Bahria University,

Karachi Campus

## LAB EXPERIMENT NO.

11

## LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| **1** | Develop a chatbot using Python and NLTK that can handle and respond to user queries by understanding their context, even if the queries do not exactly match the entries in the chatbot's knowledge base. |
|  |  |

Submitted On:

15 May 2024

(Date: DD/MM/YY)

**TASK :** Develop a chatbot using Python and NLTK that can handle and respond to user queries by understanding their context, even if the queries do not exactly match the entries in the chatbot's knowledge base.

**SOLUTION:**

import nltk

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

import numpy as np

def read\_knowledge\_base(filename):

    filename = "chatbot.txt"

    with open(filename, 'r',encoding='latin-1') as file:

        knowledge\_base = {}

        for line in file:

            parts = line.strip().split()

            if len(parts) >= 2:

                question = parts[0]

                answer = ' '.join(parts[1:])

                knowledge\_base[question.strip()] = answer.strip()

    return knowledge\_base

def preprocess(text):

    tokens = word\_tokenize(text)

    tokens = [word.lower() for word in tokens if word.isalpha()]

    stop\_words = set(stopwords.words("english"))

    tokens = [word for word in tokens if word not in stop\_words]

    lemmatizer = WordNetLemmatizer()

    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    return " ".join(tokens)

tfidf\_vectorizer = TfidfVectorizer()

knowledge\_base = read\_knowledge\_base("knowledge\_base.txt")

preprocessed\_knowledge\_base = [preprocess(question) for question in knowledge\_base.keys()]

knowledge\_base\_vectors = tfidf\_vectorizer.fit\_transform(preprocessed\_knowledge\_base)

def get\_response(user\_input):

    user\_input = preprocess(user\_input)

    user\_vector = tfidf\_vectorizer.transform([user\_input])

    similarities = cosine\_similarity(user\_vector, knowledge\_base\_vectors)

    max\_similarity\_index = np.argmax(similarities)

    max\_similarity = similarities[0, max\_similarity\_index]

    if max\_similarity > 0:

        response = list(knowledge\_base.values())[max\_similarity\_index]

    else:

        response = "Sorry, I'm not sure how to respond to that."

    return response

while True:

    user\_input = input("You: ")

    if user\_input.lower() == 'quit':

        print("Goodbye!")

        break

    else:

        response = get\_response(user\_input)

        print("Chatbot:", response)



