Name : Naikwadi Yash Shivdas

AI&DS II Experiment 02

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

THEORY:

What is a Cognitive Text-Based Application?

A cognitive text-based application is an intelligent software system that can understand and respond to human language in a meaningful way. These applications use Natural Language Processing (NLP), Machine Learning (ML), and Artificial Intelligence (AI) techniques to process user input and provide accurate, human-like responses. The main goal is to create applications that can think and respond like humans when dealing with text-based conversations.

What is Cognitive Computing?

Cognitive computing is a branch of Artificial Intelligence (AI) that tries to mimic how humans think, learn, and make decisions. It is a technology that simulates human thought processes in a computerized model. The key aspects of cognitive computing include:

- Understanding language (using Natural Language Processing)
- Learning from experience (Machine Learning)
- Making decisions or predictions based on data
- Recognizing patterns in large amounts of information
- Providing helpful responses just like a human would in real-world scenarios

These systems are able to analyze large amounts of data, recognize patterns, and provide helpful responses that feel natural and intelligent.

What is Natural Language Processing (NLP)?

Natural Language Processing (NLP) is a field of Artificial Intelligence that allows computers to understand, interpret, and respond to human language. NLP helps computers make sense of human language in a way that is both meaningful and useful. The main components of NLP include:

- Tokenization – Breaking text into individual words or sentences
- Stopword Removal – Ignoring common words like "the", "is", "and", etc.
- Lemmatization/Stemming – Converting words to their root forms (e.g., "running" → "run")
- Entity Recognition – Identifying important information like names, places, numbers, dates, etc.
- Sentiment Analysis – Understanding emotions and opinions in text

Applications of Cognitive Text-Based Systems in Different Sectors

Cognitive text-based applications have become valuable across various industries, helping solve complex problems and improving customer service:

Sector	Use Cases	Benefits
Healthcare	Disease diagnosis support, patient monitoring, medical query assistance, appointment scheduling	Faster diagnosis, 24/7 patient support, reduced workload for doctors

Insurance	Claim processing, fraud detection, policy information, premium assistance, customer support	Quick claim resolution, automated fraud detection, better customer experience
Banking & Finance	Account queries, transaction support, loan assistance, fraud detection, investment advice	Secure transactions, instant customer support, risk management
Retail & E-commerce	Product recommendations, order tracking, customer support, inventory management	Personalized shopping, efficient customer service, better sales
Education	Student queries, course information, assignment help, personalized learning	24/7 student support, customized learning experience
Government	Citizen services, document processing, query resolution, emergency services	Efficient public service, reduced paperwork, quick response
Smart Cities	Traffic management, utility services, citizen complaints, emergency response	Better city management, improved quality of life
Transportation	Ticket booking, route information, schedule queries, travel assistance	Convenient travel planning, real-time information

Key Technologies and Libraries Used
To build cognitive text-based applications, we use several important technologies and libraries:

Technology/Library	Purpose	Key Features
Python	Programming language	Easy to learn, extensive libraries, good for AI/ML projects
spaCy	Advanced NLP processing	Tokenization, lemmatization, entity recognition, stopword removal
NLTK	Natural Language Toolkit	Basic NLP tasks, text processing utilities, language resources
Machine Learning	Pattern recognition and learning	Classification, prediction, data analysis
TF-IDF Vectorization	Text to numerical conversion	Converts text into numbers that computers can understand

How Cognitive Text-Based Applications Work
The working process of a cognitive text-based application involves several important steps:
1. Data Preparation
- Collect common user queries and their expected responses
- Organize data according to the specific domain (healthcare, insurance, etc.)
- Create a knowledge base of question-answer pairs

