeding
if data:
    # Find all paragraphs within the selected HTML element
    date_text = data.find_all('p')
    # Process the paragraphs as needed
    for text in date_text:
        print(text.get_text())
else:
    print("No data found.")

```

Figure. 5 Assuming a valid link to read the link from the source to create an object using bs4 the paragraphs needed to text data.

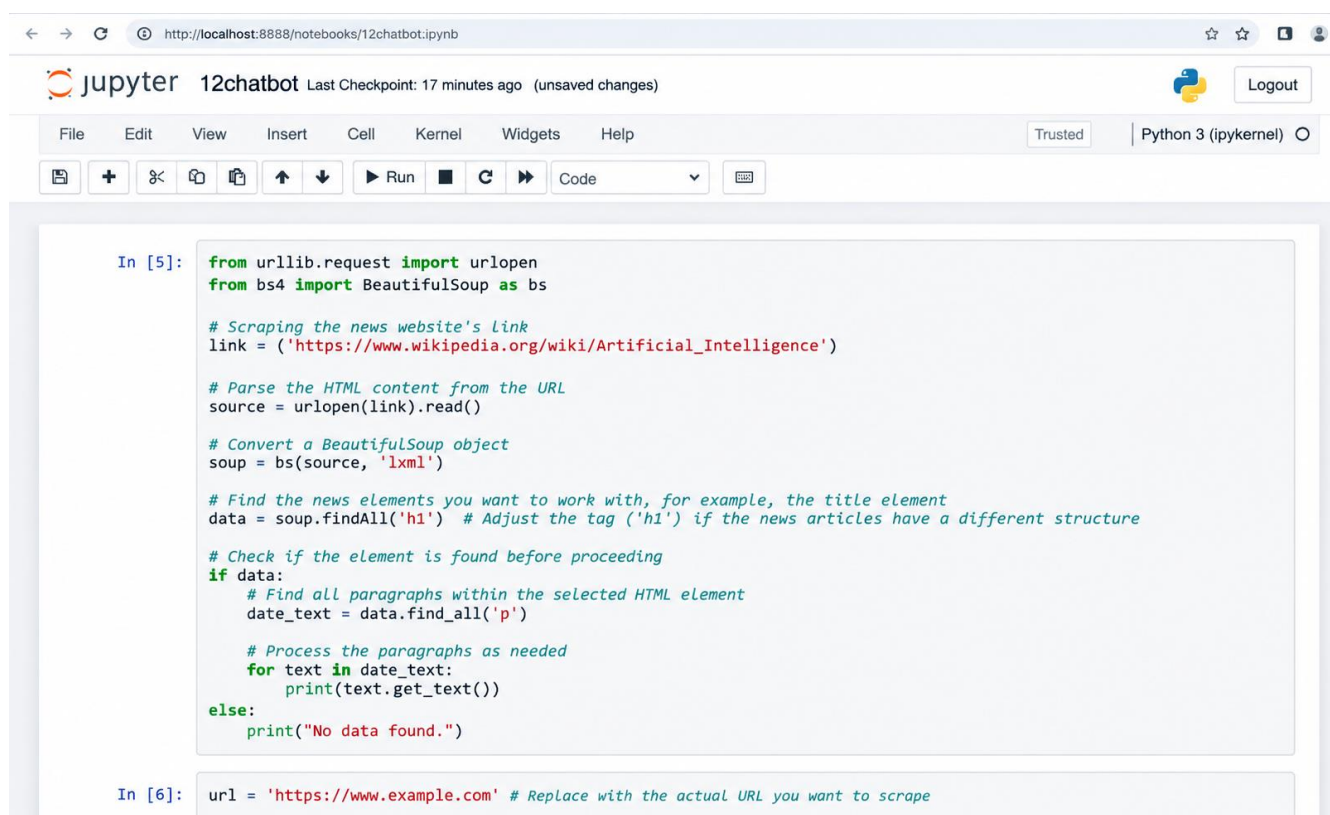
4.4 Tokenization: After completing the web scraping, do the tokenization process for data_text. It is a fundamental step in natural language processing. The tokens are usually words, punctuation marks, and meaningful text elements the chatbot can analyze. Tokenization in a chatbot is the process of breaking down a piece of text into sentences or words into individual units called tokens. It enables the chatbot to understand and generate human-like responses. In the process, the 'punkt' tokenizer and Wordnetcorpus tokenizer are used in chatbot development. It is used to convert the data_text into a list of sentences. 'punkt' tokenizer is used it is a pre-trained sentence tokenizer included NLTK.nltk.sent_tokenize(data_text), with a list of sentences called sent_tokens in a chatbot. It is used for Understanding User input.

4.4.1 Understanding user input: When a user interacts with the chatbot by entering the text, it may contain multiple sentences. It is trained to split the text into individual sentences, making it easier to process and understand each sentence separately. The code is shown in Figure 6. After the tokenization process, sentences in data_text convert the list of words. 'Wordnet 'corpus is a pre-trained word_tokenizer that is included with NLTK. It is a lexical database of English words and their semantic relationships, such as synonyms, and it provides information about their meanings and relationships.

They were using 'word_tokens' generated by nltk.word_tokenize(data_text), with a list of words called word_tokens in a chatbot. It is used for text processing.

4.4.2 Text processing: Word tokenization breaks down the sentences of text into individual words, providing the chatbot with a structured representation of the input data that can efficiently analyze user input and understand the user's intent. The code is shown in Figure 6.

4.5 Lemmatization: In this step, the process of Lemmatizer is a WordNet semantically oriented dictionary of English included in NLTK. Lemmatization is the process of reducing the words to their dictionary form. In a chatbot, a function 'def LemTokens(Tokens)' is used for the Lemmatizing tokens, which help improve the chatbot's text normalization and semantic understanding and performance in handling natural language input. In a chatbot development, another function, "dict((word(punct)," is used to remove punctuations in text data. It helps normalize such variations, making it easier for the chatbot to recognize meaningful information from the text. The code is shown in Figure 7.



