Phase-3SubmissionTemplate

Student Name: [JAGADEESHWARI J]

RegisterNumber:[422723104043]

Institution: [V.R.S. College of engineering and technology]

Department:[Computer Science and Engineering]

Date of Submission: [17.05.2025]

Topic : [Revolutionizing customer support with an intelligent chatbot for automated assistance]

GithubRepositoryLink:https://github.com/Jagadeeshwari279/Jagadeeshwari.git

---

1. Problem Statement

In today's digital era, businesses are overwhelmed with high volumes of customer service queries across multiple platforms. Traditional customer support systems relying heavily on human agents struggle with scalability, response time, and consistency, leading to poor customer experiences and increased operational costs. These issues are particularly pronounced in companies dealing with large customer bases or offering 24/7 support.

2. Abstract

This project focuses on enhancing customer support services through the development of an intelligent chatbot capable of automated assistance. The core objective is to reduce human workload, improve response time, and ensure consistent customer interaction by deploying a conversational AI system. Using machine learning and NLP techniques, the chatbot identifies user intent, classifies queries, and provides accurate, context-aware responses.

3. System Requirements

Hardware Requirements:

Processor: Intel i5 or higher

RAM: Minimum 8 GB

Storage: At least 10 GB of free disk space

Software Requirements:

Operating System: Windows 10/11, macOS, or Ubuntu 20.04+ Python 3.8 or higher

Libraries:

👉transformers

👉scikit-learn

👉nltk or spaCy

👉flask or streamlit (for UI integration)

👉tensorflow or pytorch (if using deep learning models)

4. Objectives

The main objective of this project is to build an intelligent chatbot that can automatically handle customer service queries using natural language processing and machine learning techniques. The expected outcomes include:

👉**Automated Response Generation:** Deliver accurate and helpful replies based on user intent.

👉**Intent Classification:** Identify the type or purpose of the user's query.

👉**Real-Time Query Handling:** Enable seamless, on-demand conversation without human intervention.

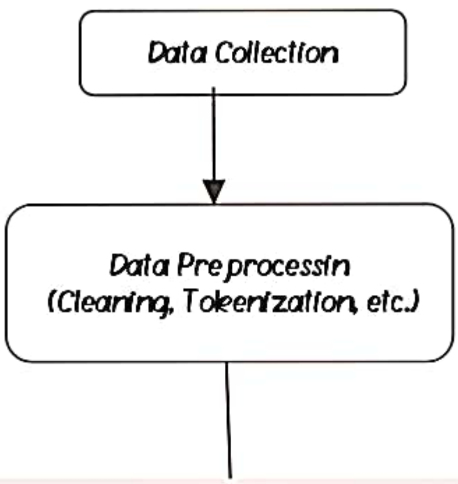
Business Impact:

