Name: Atif Ansari **Roll no:** 04 **Class:** D20B

AIDS 2 EXP 4

Aim:

To build a Cognitive Text-Based Application to Understand Context for Insurance Help.

Theory:

The goal of this experiment is to develop a basic cognitive text-based application that can assist users with insurance-related queries. This application will be able to understand user input, identify keywords, and provide relevant information or responses based on predefined answers.

Objective:

The primary objective is to build a chatbot that can understand the user's input, identify keywords, and respond appropriately based on insurance-related information. This chatbot will demonstrate how text processing and natural language understanding can be implemented in insurance assistance.

Theoretical Background:

1. **Natural Language Processing (NLP):**
    - NLP is a branch of Artificial Intelligence that enables computers to understand, interpret, and generate human language. It involves tasks like tokenization, part-of-speech tagging, and named entity recognition, which help in understanding the context of a text.
2. **Keyword-Based Text Analysis:**
    - Keyword-based text analysis involves searching for specific words or phrases in a user query to extract relevant information. By detecting the keywords in a query, we can map it to predefined responses.
    - This approach is effective for building simple chatbots where the goal is to match user queries with a set of predefined answers.

Libraries Used:

- **spaCy**: A powerful Python library for natural language processing (NLP), used for tokenization, lemmatization, stop-word removal, and other text processing tasks.

- **nltk**: The Natural Language Toolkit provides easy-to-use tools for text analysis, including classification, tokenization, and sentiment analysis.

Steps:

1. **Data Preparation:**
    - Define a set of sample customer queries with relevant insurance responses. These queries will help train and test the chatbot.

2. **Text Preprocessing:**
    - Preprocess the text by tokenizing, lemmatizing, and removing stop words and punctuation using spaCy to prepare the text for analysis.

3. **Keyword Matching:**
    - Implement keyword matching to identify and respond to user queries based on keywords extracted from predefined questions.

4. **User Interaction:**
    - Set up an interactive session where users can input their queries, and the chatbot will respond based on matching keywords or provide a default response.

Code:

```
import nltk
import spacy

# Load spaCy model for text processing
nlp = spacy.load("en_core_web_sm")

# Sample customer queries and responses with keywords
queries_and_responses = [
    ("What is my policy coverage?", "Your policy covers damage to your
vehicle caused by accidents."),
```

```
    ("How can I file a claim?", "You can file a claim online through our
website."),
    ("What is the premium for auto insurance?", "The premium for your auto
insurance is ₹500 per month."),
    ("Are rental car expenses covered?", "Yes, rental car expenses are
covered under your policy."),
    ("How can I update my contact information?", "You can update your
contact information in your online account."),
    ("What is the grace period for premium payment?", "The grace period
for premium payment is 30 days."),
    ("Can I add a new driver to my policy?", "Yes, you can add a new
driver to your policy. Contact our support for details."),
    ("How do I cancel my policy?", "To cancel your policy, please contact
our customer service department."),
    ("Do you offer any discounts?", "We offer various discounts based on
your driving history and more."),
]

# Default responses for various scenarios
default_responses = {
    "greeting": "Hello! How can I assist you today?",
    "farewell": "Goodbye! Have a great day!",
    "default": "I'm sorry, I didn't understand your question. Please ask
something else.",
}

# Function to preprocess the text (tokenize, remove stop words, etc.)
def preprocess_text(text):
    doc = nlp(text.lower())  # Convert to lowercase and process with spaCy
    processed_text = " ".join([token.lemma_ for token in doc if not
token.is_stop and not token.is_punct])
    return processed_text

# Function to classify user queries
def classify_query(user_query):
    user_query = preprocess_text(user_query)  # Preprocess user input

    # Check for greetings and farewells
    if any(greeting in user_query for greeting in ["hi", "hello", "hey"]):
        return "greeting"
```

```
    elif any(farewell in user_query for farewell in ["bye", "goodbye",
"see you"]):
        return "farewell"

    # Check for keyword matches in queries
    for keywords, response in queries_and_responses:
        processed_keywords = preprocess_text(keywords)
        if any(keyword in user_query for keyword in
processed_keywords.split()):
            return response

    return default_responses["default"]  # If no match found, return
default response

# Interactive user interface
while True:
    user_query = input("You: ")  # Get user input
    if user_query.lower() in ['exit', 'quit', 'end']:
        print("Chatbot: Goodbye! Have a great day!")
        break

    # Classify the user's query
    response = classify_query(user_query)

    # Provide the appropriate response
    print("Chatbot:", response)
```

Code Explanation:

1. **Preprocessing User Input:**
    - **`preprocess_text()`**: This function preprocesses the user's query by converting it to lowercase, removing stop words (like "is", "and", etc.), and lemmatizing the words (reducing words to their base forms, e.g., "running" to "run").
2. **Classifying Queries:**
    - **`classify_query()`**: This function first processes the user's query and then checks whether the query contains greetings, farewells, or keywords related to insurance (using predefined queries and responses). It matches the keywords

after preprocessing the text.

3. **User Interaction:**
    - The chatbot asks for the user's input, processes it, and provides a relevant response based on the keywords found in the user's query.
4. **Response Handling:**
    - The chatbot responds to greetings, farewells, or questions related to insurance based on keyword matching. If no keyword matches, it returns a default response.

Output:

```
You: Hi
Chatbot: greeting
You: I want to know about my insurance policy.
Chatbot: Your policy covers damage to your vehicle caused by accidents.
You: How can I update my contact information?
Chatbot: You can update your contact information in your online account.
You: I want to know about premium
Chatbot: The premium for your auto insurance is ₹500 per month.
You: I want to know about my claim
Chatbot: You can file a claim online through our website.
You: I want to know about discounts
Chatbot: We offer various discounts based on your driving history and more.
You: Goodbye
Chatbot: farewell
```

Conclusion:

In this experiment, we successfully developed a basic cognitive text-based application designed to help users with insurance-related queries. By using Natural Language Processing (NLP) and keyword-based text analysis, the chatbot can understand user queries and respond with relevant information about insurance policies, premium payments, and more.

This experiment demonstrates how basic cognitive text-based applications can be built using NLP techniques such as tokenization, lemmatization, and keyword matching. These applications can be extended with more sophisticated features like intent recognition, entity extraction, and machine learning models for more dynamic interactions in real-world scenarios.