Revolutionizing Customer Support with an Intelligent Chatbot

# 1. Problem Statement

Customer service systems often struggle with high volumes of repetitive queries, resulting in long wait times and inconsistent service.

The goal is to create an intelligent chatbot that provides real-time, automated responses using natural language understanding.

- Problem Type: Text classification and sequence modeling (NLP)

- Why it Matters: Automates routine support tasks, reduces costs, improves customer experience, and enables 24/7 assistance.

# 2. Project Objectives

- Key Technical Objective: Design a conversational agent using NLP techniques to classify intent and respond accurately.

- Model Goals:

- High accuracy in intent recognition

- Fast response time

- User satisfaction improvement

- Evolved Goals: Post-data exploration, focus on intent disambiguation and support for multi-turn dialogues.

# 3. Flowchart of the Project Workflow

Data Collection → Data Preprocessing → Intent Classification → Entity Recognition → Dialog Management → Evaluation & Deployment

# 4. Data Description

- Dataset Name: Customer Support Dataset (e.g., from Kaggle or custom collected)

- Origin: Open APIs, company chat logs, or public datasets

- Type: Unstructured (text)

- Records: ~10,000+ support dialogues

- Features: Text, intents, entities

- Target: Intent label (for supervised learning)

- Nature: Dynamic (new queries constantly added)

# 5. Data Preprocessing

- Missing Values: Removed incomplete queries

- Duplicates: Merged identical utterances

- Outliers: Removed non-relevant text using regex filters

- Data Types: All converted to lowercase strings

- Encoding: Tokenization, padding, and embeddings

- Normalization: Removed stopwords, punctuation

- Documented in preprocessing scripts

# 6. Exploratory Data Analysis (EDA)

- Univariate: Most frequent intents were 'password reset', 'billing', and 'technical support'

- Bivariate: Common co-occurrence of intents and entities analyzed using co-occurrence matrices

- Multivariate: Clustered query embeddings using t-SNE

- Insights:

- Some intents are ambiguous and need disambiguation

- Need for fallback and sentiment handling

# 7. Feature Engineering

- Extracted named entities (NER)

- Word embeddings (Word2Vec, BERT)

- POS tagging and dependency parsing

- Created response templates for high-frequency intents

- Justified by performance gains in intent classification

# 8. Model Building

- Models Used: Logistic Regression (baseline), BERT (transformer-based)

- Split: 80/20 Train-Test split with stratification

- Metrics: Accuracy, F1-score, Confusion Matrix

- Justification:

- Logistic Regression for baseline performance

- BERT for contextual understanding and multi-intent classification

# 9. Visualization of Results & Model Insights

- Confusion Matrix: Highlighted confusions between similar intents

- ROC Curve: Evaluated binary and multi-class performance

- Attention visualization: Used BERT attention to interpret predictions

- Feature Importance: Word embeddings and context windows shown to impact predictions

# 10. Tools and Technologies Used

- Programming Language: Python

- IDE/Notebook: Jupyter Notebook, Google Colab

- Libraries: pandas, sklearn, spaCy, NLTK, transformers, matplotlib, seaborn

- Visualization: Matplotlib, Plotly

# 11. Team Members and Contributions

- Alice: Data Collection, Preprocessing

- Bob: NLP Pipeline, Intent Classification Model

- Carol: Dialog Management, Evaluation

- Dave: Visualization, Reporting, Deployment