```

In [5]: from urllib.request import urlopen
        from bs4 import BeautifulSoup as bs

        # Scraping the news website's Link
        link = ('https://www.wikipedia.org/wiki/Artificial_Intelligence')

        # Parse the HTML content from the URL
        source = urlopen(link).read()

        # Convert a BeautifulSoup object
        soup = bs(source, 'lxml')

        # Find the news elements you want to work with, for example, the title element
        data = soup.findAll('h1') # Adjust the tag ('h1') if the news articles have a different structure

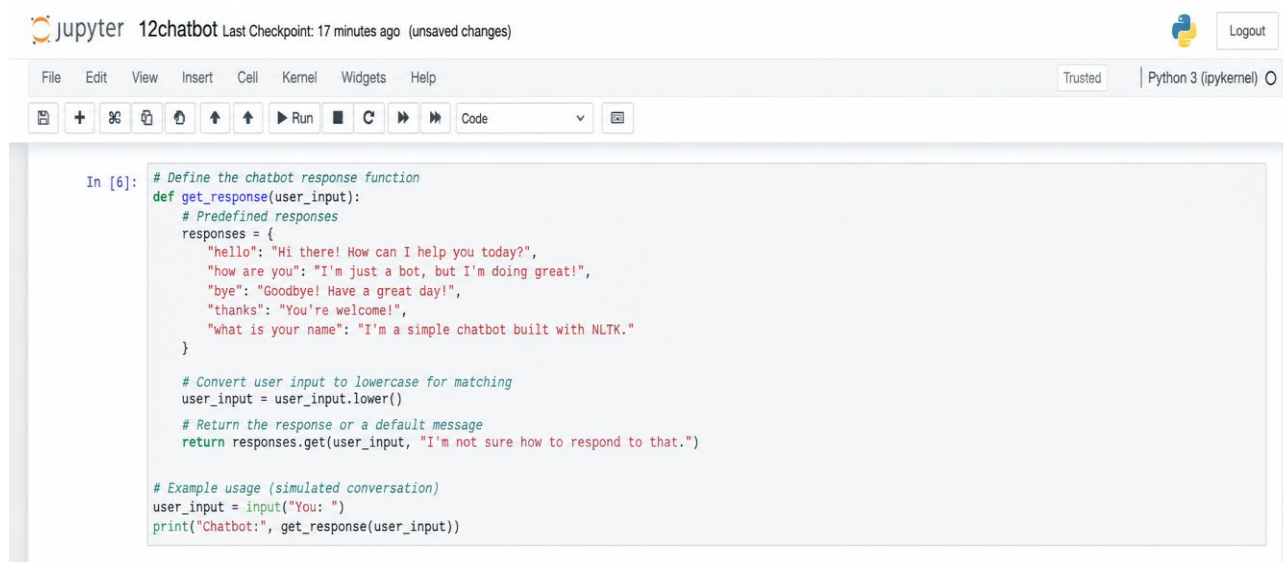
        # Check if the element is found before proceeding
        if data:
            # Find all paragraphs within the selected HTML element
            date_text = data.find_all('p')

            # Process the paragraphs as needed
            for text in date_text:
                print(text.get_text())
        else:
            print("No data found.")

