AIDS II

Experiment 2

Name: Sharvari Kishor More

Class: D20B

Roll no.: 35

Aim:

To build a Cognitive text based application to understand context for a Customer service application/ Insurance/ Healthcare Application/ Smarter Cities/ Government

Theory:

The objective of this experiment is to develop a basic cognitive text-based healthcare application that can assist users with medical-related queries. The application will understand user input, identify keywords, and provide appropriate healthcare information or responses. The primary goal is to demonstrate the fundamental principles of building a text-based chatbot for healthcare assistance using Natural Language Processing (NLP).

Theoretical Background:

Natural Language Processing (NLP): NLP is a field of Artificial Intelligence that deals with the interaction between computers and human (natural) languages. It enables machines to understand, interpret, and generate human language. In this experiment, we use NLP techniques to process user queries, extract meaningful information, and generate appropriate healthcare responses.

Keyword-Based Text Analysis: Keyword-based text analysis involves detecting specific words or phrases in user input to infer intent. This approach is useful for mapping symptoms, appointments, or insurance-related queries to predefined categories. It forms the core logic behind intent classification in the chatbot.

Libraries Used

spaCy: A modern NLP library used for advanced text processing tasks such as tokenization, lemmatization, and stop word removal.

NLTK (Natural Language Toolkit): A classic Python toolkit for text analysis and tokenization. It is useful for intent classification using keyword matching.

Steps:

Data Preparation: Define a set of sample healthcare queries (e.g., symptoms, appointments, insurance) and their corresponding responses. These samples are used to simulate real-time user interaction.

Text Preprocessing: Clean and process the input using spaCy. This includes tokenization, lemmatization, and removal of stop words and punctuation to normalize the text.

Keyword Matching: Implement a basic keyword matching logic using NLTK to classify user intent based on the presence of relevant medical terms.

User Interaction: The chatbot accepts user queries through console input or script and responds with a suitable message depending on the recognized intent.

Expected Outcome:

Understand and respond to health-related queries such as symptoms, appointments, insurance, and medications.

Use spaCy for text preprocessing and NLTK for classifying intents based on keywords.

Demonstrate a foundational understanding of how cognitive chatbots can be built for healthcare assistance using NLP.

Code

1. Installing the necessary python libraries

```
!pip install nltk spacy
!python -m nltk.downloader punkt stopwords
!python -m spacy download en_core_web_sm
```

```
Requirement already satisfied: spacy in /usr/local/lib/python3.11/dist-packages (3.8.7)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.11/dist-packages (from spacy) (
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.11)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.0.10
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.11/dist-packages (from spacy) (8.3.6)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.5.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.16.0)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.32
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (25.0)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.5.
Requirement already satisfied: language-data>=1.2 in /usr/local/lib/python3.11/dist-packages (from langcodes<4.0.0,>
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->s
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from weasel<0.
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->spacy) (3.0.
Requirement already satisfied: marisa-trie>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from language-data>=1.
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0-
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open<8.0.0,>=5.2.1->weas
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->ri
<frozen runpy>:128: RuntimeWarning: 'nltk.downloader' found in sys.modules after import of package 'nltk', but prior
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 94.1 MB/s eta 0:00:00
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

2. Importing the libraries

```
import nltk
import spacy
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

3. Download required resources

```
nltk.download('punkt')
nltk.download('stopwords')
```

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

4. Load spaCy English model

```
nlp = spacy.load("en_core_web_sm")
```

5. Stop words

```
stop_words = set(stopwords.words('english'))
```

6. Define intents and possible responses

```
responses = {
    "greeting": "Hello! How can I assist you with your health today?",
    "symptom": "Please describe your symptoms in detail so I can help you better.",
    "appointment": "Sure, I can help you schedule an appointment. Please provide the preferred date and time.",
    "insurance": "Yes, I can assist you with insurance information. Are you asking about coverage, claims, or plans?",
    "medication": "Please specify the medication name you're asking about so I can give accurate details.",
    "emergency": "If this is a medical emergency, please contact emergency services or visit the nearest hospital immediat
    "goodbye": "Take care! Wishing you good health.",
    "unknown": "I'm sorry, I didn't understand that. Could you please rephrase?"
}
```

7. Keywords to classify intent

```
intent_keywords = {
    "greeting": ["hello", "hi", "good morning", "hey"],
    "symptom": ["fever", "cold", "pain", "headache", "cough", "sick", "vomit"],
    "appointment": ["appointment", "schedule", "book", "visit", "consult"],
    "insurance": ["insurance", "policy", "claim", "coverage"],
    "medication": ["medicine", "medication", "dose", "tablet", "prescription"],
    "emergency": ["emergency", "urgent", "immediately", "accident", "bleeding"],
    "goodbye": ["bye", "thanks", "goodbye", "see you"]
}
```

8. Pre-processing the user input to make it case insensitive.

```
def preprocess_text(text):
    doc = nlp(text.lower())
    return [token.lemma_ for token in doc if not token.is_stop and token.is_alpha]
```

```
def classify_intent(user_input):
    tokens = preprocess_text(user_input)
    for intent, keywords in intent_keywords.items():
        if any(word in tokens for word in keywords):
            return intent
    return "unknown"
```

9. Generating responses for the user input

```
def generate_response(user_input):
    intent = classify_intent(user_input)
    response = responses.get(intent, responses["unknown"])
    print(f"User: {user_input}")
    print(f"Bot: {response}\n")
```

10. Demo run (OUTPUT)

```


if __name__ == "__main__":
    queries = [
        "Hi there!",
        "I have a fever and headache.",
        "Can I book an appointment for tomorrow?",
        "Tell me about my insurance policy.",
        "What's the dose for this medicine?",
        "This is an emergency!",
        "Thanks, bye!"
    ]

    for query in queries:
        generate_response(query)
```

```
User: Hi there!
Bot: Hello! How can I assist you with your health today?

User: I have a fever and headache.
Bot: Please describe your symptoms in detail so I can help you better.

User: Can I book an appointment for tomorrow?
Bot: Sure, I can help you schedule an appointment. Please provide the preferred date and time.

User: Tell me about my insurance policy.
Bot: Yes, I can assist you with insurance information. Are you asking about coverage, claims, or plans?

User: What's the dose for this medicine?
Bot: Please specify the medication name you're asking about so I can give accurate details.

User: This is an emergency!
Bot: If this is a medical emergency, please contact emergency services or visit the nearest hospital immediately.

User: Thanks, bye!
Bot: Take care! Wishing you good health.
```

Conclusion:

In this experiment, we successfully developed a cognitive text-based healthcare application capable of understanding and responding to user queries using Natural Language Processing techniques. By integrating spaCy for preprocessing and NLTK for intent classification through keyword matching, the system was able to identify user intents such as greetings, symptoms, appointment bookings, insurance inquiries, and emergencies.

This approach demonstrates the foundational principles behind building intelligent chatbots for the healthcare domain. Although simple in its current form, the application lays the groundwork for more advanced features such as Named Entity Recognition (NER), sentiment analysis, contextual conversation flow, and integration with hospital databases.

Overall, this project showcases how NLP techniques can be applied to improve patient interaction and accessibility in healthcare systems and can be further expanded for real-world deployment in clinical or telehealth environments