Case Study: Chatbot System with Recommendation System

Problem Statement:

Many businesses struggle to provide timely customer service and personalized product recommendations simultaneously. Customers often experience long wait times for support and receive generic product suggestions that do not match their preferences or previous interactions.

Aim:

To develop an intelligent chatbot system integrated with a recommendation system that can respond to user queries in real time and provide personalized product or content recommendations based on user preferences and interaction history.

Objectives:

- Develop a chatbot capable of understanding and responding to natural language queries.
- Integrate a recommendation system within the chatbot to suggest relevant products or content.
- Use machine learning/NLP techniques to improve chatbot accuracy.
- Enhance user experience through personalization.
- Reduce response time and improve customer satisfaction.

Description:

The system combines **Natural Language Processing (NLP)** for the chatbot and a **collaborative filtering or content-based recommendation algorithm** to suggest items to users. When a user interacts with the chatbot, it understands the intent using NLP and responds accordingly. Based on the user's previous interactions or preferences, it recommends items.

Algorithm:

1. Chatbot Module (NLP-based):

- Preprocess user input (tokenization, stop word removal, stemming).
- Intent Recognition using:
  - Rule-based matching or
  - ML/DL model like BERT or a trained intent classifier
- Generate response based on intent.

2. Recommendation Module:

- *Collaborative Filtering (User-Item Matrix).*
- Use historical data (user-item interactions).
- Apply cosine similarity or matrix factorization to find similar users/items.

Program:

```
import random

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

products = {

   1: "Wireless Mouse",

   2: "Bluetooth Headphones",

   3: "Gaming Keyboard",

   4: "USB-C Charger",

   5: "Laptop Stand"

}

user_history = ["Bluetooth Headphones", "Wireless Mouse"]
```

```
intents = {
    "greet": ["hi", "hello", "hey"],
 "bye": ["bye", "exit", "see you"],
    "recommend": ["recommend", "suggest", "advise"]
}
def get_intent(user_input):
    for intent, keywords in intents.items():
        for word in keywords:
            if word in user_input.lower():
                return intent
    return "unknown"
def recommend_products(user_history):
    corpus = list(products.values())
    corpus.extend(user_history)
    tfidf = TfidfVectorizer().fit_transform(corpus)
    similarity_matrix=cosine_similarity(tfidf[-len(user_history):],
tfidf[:-len(user_history)])
    scores = similarity_matrix.mean(axis=0)
    top_indices = scores.argsort()[-3:][::-1]
    return [list(products.values())[i] for i in top_indices]
print("Chatbot: Hello! How can I assist you today?")
while True:
    user_input = input("You: ")
    intent = get_intent(user_input)
    if intent == "greet":
        print("Chatbot: Hi there! Need any help?")
    elif intent == "recommend":
        recommendations = recommend_products(user_history)
  print("Chatbot: Based on your interests, you might like:")
        for rec in recommendations:
            print(f"- {rec}")
    elif intent == "bye":
        print("Chatbot: Goodbye! Have a nice day.")
        break
    else:
        print("Chatbot: I'm not sure how to help with that.")
```

Output:

```
Chatbot: Hello! How can I assist you today?
You: Can you recommend something?
Chatbot: Based on your interests, you might like:
- Wireless Mouse
- Bluetooth Headphones
- Gaming Keyboard
```

Conclusion:

Integrating a recommendation system with a chatbot enhances the user experience by delivering personalized and efficient interactions. The chatbot handles queries in real-time while the recommendation system ensures users receive relevant suggestions. This system is highly useful in e-commerce, entertainment, and customer service domains.