In [6]: url = 'https://www.example.com' # Replace with the actual URL you want to scrape

```

Figure.6.Tokenization from nltk (data_text) to convert a list of sentences and a list of words.



```

In [6]: # Define the chatbot response function
def get_response(user_input):
    # Predefined responses
    responses = {
        "hello": "Hi there! How can I help you today?",
        "how are you": "I'm just a bot, but I'm doing great!",
        "bye": "Goodbye! Have a great day!",
        "thanks": "You're welcome!",
        "what is your name": "I'm a simple chatbot built with NLTK."
    }

    # Convert user input to lowercase for matching
    user_input = user_input.lower()

    # Return the response or a default message
    return responses.get(user_input, "I'm not sure how to respond to that.")

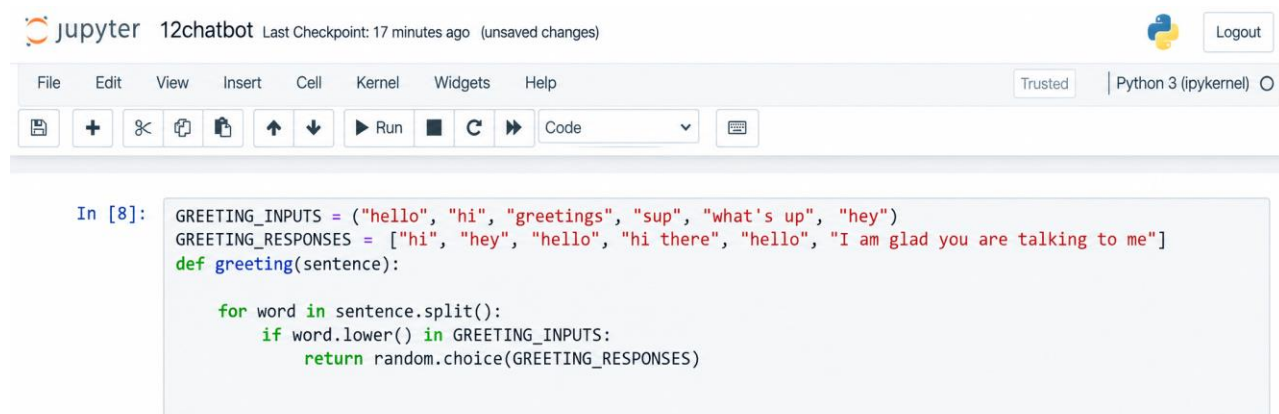
# Example usage (simulated conversation)
user_input = input("You: ")
print("Chatbot:", get_response(user_input))

```

Figure.7. Lemmatizer is processed_text, a semantically-oriented dictionary of English.

4.6 Greeting Inputs: In this step, define a function 'for a word in a sentence.split()' method is used to split the input sentences into individual words based on whitespace or delimiters. The chatbot enables the process of analysis of text data, such as text understanding. After that, the list of words allows the chatbot to process each word separately;

then, using a GREETING_INPUT function of the list of words, the chatbot recognizes the user is greeting it. After identifying the greetings from the user, the chatbot will understand the context of the conversation and respond appropriately. The code is shown in Figure 8.



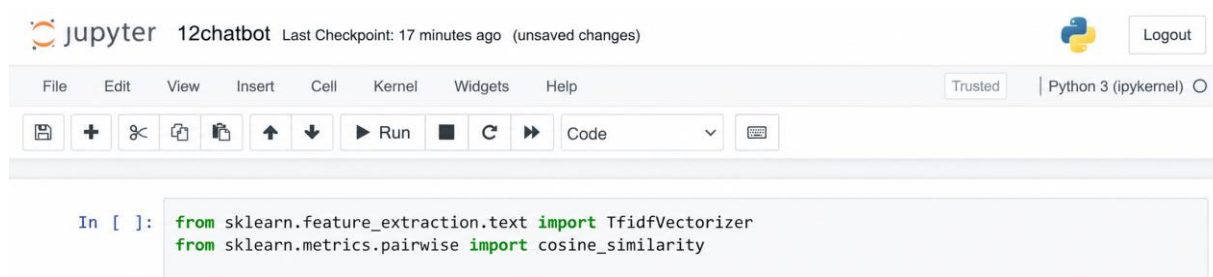
```