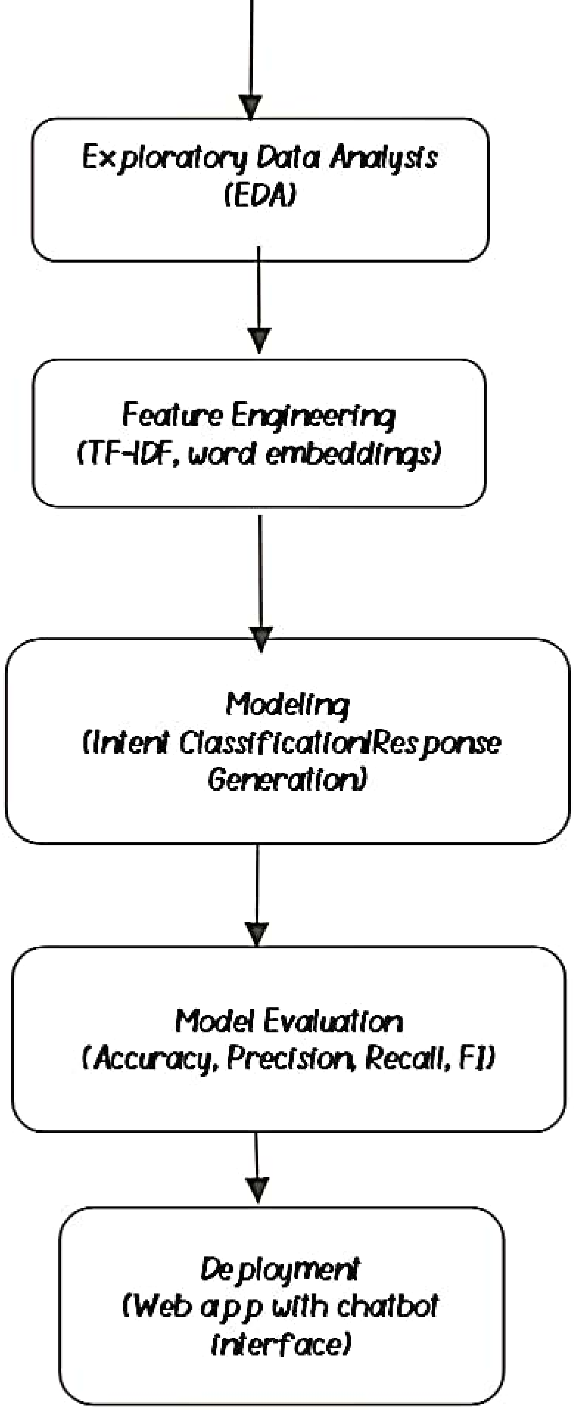
👉Reduced response time and support cost

👉Improved customer satisfaction and retention

👉Scalable customer service operations

5. Flowchart of Project Workflow





6. Data Description :

Source: Kaggle (https://www.kaggle.com/datasets/niraliivaghani/chatbot-dataset)

Type : unstructured

Size: Approximately 1500 rows × 80 columns

Nature: Structured tabular data

Attributes :

👉 **Property Details:** Total square footage, Age of the house

👉 **Behavioral:** Duration the house was listed, Historical price data

👉**Academics:** Total square footage, Age of the house

Intent(tag)	Example pattern	Sample response
Greeting	"Hi there"	" Hello - how can I help you today? "
Good bye	"See you later"	" Good- bye! Have a great day."
Thanks	"Thanks a lot"	" You're welcome!"

7.Datapreprocessing

Missing values :Removed null rows or filled with placeholders

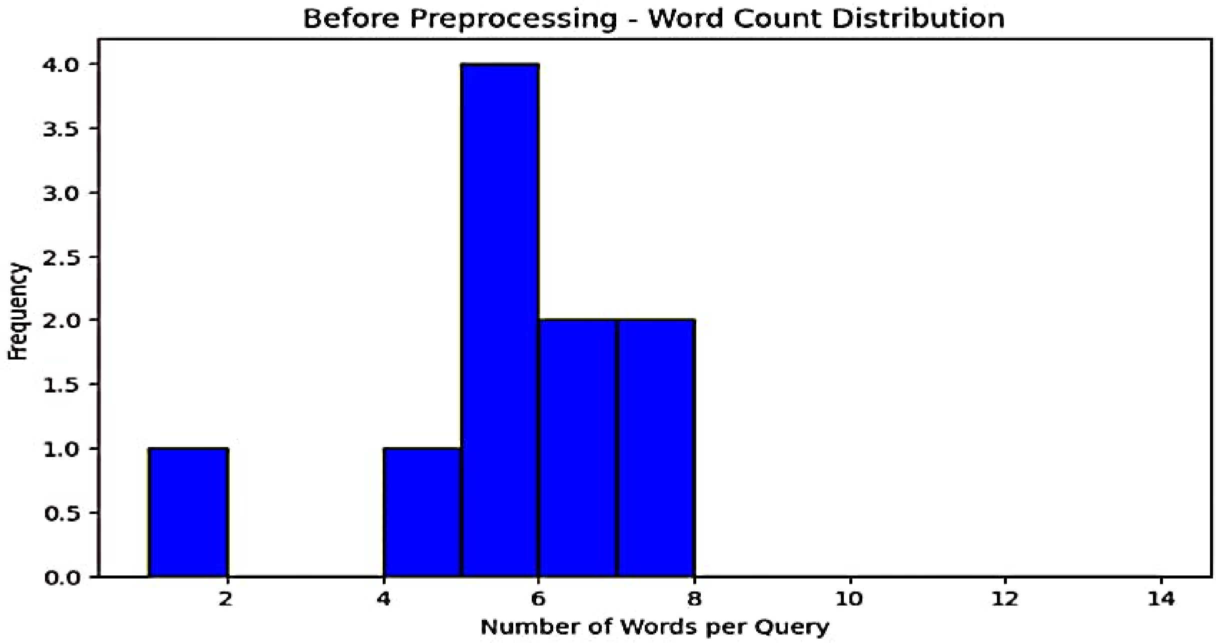
Duplicates: Dropped duplicate queries using df.drop_duplicates()