2. Text Preprocessing
   - Clean user input by converting to lowercase
   - Remove unnecessary words (stopwords) like "the", "is", "and"
   - Break text into individual words (tokenization)
   - Convert words to their root forms (lemmatization)
3. Keyword Matching and Analysis
   - Identify important keywords in user input
   - Match keywords with predefined categories
   - Calculate similarity between user query and stored responses
4. Intent Classification
   - Determine what the user wants (intent)
   - Categorize queries (e.g., greeting, question, complaint, request)
   - Use machine learning to improve classification accuracy
5. Response Generation
   - Select the most appropriate response based on classification
   - Provide contextual and relevant information
   - Handle unknown queries with default responses
6. User Interaction
   - Create an interactive interface for users
   - Maintain conversation context
   - Provide continuous service until user exits

Types of Text Analysis Approaches

1. Keyword-Based Analysis

This is the simplest approach where the system looks for specific important words (keywords) in the user's input. Based on these keywords, it provides predefined responses.
Example:
- User asks: "How to save electricity?"
- System detects keywords: "save" and "electricity"
- System responds: "Here are some tips to save electricity..."

2. Machine Learning-Based Analysis

More advanced systems use machine learning algorithms to understand patterns in text and make predictions about what users want.
Features:
- Learns from previous conversations
- Improves accuracy over time
- Can handle complex queries
- Provides more personalized responses

Benefits of Cognitive Text-Based Applications
1. 24/7 Availability – Always available to help users
2. Cost Effective – Reduces need for human customer service agents
3. Quick Response – Instant answers to user queries
4. Consistent Service – Same quality of service every time
5. Scalability – Can handle multiple users simultaneously
6. Language Understanding – Can understand natural human language
7. Learning Capability – Gets better with more data and usage

Challenges and Limitations

1. Understanding Context – Difficulty in understanding complex or ambiguous queries
2. Language Variations – Challenges with different accents, slang, or informal language
3. Emotional Intelligence – Limited ability to understand emotions and feelings
4. Domain Knowledge – Requires extensive training for specific domains
5. Continuous Updates – Needs regular updates to stay current and accurate

Future Scope and Applications

Cognitive text-based applications are continuously evolving and finding new applications:

- Voice Integration – Combining with speech recognition for voice-based interactions
- Multilingual Support – Supporting multiple languages for global reach
- Advanced AI – Integration with more sophisticated AI models
- IoT Integration – Connecting with Internet of Things devices
- Predictive Analytics – Predicting user needs before they ask
- Emotional AI – Better understanding of human emotions and sentiments

---

CODE:

Install and Import Required Libraries

```
!pip install spacy nltk --quiet
!python -m spacy download en_core_web_sm --quiet

import nltk
import spacy
import re
from datetime import datetime
import random

# Download NLTK resources
nltk.download('punkt', quiet=True)
nltk.download('stopwords', quiet=True)
nltk.download('punkt_tab', quiet=True)

# Load spaCy model
nlp = spacy.load("en_core_web_sm")
```

