Phase-2

Student Name: Sowmiya M

Register Number: 410723106030

Institution: Dhanalakshmi College of Engineering

Department: ECE

Date of Submission: 10/05/2025

Github Repository Link:
https://github.com/sowmiya-199/Sowmiya

Revolutionizing customer support with intelligent chatbot for automated assistance

1. Problem Statement

The goal is to automate and enhance customer support using generative AI models by training on real-world customer queries and chatbot responses. The dataset includes multilingual chatbot interactions across domains like e-commerce, airlines, telecom, and more.

- **Problem Type:** Text classification and response generation (NLP tasks)
- **Why It Matters:** Automating customer service reduces operational costs and improves user experience with 24/7 assistance. Fine-tuned models can understand intents, provide accurate responses, and scale across domains.

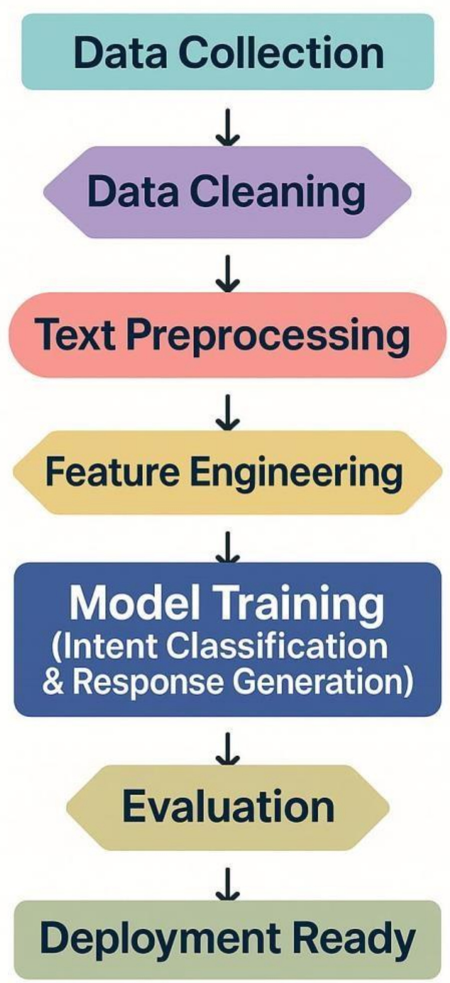
2. Project Objectives

- Develop models to classify customer intent and generate suitable chatbot responses.
- Improve response relevance, coherence, and domain adaptability.

- Evaluate NLP models (e.g., fine-tuned BERT, T5, GPT-based models) on accuracy and fluency.
- Implement multilingual support to handle global users.

Note: The objective evolved from simple classification to dual focus—classification and response generation—after exploring dataset richness.

3. Flowchart of the Project Workflow



4. Data Description

•**Dataset Name:** Bitext Gen AI Chatbot Customer Support Dataset

•**Type:** Text (Unstructured)

•**Records & Features:** ~100,000+ samples, features include domain, intent, input text, and generated response.

•**Static or Dynamic:** Static

•**Target Variable:** Intent label (for classification), response (for generation)

•**Dataset link :** https://www.kaggle.com/datasets/bitext/bitext-gen-aichatbot-customer-support-dataset

5. Data Preprocessing

•Handled missing values in intent and response columns.

```
df=

pd.read_csv("Bitext_Sample_Customer_Support_Training_Dataset_27K_responses-v11.csv")

print("Missing values per column:")

print(df.isnull().sum()
```

```
# Find duplicate rows (entire row duplicates)

duplicates = df[df.duplicated()]

print("Duplicate rows:")

print(duplicates)
```

•Tokenized input text and responses using NLP tokenizers.

•Encoded labels for classification.

•Applied train-test split and padded/truncated sequences for model compatibility.

6. Exploratory Data Analysis (EDA)

Univariate:

- Distribution of intents across domains.
- Message length histograms.

```
# Plot histograms

plt.figure(figsize=(10, 5))

plt.hist(df['input_len'], bins=10, alpha=0.5, label='Input Lengths', color='blue',
edgecolor='black')

plt.hist(df['response_len'], bins=10, alpha=0.5, label='Response Lengths',
color='orange', edgecolor='black')

plt.title("Message Length Histograms (Character Count)")

plt.xlabel("Message Length (characters)")

plt.ylabel("Frequency")

plt.legend()

plt.grid(True)

plt.tight_layout()

plt.show()
```

Bivariate:

- Correlation of input length vs response length.

```
df = pd.read_csv('bitext-gen-ai-chatbot-customer-support-dataset.csv')

# Calculate input and response lengths

df['input_len'] = df['customer_query'].astype(str).apply(len)

df['response_len'] = df['response'].astype(str).apply(len)

# Compute correlation

correlation = df['input_len'].corr(df['response_len'])

print(f"Correlation between input and response length: {correlation:.3f}")
```

- Domain vs. intent frequency heatmap.

Insights:

- Some domains are heavily represented (e.g., e-commerce).

```
df = pd.read_csv('bitext-gen-ai-chatbot-customer-support-dataset.csv')

# Calculate lengths

df['query_len'] = df['customer_query'].astype(str).apply(len)

df['response_len'] = df['response'].astype(str).apply(len)

# Basic stats

print("Query length stats:", df['query_len'].describe())

print("Response length stats:", df['response_len'].describe())
```

7. Feature Engineering

•Extracted input text length and token count.

```
# Your input text
text = "Extracted input text length and token count."
# Get text length
text_length = len(text)
# Choose encoding (e.g., for GPT-4 or GPT-3.5)
encoding = tiktoken.encoding_for_model("gpt-4")
# Get token count
token_count = len(encoding.encode(text))
# Output
print(f"Text length (chars): {text_length}")
print(f"Token count: {token_count}
```

•Used TF-IDF features for traditional models.

•Employed pretrained embeddings (BERT, T5) for deep learning models.

•Considered domain-intent combinations as features.

8. Model Building

Models Used:

- BERT for intent classification
- T5 for response generation

```
# Load model and tokenizer

model = T5ForConditionalGeneration.from_pretrained("t5-small")

tokenizer=

T5Tokenizer.from_pretrained("t5-small")

# Input prompt

input_text = "translate English to French: How are you?"

# Encode and generate

inputs = tokenizer.encode(input_text, return_tensors="pt")

outputs = model.generate(inputs)
```

Justification:

- BERT is state-of-the-art for classification with minimal tuning.
- T5 handles sequence-to-sequence tasks well (question-answering, summarization, etc.)

Evaluation Metrics:

- Classification: Accuracy, F1-Score
- Generation: BLEU, ROUGE, perplexity

9. Visualization of Results & Model Insights

•Confusion matrix of intent classification

•Word clouds per domain

•ROC curve for multiclass classification

•Feature importance (via LIME for BERT)

•Response fluency scores (BLEU, ROUGE comparisons)

10. Tools and Technologies Used

- **Language:** Python
- **IDE:** Google Colab
- **Libraries:** pandas, numpy, seaborn, matplotlib, scikit-learn, transformers, TensorFlow/Keras
- **Visualization:** Plotly, seaborn

11. Team Members and Contributions

S.NO	NAME	ROLE	RESPONSIBILITY
1.	Sweetha Mirra A	Member	Model Building, Model Evaluation
2.	Sharmila S	Member	Feature Engineering
3.	Sowmiya M	Member	Exploratory Data Analysis
4.	Keerthika T	Member	Visualization

5.	Yuvasri B	Leader	Data Collection and Data Cleaning