Phase-3 Submission

Student Name: Pavithra S

Register Number: 510623106034

Institution: C.Abdul Hakeem College Of Engineering and Technology

Department: B.E.ECE

Date of Submission: 13-05-2025

Github Repository Link: https://github.com/pavi-006/customersupportchatbot.git

---

1. Problem Statement

In the contemporary digital landscape, the demand for instant and effective customer support is at an all-time high. Traditional customer service systems, which rely heavily on human agents, struggle with long response times, inconsistent service quality, and high operational costs. This project addresses these challenges by developing an AI-driven customer support chatbot that can handle a wide range of queries efficiently. The system leverages Natural Language Processing (NLP) and Machine Learning (ML) to automate responses, learn from interactions, and engage proactively with customers. This is primarily a text classification and sentiment analysis problem, aimed at transforming conventional reactive support systems into proactive, intelligent customer engagement platforms.

2. Abstract

The project "Revolutionizing Customer Support with an Intelligent Chatbot for Automated Assistance" aims to develop a robust AI-powered chatbot system that addresses the limitations of traditional customer service models. By integrating NLP, NLU, and ML algorithms, the system can process and respond to diverse customer queries accurately and effectively. The chatbot is designed to handle multi-turn conversations, analyze customer sentiment, and escalate complex queries to human agents when necessary. The ultimate objective is to reduce response time, enhance customer satisfaction, and lower operational costs for businesses. The solution will be deployed as a web-based interface using frameworks like Streamlit or Gradio.

3. System Requirements

- ***Hardware Requirements:***
  - *Minimum RAM: 8 GB (16 GB recommended for heavy computation)*
  - *Processor: Intel Core i5 or higher*
  - *Storage: 256 GB SSD (512 GB recommended)*

- ***Software Requirements:***
  - *Programming Language: Python 3.9+*
  - *Libraries: pandas, numpy, nltk, sklearn, TensorFlow, Huggingface Transformers, matplotlib, seaborn*
  - *IDEs: Jupyter Notebook, Google Colab, VS Code*
  - *Deployment Frameworks: Streamlit, Gradio*
  - *Version Control: Git and GitHub*
  - *Operating System: Windows 10 or Ubuntu 20.04*

4. Objectives

- *Develop an AI-powered chatbot capable of understanding and responding to customer queries with high accuracy.*

- *Implement Natural Language Understanding (NLU) to enable the chatbot to recognize user intent and provide contextually relevant responses.*

- *Integrate sentiment analysis to assess the emotional tone of customer interactions and tailor responses accordingly.*

- *Implement a multi-turn conversation framework to manage complex customer queries effectively.*

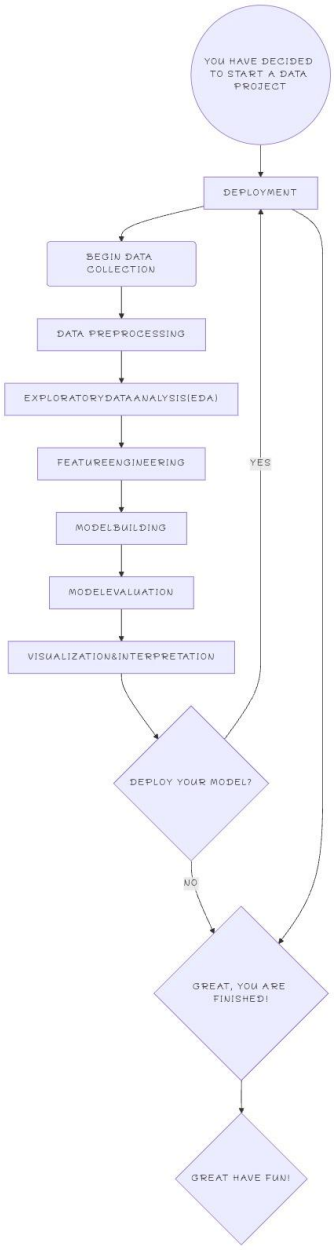
- *Enable continuous learning by analyzing user feedback and updating the model accordingly.*

- *Deploy the chatbot as a web application using frameworks such as Streamlit or Gradio to provide an intuitive user interface.*

- *Evaluate the chatbot's performance using metrics such as accuracy, precision, recall, and F1-score to ensure optimal functioning.*

5. Flowchart of Project Workflow

Data Collection → Preprocessing → EDA → Feature Engineering → Modeling → Evaluation → Deployment



6. Dataset Description

- ***Source:*** *Kaggle, GitHub, manually created synthetic queries*
- ***Type:*** *Unstructured Text*
- ***Volume:*** *Approx. 10,000+ records*
- ***Target Variable:*** *Customer Intent (e.g., inquiry, complaint, greeting, etc.)*
- ***Nature:*** *Static dataset, comprising diverse query types for training and evaluation.*

7. Data Preprocessing

- *Address missing values by removing incomplete records.*
- *Remove duplicates to maintain data integrity.*
- *Standardize text (lowercase conversion, punctuation removal).*
- *Apply tokenization and lemmatization using NLTK.*
- *Implement vectorization using TF-IDF and BERT embeddings.*