Data Preparation - Food Delivery Queries and Responses

```
# Food delivery related queries and responses
food_delivery_queries = [
    # Order Status Queries
    ("order status tracking delayed", "Your order #FD12345 is currently being
prepared by the restaurant. Expected delivery time is 25-30 minutes from
now."),
    ("where is my food order", "Your order is with our delivery partner Raj
(9876543210) and will reach you in 15 minutes. Track live location in app."),
    ("order taking too long", "We apologize for the delay. Your order is prioritized and will be delivered within 10 minutes with a complimentary dessert."),

    # Menu and Restaurant Queries
    ("restaurant menu prices vegetarian", "We have 500+ vegetarian restaurants nearby. Popular ones: Punjabi Dhaba (₹150-300), South Indian Corner (₹80-200), Pizza Palace (₹250-450)."),
    ("healthy food options salad", "Try our healthy partners: Fresh Bowl (salads ₹120-180), Protein Kitchen (quinoa bowls ₹200-280), Green Bites (smoothies ₹80-150)."),
    ("spicy chicken biryani available", "Yes! Biryani Paradise has spicy chicken biryani (₹220) with extra raita. Hyderabadi Kitchen also serves special spicy version (₹280)."),

    # Payment and Refund Issues
    ("payment failed money deducted", "Don't worry! Your payment of ₹347 will be automatically refunded within 3-5 business days. Order is cancelled, you can reorder now."),
    ("wrong item delivered refund", "Sorry for the confusion! Full refund of ₹425 initiated. Keep the wrong item as goodwill gesture. Reorder with 50% discount code: SORRY50"),
    ("coupon code not working discount", "The code SAVE20 is valid till midnight today. Try NEWUSER30 for 30% off (max ₹100) or WEEKEND25 for 25% off your order."),

    # Delivery and Location Issues
    ("delivery address change update", "Address can be changed within 2 minutes of order placement. Current address: [Old Address]. Please provide new complete address with landmark."),
    ("delivery boy not found location", "Our delivery partner is trying to reach you. Please share your live location or call him at 9876543210. We can also deliver to nearby landmark."),
    ("contactless delivery safe", "Yes! We offer contactless delivery. Food will be left at your doorstep with call notification. All partners wear masks and sanitize regularly."),

    # Restaurant and Quality Issues
    ("food quality bad taste", "We're sorry about the poor food quality. Full refund initiated + ₹100 FoodCash credited to your account. We'll review this restaurant immediately."),
    ("restaurant closed unavailable", "The restaurant is temporarily closed. Try similar options: Tasty Treats (4.2★) or Foodie Paradise (4.5★) with similar cuisine and pricing."),
    ("cold food delivered", "Apologies for cold food! ₹150 refund processed instantly. Next time, try our 'Hot & Fresh Guarantee' restaurants for piping hot delivery."),

    # Account and App Issues
    ("login problem forgot password", "Click 'Forgot Password' on login page.
Enter registered mobile number, you'll get OTP. Or login with Google/Facebook
for quick access."),
    ("app not working crash", "Try these steps: 1) Clear app cache 2) Update
app from Play Store 3) Restart phone 4) Use our website fooddelivery.com as
backup."),
    ("delete account cancel subscription", "Account deletion request noted. All
data will be removed in 7 days. Premium subscription of ₹99/month will be
cancelled immediately."),
]

# Special time-based responses
time_based_responses = {
    "breakfast": "Good morning! Try our breakfast combos: Poha + Tea (₹60), Upma
+ Coffee (₹55), Masala Dosa + Filter Coffee (₹85).",
    "lunch": "Lunch time! Popular choices: Dal Rice + Sabji (₹120), Chicken
Curry + Roti (₹180), Veg Thali (₹150).",
    "dinner": "Dinner specials: Butter Chicken + Naan (₹280), Paneer Tikka +
Roti (₹220), Family Pizza Large (₹450).",
    "late night": "Late night cravings? Try: Maggi (₹40), Sandwich (₹80),
Chinese combo (₹160), or 24/7 restaurants near you!"
}

# Default response categories
default_responses = {
    "greeting": "🍕 Welcome to FoodExpress! I'm your food delivery assistant.
How can I help you today? Ask about orders, restaurants, or any food-related
query!",
    "farewell": "🙏 Thanks for using FoodExpress! Enjoy your meal and have a
great day! Rate us 5⭐ if you're happy with our service!",
    "default": "I didn't understand that specific query. Try asking about:
order status, restaurant menu, payment issues, delivery problems, or food
recommendations."
}
```

Text Preprocessing Functions

```
def preprocess_text(text):
    """Advanced text preprocessing using spaCy"""
    # Convert to lowercase
    text = text.lower()

    # Process with spaCy
    doc = nlp(text)

    # Extract meaningful tokens (excluding stopwords, punctuation, and spaces)
    tokens = []
    for token in doc:
        if not token.is_stop and not token.is_punct and not token.is_space and
len(token.text) > 2:
```

```
            tokens.append(token.lemma_)

    return tokens

def extract_entities(text):
    """Extract named entities from text"""
    doc = nlp(text)
    entities = [(ent.text, ent.label_) for ent in doc.ents]
    return entities

def get_time_context():
    """Determine time of day for contextual responses"""
    current_hour = datetime.now().hour
    if 5 <= current_hour < 11:
        return "breakfast"
    elif 11 <= current_hour < 16:
        return "lunch"
    elif 16 <= current_hour < 22:
        return "dinner"
    else:
        return "late night"
```