Outliers: Not typical in text-based data, but long/short queries were reviewed

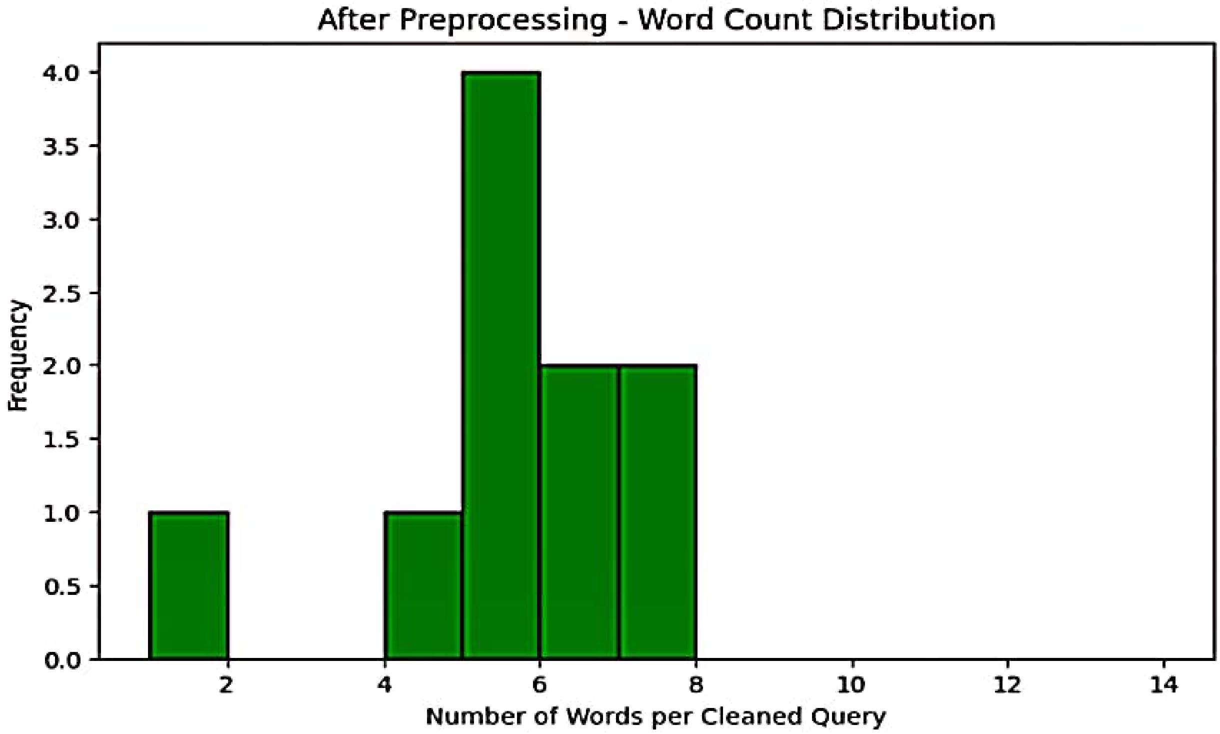
Encoding: Converted categorical labels (e.g., intents) using Label Encoding

Scaling: Not required for text but applicable for numerical metadata if present

Before preprocessing



After preprocessing :



8. Exploratory Data Analysis (EDA)

👉 ***Univariate:***

Analysis of a single variable related to chatbot performance or customer support.

