AIDS 2 Lab Experiment - 2

Name: Deepa Behrani
Class: D20B
Roll no: 05

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

THEORY:

The objective of this experiment is to develop a basic cognitive text-based application that can assist users with railway ticket booking-related queries. The application will be able to understand user input, identify keywords, and provide relevant information or responses. The primary goal is to demonstrate the basic principles of building a text-based chatbot for railway ticket booking help.

Theoretical Background

1. **Natural Language Processing (NLP)**: NLP is a field of artificial intelligence that focuses on the interaction between computers and human language. It includes techniques for text analysis, sentiment analysis, and language understanding, which are essential for building chatbots.
2. **Keyword-Based Text Analysis**: Keyword-based text analysis involves searching for specific words or phrases in a text to extract relevant information. In this experiment, we will use keyword matching to identify user queries related to railway ticket booking.

Libraries Used

- **spaCy**: We will use the spaCy library for advanced text processing, including tokenization, lemmatization, and stop word removal.
- **nltk**: NLTK (Natural Language Toolkit) is a popular Python library for natural language processing, offering a wide range of tools and resources for tasks such as text analysis, tokenization, and part-of-speech tagging.

Steps

1. **Data Preparation**: Define a set of sample customer queries and their corresponding railway ticket booking responses. These queries will be used to train and test our chatbot.
2. **Text Preprocessing**: Preprocess the text by tokenizing, lemmatizing, and removing stop words and punctuation using spaCy.
3. **Keyword Matching**: Implement keyword matching to identify relevant queries based on the presence of specific keywords.
4. **User Interaction**: Users will interact with the chatbot by entering queries, and the chatbot will respond based on the identified keywords and similarity scores.

Expected Outcome

The experiment is expected to result in a basic **railway ticket booking assistance chatbot** that can respond to user queries by matching them with predefined keywords and responses. It will demonstrate the fundamental principles of building a cognitive text-based application for railway ticket booking help.

CODE:

```
[3] import spacy
    import nltk
    from nltk.tokenize import word_tokenize
    nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
[4] nlp = spacy.load("en_core_web_sm")
```

```
[5] responses = {
        "book": "To book a ticket, please visit the IRCTC website or use our app.",
        "cancel": "To cancel a ticket, go to 'My Bookings' and click on 'Cancel'.",
        "schedule": "You can check the train schedule on the IRCTC portal.",
        "delay": "You can check train delays using the train number.",
        "platform": "Platform information is updated 30 minutes before the train departure."
    }
```

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

```
[7] def classify_intent(tokens):
        for word in tokens:
            if word in responses:
                return word
        return "unknown"
```

```
[8] def generate_response(user_input):
        tokens = preprocess_text(user_input)
        intent = classify_intent(tokens)
        if intent != "unknown":
            return responses[intent]
        else:
            return "Sorry, I didn't understand that. Can you rephrase?"
```

```python
# Example Queries
queries = [
    "Can I book a train ticket?",
    "What platform will my train arrive on?",
    "Is the train delayed?",
    "I want to cancel my booking.",
    "Show me the schedule of trains to Mumbai."
]

for query in queries:
    print(f"User: {query}")
    print("Bot:", generate_response(query))
    print()
```

OUTPUT:

```
User: Can I book a train ticket?
Bot: To book a ticket, please visit the IRCTC website or use our app.

User: What platform will my train arrive on?
Bot: Platform information is updated 30 minutes before the train departure.

User: Is the train delayed?
Bot: You can check train delays using the train number.

User: I want to cancel my booking.
Bot: To cancel a ticket, go to 'My Bookings' and click on 'Cancel'.

User: Show me the schedule of trains to Mumbai.
Bot: You can check the train schedule on the IRCTC portal.
```

Explanation

1. **Install Libraries**: Install `spaCy` and `nltk` and download the necessary language models and data.
2. **Import Libraries**: Import `spaCy` for NLP tasks and `nltk` for tokenization and intent classification.
3. **Load spaCy Model**: Load the `en_core_web_sm` spaCy model.
4. **Data Preparation**: Define a set of sample customer queries and their corresponding responses.
5. **Preprocess Text**: Define a function `preprocess_text` to preprocess the user input by tokenizing, lemmatizing, and removing stop words and punctuation using `spaCy`.
6. **Intent Classification**: Define a function `classify_intent` that uses `nltk` to classify the intent of the user input based on keywords.

7. **Generate Response**: Define a function `generate_response` that generates a response based on the classified intent and matches it with the predefined responses. 8. **Example Usage**: Test the functions with example questions related to railway ticket booking to see how the application processes different inputs and generates responses.

Conclusion:

This approach leverages both `spaCy` and `nltk` to provide a more sophisticated and cognitive response generation system. The `classify_intent` function classifies the user's intent, while `spaCy` preprocesses the text for more accurate keyword matching. This setup can be further expanded to handle more complex scenarios and integrate additional NLP techniques.