In [8]: GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's up", "hey")
GREETING_RESPONSES = ["hi", "hey", "hello", "hi there", "hello", "I am glad you are talking to me"]
def greeting(sentence):

    for word in sentence.split():
        if word.lower() in GREETING_INPUTS:
            return random.choice(GREETING_RESPONSES)

```

Figure. 8. Greeting Functions of inputs and responses.



```

In [ ]: from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

```

Figure. 9. import sci-kit-learn libraries

4.7 Import sci-kit-learn libraries: In this step import the sci-kit-learn machine learning libraries from the 'feature_extraction.text' module and 'metrics.pairwise'

module is imported to develop a chatbot. The scikit-learn contains the utilities for working with text data. The 'TfidfVectorizer' is a class within sci-kit-learn of 'feature_extraction'. The text module is used for text data pre-processing and feature extraction

vectorization. After that, the process of 'cosine_similarity' from the 'sci-kit-learn.metrics.pairwise' module is used for chatbot development for measuring the similarity between

4.8. TF-IDF vectorization: In this step, the chatbot uses the 'TfidfVectorizer' function is used for text vectorization. Feature extraction methods are used to improve the quality of the text. 'stop_words =' English for language-specific pre-processing is used to reduce the noise data.

4.8.1 Text Vectorization: 'TfidfVectorizer' function converts text data into numerical vectors for the text pre-processing step. The tfidfvectorizer applies the TF-IDF transformation to the input text data('sent_tokens'); the TF-IDF vectorization converts each document or sentence in the collection into numerical vector representation for each dimension corresponds to a unique term (word) in the vocabulary in pre-defined corpus library.

4.8.2 Text Representation: TF-IDF vectors represent the text data in a high dimensional space; each dimension corresponds to a unique term (word) in the vocabulary. The values in the vectors reflect both the frequency of each term in the document(TF) and across the entire collection of documents (IDF), allowing the chatbot to capture the distinguishing documents.

4.8.3 Feature Extraction: The TF-IDF vectors serve as input features from text processing tasks in the chatbot. These vectors encode semantic information about the text data, capturing the importance of each term (word) in vocabulary and representing the textual content in a numerical format. The chatbot learns patterns and relationships in data to analyze and understand the text data and make informed decisions based on the semantic content of the text.

4.8.4 Language-specific pre-processing: In the chatbot, specifying 'stop_words='english' is used for English-language text processing. Stopwords are common words that appear frequently to have semantic meanings in the text. The setting of the 'stop_words='English' function instructs with 'Tfidfvectorizer' to remove English stop words in the text before vectorization. Removing stopwords can improve the quality of TF-IDF vectors and reduce noise data. The code is shown in Fig.10.

4.9. Cosine_similarity: 'cosine_similarity' is a function provided by the 'sklearn.metrics.pairwise' module is used in chatbot development for measuring the similarity between text inputs, enabling various tasks such as semantic similarity, identifying most similar past interactions, and accessing previous context.

4.9.1 Semantic similarity: In chatbot using function' vals = cosine_similarity(tfidf[-1], tfidf)' computes the cosine

the text inputs, enabling semantic similarity, contextual understanding, and Response generation. The code is shown in Figure 9.

