# AI-Powered Virtual Assistant for E-Commerce

## 1. Introduction

This research paper details the design, development, and implementation of an AI-powered virtual assistant for an e-commerce platform. The virtual assistant is designed to help users discover products, answer frequently asked questions (FAQs), and assist with order tracking. The assistant integrates several advanced technologies, including Natural Language Processing (NLP), Retrieval-Augmented Generation (RAG), and Large Language Models (LLMs).

## 2. Chatbot Platform Selection

The chosen chatbot platform for this project is Flask, a lightweight Python framework that is perfect for building web applications and integrating machine learning models. Flask was selected due to its simplicity, flexibility, and ease of integration with various libraries such as SpaCy for NLP tasks and Scikit-learn for intent classification. Flask provides an easy way to expose the chatbot as an API, making it suitable for building a scalable virtual assistant.

## 3. NLP Intent Classifier and NER Implementation

### 3.1 NLP Intent Classifier

The NLP intent classifier is implemented using a Support Vector Classifier (SVC) model, which is trained using a dataset of sample queries. The model utilizes the TF-IDF vectorization technique to convert the text data into numerical features for training. Once trained, the model can predict the intent of user queries, such as product search, order tracking, and FAQ inquiries. Below is the code used for the intent classifier:

```
from sklearn.svm import SVC
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import make_pipeline

# Sample training data for intent classification
data = {
   'text': [
      'I am looking for a wireless headphone under $100.',
      'Where is my order #12345?',
      'What is your return policy?',
        'Show me some product recommendations.',
        'What is the status of my order?'
    ],
    'intent': [
        'product_search',
        'order_tracking',
        'faq_inquiry',
        'product_search',
        'order_tracking'
    ]
}

# Create and train the model
vectorizer = TfidfVectorizer()
model = make_pipeline(TfidfVectorizer(), SVC(kernel='linear'))
X_train = data['text']
y_train = data['intent']
model.fit(X_train, y_train)

# Function to predict intent
def predict_intent(text):
    return model.predict([text])[0]
```

### 3.2 Named Entity Recognition (NER)

Named Entity Recognition (NER) is implemented using the SpaCy library, which is used to extract important entities such as product names and order IDs from user input. The model identifies and categorizes these entities into predefined labels (such as ORDER_ID). Below is the code for NER implementation:

```
import spacy

# Load SpaCy for NER
nlp = spacy.load("en_core_web_sm")

# Function to extract entities using SpaCy NER
def extract_entities(text):
    doc = nlp(text)
    return [(entity.text, entity.label_) for entity in doc.ents]
```

## 4. Retrieval-Augmented Generation (RAG) Implementation

Retrieval-Augmented Generation (RAG) is used to answer user queries by retrieving relevant answers from a predefined FAQ dataset. The retrieval system uses cosine similarity to compare the query with the FAQ questions and returns the most relevant answer. Below is the code for the RAG implementation:

```
from sklearn.metrics.pairwise import cosine_similarity

# Sample FAQ dataset
faq_data = [
    {"question": "What is the return policy?", "answer": "You can return items within 30
days."},
    {"question": "How do I track my order?", "answer": "You can track your order by logging
into your account."},
    {"question": "What payment methods are accepted?", "answer": "We accept credit cards,
PayPal, and more."}
]

# Function to retrieve the most relevant FAQ answer
def retrieve_answer(query):
    faq_texts = [faq["question"] for faq in faq_data]
    faq_matrix = vectorizer.fit_transform(faq_texts)
    query_vector = vectorizer.transform([query])
    similarities = cosine_similarity(query_vector, faq_matrix)
    best_match_idx = similarities.argmax()
    return faq_data[best_match_idx]["answer"]
```

## 5. LLM Prompts and Outputs

### 5.1 Product Recommendations

Prompt: 'Can you recommend me some wireless headphones under $100?'
Output: 'Here are some wireless headphones under $100 that you might like: [Product 1, Product 2, Product 3].'

### 5.2 Order Tracking

Prompt: 'When will my order #12345 be delivered?'
Output: 'Your order #12345 is scheduled to be delivered in 3 days.'

### 5.3 Return Policy Inquiry

Prompt: 'What is your return policy?'
Output: 'You can return items within 30 days.'

## 6. Test Use Case Interaction

Below is an example transcript of the interaction between the user and the AI-powered virtual assistant:

1. User: 'Hi, I am looking for a wireless headphone under $100.'
   Bot: 'Here are some wireless headphones under $100 that you might like: [Product 1, Product 2, Product 3].'
2. User: 'When will my order #12345 be delivered?'
   Bot: 'Your order #12345 is scheduled to be delivered in 3 days.'
3. User: 'What is your return policy?'
   Bot: 'You can return items within 30 days.'