Query Classification and Response Generation

```
def classify_food_delivery_query(user_input):
    """Classify user query and generate appropriate response"""

    # Preprocess user input
    user_tokens = set(preprocess_text(user_input))
    original_input = user_input.lower()

    # Check for greetings
    if any(greeting in original_input for greeting in ["hi", "hello", "hey",
"namaste"]):
        return "greeting"

    # Check for farewells
    if any(farewell in original_input for farewell in ["bye", "goodbye",
"thanks", "thank you"]):
        return "farewell"

    # Time-based food suggestions
    time_keywords = ["breakfast", "lunch", "dinner", "late night", "morning",
"afternoon", "evening"]
    if any(keyword in original_input for keyword in time_keywords):
        time_context = get_time_context()
        return time_based_responses[time_context]

    # Find best matching response using keyword overlap
    best_match = None
    best_score = 0

    for keywords, response in food_delivery_queries:
        keyword_tokens = set(keywords.split())

        # Calculate overlap score
        overlap = len(user_tokens.intersection(keyword_tokens))
        if overlap > best_score:
            best_score = overlap
            best_match = response

    # Return best match if good enough, otherwise default
    if best_score >= 1:  # At least 1 keyword match
        return best_match
    else:
        return "default"

def generate_order_id():
    """Generate a random order ID for demo purposes"""
    return f"FD{random.randint(10000, 99999)}"
```