similarity between the TF-IDF vector representation of latest user input('tfidf[-1]') and all other TF-IDF vectors representing past interactions or responses(tfidf), using these functions the chatbot can assess how similar the current input to past interactions this helps the chatbot can understand the context of conversation and identify relevant responses based on previous user interactions.

4.9.2 Identified most similar interaction: The Chatbot performs 'idx = vals.argsort()[0][-2]' function is used to retrieve the index of the identified most similar previous interaction to the current user input based on the cosine similarity scores calculated on the value function the chatbot using 'flat.sort()' is a flattened array or arranged list of values sorted in ascending order to determine the index of the previous interaction with the highest similarity to the current user input. The index of the interaction is stored in 'IDX'. The chatbot allows an organized and systematic arrangement of values to make information effective.

4.9.3 Accessing the previous context: The index('IDX') of the most similar previous interactions is identified so the chatbot can access the corresponding previous information or responses from its memory. In the next step of chatbot contexts' req_tfidf = flat[-2]' function is used to retrieve the second highest value from a sorted array of TF-IDF scores; they might be done by identifying Relevant content, selecting top keywords, contextual understanding, and Response generation.

4.9.4 Identifying relevant content: The TF-IDF scores represent the importance of terms in distinguishing documents or pieces of text; by sorting these scores in descending order, the chatbot can identify the most relevant keywords in the text data.

4.9.5 Selecting Top keywords: The second highest TF-IDF score ('flat[-2]') corresponds to the second most crucial term in the text data; after sorting TF-IDF scores, the chatbot may want to select the top keywords that are the most representative context of the conversation.

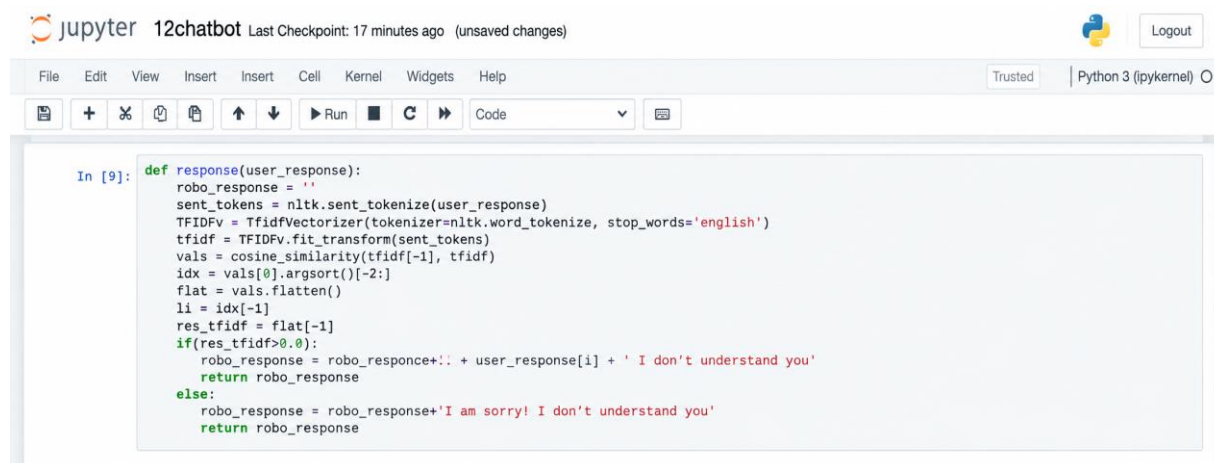
4.9.6 Contextual understanding: TF-IDF scores are used for keyword identification in text data; by receiving the second highest TF-IDF score, the chatbot can extract essential topics from the conversation, allowing it to understand the main ideas the chatbot to understand the user's intent and its response accordingly.

4.9.7 Natural language Response generation: It is used in chatbots to create human-like responses to user dynamic inputs. The chatbot checks the

condition 'if (req_tfidf == 0)' is used to check whether the second highest TF-IDF score ('req_tfidf') is equal to zero if they are filtered out as stopwords don't appear in the document in a chatbot allows for the detection and handling zero scores the chatbot to identify cases where they are no significant terms to consider further processing ensuring robustness and efficiency the chatbot is ready to response generation process. In the next step, the chatbot 'robo_response = robo_response + sent_tokens[idx]' is used to append a sentence from a list of sentences ('sent_tokens') to the response generated by the chatbot ('robo_response').

4.9.8 Dialogue management: In conversational flow systems, maintaining a history of users' past interactions in ['idx'] is essential for effective dialogue management in the list of sentences ('sent_tokens') can help the chatbot better understand the user's query the user's response to 'sent_tokens' the chatbot can keep previous interactions turns in the conversation by using NLG and this information to guide its response.

4.9.9 Contextual Robot Response: Adding sentences from the list of sentences ('sent_tokens'), the selected sentence is likely to be related to a particular topic, and the content of the previous interaction is retrieved using 'IDX' to inform the natural language generation of the chatbot to provide a more contextually relevant response to the user queries. The code is shown in Figure 10.



