

Name: **GANESH SANJAY PANDHRE**

Roll Nos : **41** | Div: **D20B**

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

✓ Experiment No: 2

Theory:

What is a Cognitive Text-Based Application?

A cognitive text-based application is a smart software system that can understand human language and give meaningful responses. It works like a chatbot, where users type messages, and the system replies based on what the user means. These applications use **Artificial Intelligence (AI)** techniques like:

- **Natural Language Processing (NLP)** – To understand text and language.
 - **Machine Learning (ML)** – To learn from data.
 - **Cognitive Computing** – To mimic how humans think and respond.
-

What is Cognitive Computing?

Cognitive computing means creating computer systems that behave like the human brain. These systems:

- Understand and analyze natural language.
- Learn from experience and data.
- Help in decision-making by giving useful responses.

They are used in real-life for solving complex problems like health diagnosis, fraud detection, city traffic management, etc.

What is Natural Language Processing (NLP)?

NLP is a part of AI that helps computers understand and work with human language (text or speech). In this experiment, NLP is used to clean and understand the user's input.

Common NLP Tasks:

- **Tokenization**: Breaking text into words or sentences.
- **Stopword Removal**: Removing common words like "the", "is", etc.
- **Lemmatization/Stemming**: Converting words to their root form (e.g., "running" → "run").
- **Entity Recognition**: Finding important things like names, dates, numbers in text.

We use **Python libraries like spaCy and NLTK** to do these NLP tasks.

What is Keyword-Based Analysis?

In basic cognitive applications, we use **keyword matching**. It means the chatbot checks if important words (keywords) are present in the user's question. Based on these keywords, the system replies with a matching answer.

Example:

- User: "What time is garbage collected?"

- Keyword detected: "garbage"
- Response: "Garbage is collected daily at 8 AM."

Technologies & Libraries Used:

Library	Use
spaCy	For advanced NLP tasks like tokenization, lemmatization, etc.
NLTK	For basic text processing and word analysis.
Python	Programming language used to build the chatbot.

Real-Life Use Cases of Cognitive Applications:

Sector	Use Case
Healthcare	Chatbots for patient help, symptom checkers, report analysis.
Insurance	Claim support, policy queries, fraud detection.
Customer Service	Virtual assistants to solve user queries instantly.
Smarter Cities	Assistants for electricity, water, parking, pollution alerts.
Government	Help with public services, emergency info, and citizen interaction.


Steps to Build the Application:

1. Data Preparation
 - Create a list of user questions and expected responses for a specific domain.
2. Text Preprocessing
 - Clean the user input using NLP (remove stopwords, tokenize, etc.).
3. Keyword Matching
 - Check the input text for important keywords.
4. Response Generation
 - If keywords are matched, reply with the related response.
5. User Interaction
 - Keep taking queries from the user until they say "bye".

```
# Step 1: Install & Import Required Libraries
!pip install spacy --quiet
import nltk
import spacy

# Download resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')

# Load SpaCy English model
import en_core_web_sm
nlp = en_core_web_sm.load()
```



```
[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!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt_tab.zip.
```

```
# Step 2: Define Predefined Queries and Responses
# Common apartment-related queries and responses
queries_and_responses = [
    ("water problem", "Water supply will be restored by 4 PM. Maintenance is in progress."),
    ("electricity issue", "There is a power cut scheduled from 2 PM to 3 PM."),
    ("visitor entry", "Visitors are allowed only between 9 AM and 9 PM. Please inform the security desk."),
    ("garbage collection", "Garbage is collected daily at 8 AM. Please keep bins outside before that."),
    ("parking", "You have been assigned Parking Slot A-12."),
    ("maintenance request", "You can raise a maintenance request via the Resident App."),
    ("society meeting", "The next society meeting is on Saturday at 5 PM in the clubhouse."),
    ("package delivery", "Your package is at the security desk. Kindly collect it."),
    ("gym timing", "The gym is open from 6 AM to 10 PM."),
    ("swimming pool", "The swimming pool is closed on Mondays for cleaning.")
]
```

```
# Step 3: Define Default Responses and Tokenization Logic
default_responses = {
    "greeting": "Hello! I'm your Apartment Assistant. How can I help you today?",
    "farewell": "Goodbye! Have a peaceful day in your apartment!",
    "default": "Sorry, I didn't understand that. Try asking about water, electricity, or parking."
}
```

```
# Step 4: Classify User Input & Match Response
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

stop_words = set(stopwords.words("english"))

def classify_query(user_query):
    user_query = user_query.lower()
    user_tokens = set(word_tokenize(user_query))
    user_tokens = [word for word in user_tokens if word not in stop_words]

    if any(greet in user_query for greet in ["hi", "hello", "hey"]):
        return "greeting"
    elif any(farewell in user_query for farewell in ["bye", "goodbye", "see you"]):
        return "farewell"

    for keywords, response in queries_and_responses:
        keyword_tokens = set(keywords.split())
        if set(user_tokens).intersection(keyword_tokens):
            return response

    return "default"
```

```
# Step 5: Run the Chatbot Loop
def apartment_chatbot():
    print("Apartment Assistant: Hello! Type your query or say 'bye' to exit.")
    while True:
        user_input = input("You: ")
        result = classify_query(user_input)

        if result == "greeting":
            print("Apartment Assistant:", default_responses["greeting"])
        elif result == "farewell":
            print("Apartment Assistant:", default_responses["farewell"])
            break
        elif result == "default":
            print("Apartment Assistant:", default_responses["default"])
        else:
            print("Apartment Assistant:", result)
```

```
# Run the chatbot
apartment_chatbot()
```

↩ Apartment Assistant: Hello! Type your query or say 'bye' to exit.
You: Hello
Apartment Assistant: Hello! I'm your Apartment Assistant. How can I help you today?
You: When will garbage be collected?
Apartment Assistant: Garbage is collected daily at 8 AM. Please keep bins outside before that.
You: What are the gym timings?
Apartment Assistant: The gym is open from 6 AM to 10 PM.
You: Did my courier arrive?
Apartment Assistant: Sorry, I didn't understand that. Try asking about water, electricity, or parking.
You: Bye

Conclusion:

In this experiment, we successfully developed a basic cognitive text-based chatbot using Python, spaCy, and NLTK. The chatbot was able to understand user queries using natural language processing and respond based on keyword matching. This approach helps in automating user interactions and can be applied in real-world scenarios like customer service, healthcare, insurance, and smart city systems. Overall, we learned how cognitive computing and NLP techniques can be used to build intelligent and helpful applications.