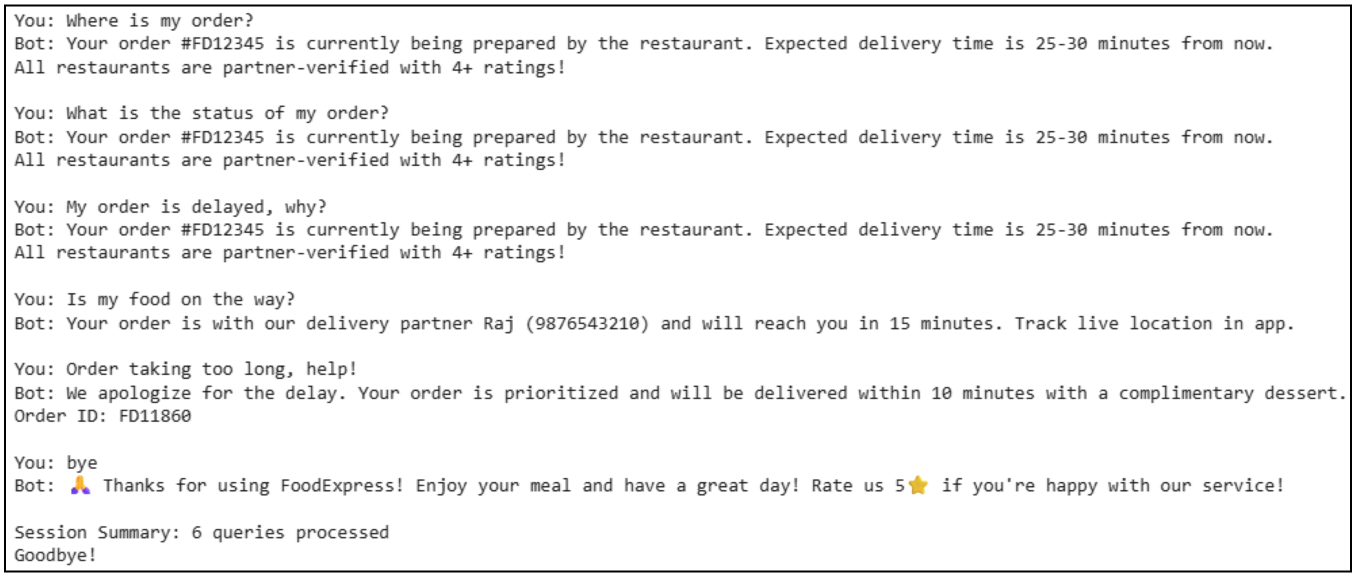
Interactive Chatbot Interface

```
def food_delivery_chatbot():

    conversation_count = 0

    while True:
        try:
            user_input = input("\n🔤 You: ").strip()

            if not user_input:
                print("🤖 Bot: Please type something! I'm here to help with
your food delivery queries.")
                continue

            conversation_count += 1

            # Get response
            response = classify_food_delivery_query(user_input)

            # Handle different response types
            if response == "greeting":
                print(f"🤖 Bot: {default_responses['greeting']}")

            elif response == "farewell":
                print(f"🤖 Bot: {default_responses['farewell']}")
                print(f"\n📊 Session Summary: {conversation_count} queries
processed")
                print("👋 Goodbye!")
                break

            elif response == "default":
                print(f"🤖 Bot: {default_responses['default']}")

                # Extract entities for better help
                entities = extract_entities(user_input)
                if entities:
                    print(f"💡 I detected: {', '.join([ent[0] for ent in
entities[:3]])}")
                    print("   Can you rephrase your question more
specifically?")

            else:
                print(f"🤖 Bot: {response}")

                # Add some dynamic elements
                if "order" in user_input.lower() and "#" not in response:
                    print(f"📋 Order ID: {generate_order_id()}")

                if "refund" in response:
                    print("💳 Refund will be processed to your original payment
method.")

                if "restaurant" in response:
                    print("⭐ All restaurants are partner-verified with 4+
ratings!")

        except KeyboardInterrupt:
            print(f"\n\n🤖 Bot: {default_responses['farewell']}")
            print(f"📊 Session Summary: {conversation_count} queries
processed")
            break
        except Exception as e:
            print(f"🤖 Bot: Sorry, I encountered an error. Please try again!")
            print(f"Error details: {str(e)}")
```

Run the Chatbot

```
food_delivery_chatbot()
```

OUTPUT:



```
You: Where is my order?
Bot: Your order #FD12345 is currently being prepared by the restaurant. Expected delivery time is 25-30 minutes from now.
All restaurants are partner-verified with 4+ ratings!

You: What is the status of my order?
Bot: Your order #FD12345 is currently being prepared by the restaurant. Expected delivery time is 25-30 minutes from now.
All restaurants are partner-verified with 4+ ratings!

You: My order is delayed, why?
Bot: Your order #FD12345 is currently being prepared by the restaurant. Expected delivery time is 25-30 minutes from now.
All restaurants are partner-verified with 4+ ratings!

You: Is my food on the way?
Bot: Your order is with our delivery partner Raj (9876543210) and will reach you in 15 minutes. Track live location in app.

You: Order taking too long, help!
Bot: We apologize for the delay. Your order is prioritized and will be delivered within 10 minutes with a complimentary dessert.
Order ID: FD11860

You: bye
Bot: 🙏 Thanks for using FoodExpress! Enjoy your meal and have a great day! Rate us 5⭐ if you're happy with our service!

Session Summary: 6 queries processed
Goodbye!
```

CONCLUSION:

Cognitive text-based applications represent a significant advancement in how computers interact with humans. By combining Natural Language Processing, Machine Learning, and Artificial Intelligence, these systems can provide intelligent, human-like responses to user queries. They are transforming various sectors by providing efficient, cost-effective, and scalable solutions for customer service and information management. As technology continues to advance, these applications will become even more sophisticated and capable of handling complex human interactions, making them an essential part of our digital future.

39_colab_02