```
In [9]: def response(user_response):
    robo_response = ''
    sent_tokens = nltk.sent_tokenize(user_response)
    TFIDFv = TfidfVectorizer(tokenizer=nltk.word_tokenize, stop_words='english')
    tfidf = TFIDFv.fit_transform(sent_tokens)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx = vals[0].argsort()[::-1]
    flat = vals.flatten()
    li = idx[-1]
    res_tfidf = flat[-1]
    if(res_tfidf>0.0):
        robo_response = robo_response+'' + user_response[i] + ' I don't understand you'
        return robo_response
    else:
        robo_response = robo_response+'I am sorry! I don't understand you'
        return robo_response
```

Figure.10. Generate Robo response

4.10.4 Greeting Recognition: In the chatbot, condition checks 'if(greeting(user_response)!=None)' function is designed to identify common greetings in the user's inputs such as "hello," hi, "hey" responding to greetings helps increase user engagement by creating a more interactive and welcome conversation environment.

4.10.5 Conversation continuity: It uses the chatbot function ("AI sciences: "+ greeting(user_response)) to print a response that includes a greeting recognized from the

4.10 Bot Response from AI Sciences: This step provides an introductory message to the user, informing them that they are interacting with a chatbot specializing in artificial intelligence (AI). The context of the conversation is for the users to ask questions about artificial intelligence, which the chatbot is ready and available to respond to on this particular topic.

4.10.1 User interaction: In chatbot 'user_response = input()', the line of code allows the chatbot to interact with users by prompting them to enter text input.

4.10.2 Normalization: In the next step, the process is 'user_response = user_response.lower()' function is used in the chatbot to help standardize the user's input and simplify text processing in the chatbot to prevent these issues from treated as "Hello" and "hello as two different inputs.

4.10.3 Conversational flow: The chatbot constructs conversational flow 'while(flag == True)' is typically used to create a loop controlled by a flag, the chatbot can maintain the flow of conversation and respond dynamically to the user's input.

user's input, allows the chatbot to reciprocate the greeting in a polite and friendly manner and print recognized input greeting along with response from the chatbot, it acknowledges the user's input and signals that the chatbot is actively engaging with the conversation it helps to build a connection with the user. Recognizing input greetings is part of managing the flow of conversation; it ensures that the chatbot acknowledges the user's input before proceeding to address any queries or provide assistance that the user may have requested.

4.10.6 Avoid Redundancy: After the user's input has been processed and used to generate a response, it may be desirable to remove it from the list of sentences in a chatbot using the 'sent_tokens.remove(user_response)' function to avoid redundancy that the chatbot ensures that subsequent responses are not biased towards repeating the same content it helps improve the quality and diversity of response provided by the AI sciences chatbot making the conversation and used to generate a response

4.10.7 Displaying Response: Using 'print("AI sciences:" + response(user_response))' in a chatbot allows for the display of the chatbot responses to the user input, contributing to a more interactive and practical conversation in real-time. It makes the interaction feel more dynamic and responsive chatbot is built from AI sciences. You can ask any questions regarding Artificial intelligence. The code is shown in Figure 11.

```

In [*]: flag=True
while(flag==True):
    user_que=input("👋 Hi, I am from AI chatbot, you can ask me any question regarding Artificial_Intelligence")
    while(flag==True):
        user_response = input("👋 You: ")
        if(user_response.lower()=="thanks" or user_response=="Thank you"):
            flag=False
            print("AI : Welcome!. You are welcome.")
        else:
            if(user_response=="AI":
                print("AI : AI stands for Artificial Intelligence.")
            elif(user_response=="ML"):
                print("AI : Machine Learning.")
            else:
                print("AI : Unknown.")
                sent_ = (user_response)

        flag=False
        print("AI : Reference : Byte1 takes care of yourself.")