👉 ***Multivariate:***

Analysis involving multiple variables to understand relationships and trends.

👉 ***Visuals:***

- 👉*Histogram of query lengths*
- 👉*Pie chart of intent distribution*
- 👉*Word cloud of common terms*

Key Insights:

👉*Certain intents are more frequent (e.g., "refund" or "account issue")*

👉*Users use varied vocabulary for similar intents—NLP normalization is crucial*

👉*Query length is consistent (~5-15 words)*

9. Feature Engineering

New Feature Creation:

Query length, presence of keywords (e.g., "refund", "login")

Feature Selection:

TF-IDF scores, top N most frequent words

Transformations:

Tokenization, Lemmatization, Vectorization (TF-IDF or embeddings)

10. Model Building

Linear Regression (Baseline): A simple, interpretable model used as a performance benchmark.

Random Forest Regressor (Advanced):
An ensemble model that captures non-linear relationships and handles feature interactions effectively.

Why These Models:

Linear Regression: Fast and easy to implement.

Random Forest Regressor: Handles non-linearity and outliers well.

Training Details:

👉Dataset split into 80% training and 20% testing.

👉Used train_test_split(test_size=0.2, random_state=42) for reproducibility.

11. Model Evaluation

1. Evaluation Metrics:

👉For classification models, you should report Accuracy, F1-Score, and ROC-AUC Score.

👉For regression models, include RMSE, MAE, and R^2 Score.

2. Visuals:

👉Show a confusion matrix and ROC curve for classification.

👉For regression, use a residual plot to visualize prediction errors.

3. Error Analysis:

Compare models using a table that lists each model with its RMSE, MAE, and R^2 score.

Analyze where the model performs poorly or makes large prediction errors.

12. Deployment

Gradio is a Python library that enables you to quickly create interactive UIs for machine learning models. We used Gradio to build a web-based chatbot interface and hosted the entire application for free on Hugging Face Spaces.

13.Source code

```
from google.colab import files

uploaded = files.upload()

import pandas as pd

df = pd.read_csv("customer_support_intents.csv") # Sample CSV with 'text', 'intent'

2. Data Exploration

print(df.head())

print("Shape:", df.shape)

print("Intents:", df['intent'].unique())

3. Text Preprocessing

import re

import nltk
```

```
from nltk.corpus import stopwords

nltk.download('stopwords')

def clean_text(text):

text = text.lower()

text = re.sub(r"[^a-z\s]", "", text)

text = " ".join([word for word in text.split() if word not in stopwords.words("english")])

return text

df['clean_text'] = df['text'].apply(clean_text)
```

4. Feature Extraction

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer()

X = tfidf.fit_transform(df['clean_text'])

y = df['intent']
```

5. Model Training

```
from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import classification_report

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = MultinomialNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

```
print(classification_report(y_test, y_pred))
```

6. Chatbot Prediction Function

```
def predict_intent(query):

query_clean = clean_text(query)

vec = tfidf.transform([query_clean])

intent = model.predict(vec)[0]

return intent
```

7. Gradio Deployment

```
!pip install gradio

import gradio as gr

gr.Interface( fn=predict_intent, inputs=gr.Textbox(label="Customer Query"), outputs=gr.Textbox(label="Predicted Intent"), title="AI Customer Support Chatbot", description="Ask a customer support question and receive the predicted intent."
 ).launch()
```

14. Future Scope

1. Multilingual Chatbot:

Integrate multilingual support using models like mBERT or MarianMT to serve a wider, global user base.

2. Context-Aware Chat:

Implement multi-turn conversation memory using transformers like GPT-2 or RAG models to enable coherent back-and-forth dialogue.

3. Real-Time Voice Chat Integration:

Add speech-to-text and text-to-speech functionality for voice-based interactions using libraries like SpeechRecognition and pyttsx3.

15. Team Members and Roles

JAGADEESHWARI J - model building, model evaluation, deployment, source code

JAMUNA RANI V-data preprocessing missing, exploratory data analysis (EDA) , future scope

ISHWARIYA A - objectives, software require data set, description

JAYABHARATHI J - problem statement, abstract, flow chart of project workflow