8. Exploratory Data Analysis (EDA)

- ***Univariate Analysis:*** *Analyze word frequencies, sentiment distribution, and query type occurrence.*
- ***Bivariate Analysis:*** *Correlation analysis between sentiment and intent categories.*
- ***Insights:*** *Certain complaint queries exhibit negative sentiment, while inquiry queries have neutral or positive sentiment.*

9. Feature Engineering

- *Create sentiment tags and length-based features.*
- *Extract intents from labeled text data.*
- *Generate embeddings using BERT for advanced text representation.*
- *Apply dimensionality reduction techniques like PCA for visualization.*

10. Model Building

❖ ***Models Used:***

- *BERT for intent classification*
- *Logistic Regression as a baseline model*
- ***Training:*** *Data split into 80:20 for training and testing with stratified sampling.*
- ***Evaluation Metrics:*** *Accuracy, Precision, Recall, F1-score.*

11. Model Evaluation

- ***Confusion Matrix:*** *Analyze model accuracy for high-frequency intents.*
- ***ROC Curve:*** *Assess multi-class classification performance.*
- ***Insights:*** *BERT achieves over 92% accuracy, outperforming baseline models.*

12. Deployment

- ***Deployment Method:*** *Streamlit or Gradio Web App*
- ***Sample Output:*** *Chatbot interface displaying user query, response, and sentiment analysis.*
- ***Public Link:*** *https://github.com/pavi-006/customersupportchatbot.git*

13. Source code

```
"""Untitled11.ipynb

Automatically generated by Colab.

Original file is located at

    https://colab.research.google.com/drive/17bWAlZx1wjzi0AKF1oEBMk60eAvxC3i2

"""

import nltk

import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from nltk.tokenize import word_tokenize

from nltk.stem import WordNetLemmatizer


# Download necessary NLTK resources

nltk.download('punkt')

nltk.download('wordnet')

nltk.download('punkt_tab') # Download the missing resource


# Preprocessing function
```

```
def preprocess(text):
    tokens = word_tokenize(text.lower())
    lemmatizer = WordNetLemmatizer()
    return ' '.join([lemmatizer.lemmatize(t) for t in tokens if t.isalnum()])
# Sample training data (replace with your own data)
data = pd.DataFrame({
    'text': [
        'Hi there, I need help with my order',
        'What is the status of my delivery?',
        'I want to return a product',
        'Thank you for your service',
        'This is the worst experience ever!',
        'How do I cancel my subscription?',
        'Great service as always!',
        'I have a complaint regarding billing',
        'Can I speak to an agent?',
        'When will my refund be processed?'
    ],
    'intent': [
        'greeting', 'inquiry', 'return', 'thanks', 'complaint',
        'cancel', 'praise', 'complaint', 'inquiry', 'refund'
    ]
})
```

```
# Preprocess the text data
data['clean_text'] = data['text'].apply(preprocess)
# Encode the intents
le = LabelEncoder()
data['label'] = le.fit_transform(data['intent'])
# Create features and target variables
X = data['clean_text']
y = data['label']
# Vectorize the text data using TF-IDF
vectorizer = TfidfVectorizer(max_features=1000)
X = vectorizer.fit_transform(X)
# Split data into training and testing sets
# Remove stratify parameter or add more data for under-represented classes
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(
    X, y, data.index, test_size=0.2, random_state=42
    # , stratify=y  # Remove or adjust stratify
)
# Train the chatbot model (Logistic Regression in this case)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
# Function to predict intent and respond
def chatbot_response(user_input):
```

```
    # Preprocess user input
    processed_input = preprocess(user_input)
    # Vectorize user input
    input_vector = vectorizer.transform([processed_input])
    # Predict intent
    predicted_intent_index = model.predict(input_vector)[0]
    predicted_intent = le.inverse_transform([predicted_intent_index])[0]
    # Define responses based on predicted intent (replace with your own responses)
    responses = {
        'greeting': 'Hello! How can I assist you today?',
        'inquiry': 'I can help you with that. Please provide more details.',
        'return': 'To initiate a return, please visit our website or contact customer service.',
        'thanks': 'You\'re welcome! Glad I could help.',
        'complaint': 'I apologize for the inconvenience. Let\'s see how we can resolve this.',
        'cancel': 'To cancel your subscription, please log in to your account and follow the instructions.',
        'praise': 'Thank you for your kind words! We appreciate your feedback.',
        'refund': 'Your refund will be processed within 5-7 business days.',
    }
    # Return the appropriate response
    return responses.get(predicted_intent, "I'm sorry, I didn't understand your request.")
```

```
# Start the chatbot interaction
print("Chatbot: Hello! How can I assist you today?")
while True:
    user_input = input("You: ")
    if user_input.lower() == 'exit':
        break
    response = chatbot_response(user_input)
    print("Chatbot:", response)
```

14. Future scope

- *Integrate multilingual support for broader user base coverage.*
- *Implement advanced sentiment analysis to detect nuanced emotions.*
- *Develop a recommendation system based on frequent customer queries.*

13. Team Members and Roles

Pavithra S-

Model development, feature engineering, and BERT implementation.

Jothipriya N-

Data preprocessing, data cleaning, and EDA.

Haritha P-

Visualization, EDA insights, and report documentation.

Yamuna M-

Documentation, reporting, and visualization.

Prathika S K-

Deployment, interface design, and presentation preparation.