Hello I am from AI chatbot, you can ask me any question regarding Artificial_Intelligence
  
```

Figure.11. Bot Response (Hello, I am from AISciences. You can ask any question regarding AI)

5. Conclusion

This work aimed to harness the power of Artificial intelligence through machine learning using the NLTK library. Our research endeavors to explore the field of machine learning, with chatbots serving as a focal point. These versatile chatbots find applications across diverse domains, including Healthcare, Sports, Automation, and Education. Recently, AI applications have made significant inroads into the education sector. Specifically, conversational chatbots have become a valuable tool for engaging with students. Our current work introduces the AI Sciences chatterbot, a software bot developed using the NLTK-Corpus library and Machine learning model.

References

- [1]. Dsouza, R., Sahu, S., Patil, R., & Kalbande, D. R. (2019, December). Chat with bots intelligently: A critical review & analysis. In *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)* (pp. 1-6). IEEE.
- [2]. Adamopoulou, Eleni, and Lefteris Moussiades. "Chatbots: History, technology, and applications." *Machine Learning with Applications* 2 (2020): 100006.
- [3]. Okonkwo, C. W., & Ade-Ibijola, A. (2020). Python-Bot: A chatbot for teaching Python programming. *Engineering Letters*, 29.
- [4]. Bansal, H., & Khan, R. (2018). A review paper on human-computer interaction. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*
- [5]. Cunningham-Nelson, S., Boles, W., Trouton, L., & Margerison, E. (2019). A review of chatbots in education: practical steps forward. In *30th annual conference for the Australian Association for Engineering Education (AAEE 2019): educators becoming agents of change: innovate, integrate, motivate* (pp. 299-306). Engineers Australia.
- [6]. Benotti, L., Martnez, M. C., & Schapachnik, F. (2017). A tool for introducing computer science with automatic formative assessment. *IEEE transactions on learning technologies*.
- [7]. Smutny, P., & Schreiberova, P. (2020). Chatbots for

- learning: A review of educational chatbots for the Facebook Messenger. *Computers & Education*, 151, 103862.
- [8]. Sinha, S., Basak, S., Dey, Y., & Mondal, A. (2020). An educational chatbot for answering queries. *Emerging technology in modeling and graphics*. Springer.
 - [9]. Labadze, L., Grigolia, M., & Machaidze, L. (2023). Role of AI chatbots in education: systematic literature review. *International Journal of Educational Technology in Higher Education*, 20(1), 56
 - [10]. Clarizia, F., Colace, F., Lombardi, M., Pascale, F., & Santaniello, D. (2018). Chatbot: An education support system for students. In *Cyberspace Safety and Security: 10th International Symposium, CSS 2018, Amalfi, Italy, October 29–31, 2018, Proceedings 10* (pp. 291-302). Springer International Publishing.
 - [11]. Ciechanowski, L., Przegalinska, A., Magnuski, M., & Gloor, P. (2019). In the shades of the uncanny valley: An experimental study of human–chatbot interaction. *Future Generation Computer Systems*, 92, 539-548.
 - [12]. Rosruen, N., & Samanchuen, T. (2018, December). Chatbot utilization for medical consultant system. In *2018 3rd Technology Innovation Management and Engineering Science International Conference (TIMES-iCON)* (pp. 1-5). IEEE.
 - [13]. Turing, A. M. (2009). Computing machinery and intelligence. *Parsing the Turing test*. ed: Springer Dordrecht.
 - [14]. Marietto, M.d. G. B., de Aguiar, R. V., Barbosa, G.d. O., Botelho, W. T., Pimentel, E., Franca, R.d. S., & da Silva, V. L. (2013). *Artificial intelligence Markup Language: A brief tutorial*. arXiv preprint arXiv:1307.3091.
 - [15]. Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1), 36-45.
 - [16]. Wallace, R. S. (2009). *The Anatomy of Alice. Parsing the Turing test*. Springer.
 - [17]. Molnár, G., & Szüts, Z. (2018, September). The role of chatbots in formal education. In *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)* (pp. 000197-000202). IEEE.
 - [18]. Reis, A., Paulino, D., Paredes, H., Barroso, I., Monteiro, M. J., Rodrigues, V., & Barroso, J. (2018, June). Using Intelligent Personal Assistants to Assist the Elderlies: An evaluation of Amazon Alexa, Google Assistant, Microsoft Cortana, and Apple Siri. In *2018 2nd International Conference on Technology and Innovation in Sports, Health and Wellbeing (TISHW)* (pp. 1-5). IEEE.
 - [19]. Ondáš, S., Pleva, M., & Hládek, D. (2019, November). How chatbots can be involved in the education process. In *2019 17th International Conference on Emerging Learning Technologies and Applications (ICETA)* (pp. 575-580). IEEE.
 - [20]. Akcora, D. E., Belli, A., Berardi, M., Casola, S., Di Blas, N., Falletta, S., ... & Vannella, F. (2018). Conversational support for education. In *Artificial Intelligence in Education: 19th International Conference, AIED 2018, London, UK, June 27–30, 2018, Proceedings, Part II 19* (pp. 14-19). Springer International Publishing.
 - [21]. Chen, L., Chen, P., & Lin, Z. (2020). Artificial intelligence in education: A review. *Ieee Access*, 8, 75264-75278.
 - [22]. Lee, J. J. (2009). Size matters An exploratory comparison of small- and large-class university lecture introductions. *English for specific purposes*, 28(1), 42-57.
 - [23]. Albayrak, N., Özdemir, A., & Zeydan, E. (2018, May). An overview of artificial intelligence-based chatbots and an example chatbot application. In *2018 26th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4).
 - [24]. Daniel, F., Matera, M., Zaccaria, V., & Dell'Orto, A. (2018, May). Toward truly personal chatbots: on the development of custom conversational assistants. In *Proceedings of the 1st International Workshop on Software Engineering for Cognitive Services* (pp. 31-36)
 - [25]. Knill, O., Carlsson, J., Chi, A., & Lezama, M. (2004). An artificial intelligence experiment in college math education. *Preprint available at <http://www.math.harvard.edu/~knill/preprints/Sofia.pdf>*.
 - [26]. Suciu, G., Pasat, A., Uşurelu, T., & Popovici, E. C. (2019). Social media cloud contact center using chatbots. In *Future Access Enablers for Ubiquitous and Intelligent Infrastructures: 4th EAI International Conference, FABULOUS 2019, Sofia, Bulgaria, March 28-29, 2019, Proceedings 283* (pp. 437-442). Springer International Publishing.
 - [27]. Hussain, S., & Athula, G. (2018, May). Extending a conventional chatbot knowledge base to external knowledge sources and introducing user-based sessions for diabetes education. In *2018, the 32nd international conference on advanced information networking and applications workshops (WAINA)* (pp. 698-703). IEEE.
 - [28]. Pham, X. L., Pham, T., Nguyen, Q. M., Nguyen, T. H., & Cao, T. T. H. (2018, November). Chatbot as an intelligent personal assistant for mobile language learning. In *Proceedings of the 2018 2nd International Conference on Education and E-Learning* (pp. 16-21).
 - [29]. Zhao, J., Song, T., & Sun, Y. (2020). APIHelper: helping junior Android programmers learn API usage. *IAENG International Journal of Computer Science*, 47(1), 92-97.
 - [30]. Zaw, K. K., Zaw, W., Funabiki, N., & Kao, W. C. (2019). An informative test code approach in code writing problem for three object-oriented programming concepts in Java

programming learning assistant system. *IAENG International Journal of Computer Science*, 46(3), 445-453.

[31]. Holmes, W., Bialik, M., & Fadel, C. (2023). Artificial intelligence in education. Globethics Publications.

How To Cite:

Migisha Jim ,“ Design and Development of An Automated Conversational Chatbot Using Natural Language Processing and NLTK “, *International Journal of Computational Sciences and Engineering Research (IJCSEr)* , Volume 1, Issue 1 , pp 20-33, January, 2026.
CrossRef DOI: <https://doi.org/10.63328/ijcser-v1r1p4>

Declaration

Conflicts of Interest: The authors declare no conflict of interest.

Author Contribution: All authors wrote the main manuscript text and also consent to the submission.

Ethical approval: Not applicable.

Consent to Participate: All authors consent to participate.

Funding: Not applicable, and No funding was received

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Personal Statement: We declare with our best of knowledge that this research work is purely Original Work and No third party material used in this article drafting. If any such kind material found in further online publication, we are responsible only for any judicial and copyright issues.

Generative AI Statement: The authors declare that generative AI was not used in the preparation or creation of this manuscript. The alternative text (alt text) accompanying the figures in this article was generated by Frontiers with the assistance of artificial intelligence. Reasonable efforts have been made to ensure the accuracy and appropriateness of the generated alt text, including review by the authors wherever possible. If you identify any inaccuracies or issues, please contact the editorial team.

Publisher's Note: The statements, opinions, and claims presented in this article are exclusively those of the authors and do not necessarily reflect the views of their affiliated institutions, the publisher, editors, or reviewers. Any products discussed or evaluated, as well as any claims made by their manufacturers, are not warranted, approved, or endorsed by the publisher.

Acknowledgements

We thank everyone who inspired our work.