Automatic Customer Support Ticket Classification and Response Generation using RNN & Gemini LLM

---

1. Introduction

Customer support organizations handle a large volume of tickets spanning multiple departments. Manual ticket classification and routing is time-consuming and prone to inconsistencies. Automating this process using Natural Language Processing (NLP) and Deep Learning can significantly improve efficiency and customer satisfaction.

This project presents an end-to-end system for automatic classification of customer support tickets using a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM). Additionally, the system integrates Google Gemini 2.5 Flash to generate polite and empathetic automated responses, providing immediate acknowledgment to customers.

---

2. Dataset Description

- **Source:** Hugging Face
- **Dataset:** `Tobi-Bueck/customer-support-tickets`
- **Access Method:** Hugging Face Dataset API

Fields Used

- **body:** Customer support ticket text (input feature)
- **queue:** Assigned department or queue (target variable)

Dataset Characteristics

- Total samples: approximately 61,000
- Number of queue categories: 52
- Highly imbalanced class distribution

Each data record represents a real-world customer support ticket, making the dataset suitable for supervised multi-class text classification.

---

3. Exploratory Data Analysis

Exploratory analysis was conducted to understand the dataset characteristics:

- Inspection of data types and missing values
- Analysis of queue distribution
- Identification of class imbalance
- Analysis of ticket length distribution

The 95th percentile of ticket length was used to determine the maximum sequence length for text padding and truncation.

---

4. Data Preprocessing

4.1 Text Cleaning

- Conversion of text to lowercase
- Removal of punctuation, digits, and special characters
- Removal of extra whitespace

4.2 Tokenization and Padding

- Keras Tokenizer with a vocabulary size of 15,000
- Use of an out-of-vocabulary token (`<unk>`)
- Padding and truncation of sequences to a fixed length of 102 tokens

4.3 Label Encoding

- Queue labels encoded into numerical values using `LabelEncoder`
- Encoders saved for inverse transformation during inference

---

5. Dataset Splitting

To preserve class distribution across datasets, stratified sampling was used:

- Training set: 70%
- Validation set: 15%
- Test set: 15%

---

6. Model Architecture

A **Many-to-One Bidirectional LSTM** architecture was implemented using TensorFlow and Keras.

Architecture Components

- Embedding layer
- Two Bidirectional LSTM layers
- Dropout layers for regularization
- Fully connected dense layers
- Softmax output layer for multi-class classification

This architecture enables the model to capture contextual dependencies within ticket text while reducing overfitting.

---

7. Class Imbalance Consideration

The dataset exhibits significant class imbalance, with certain queues such as *Technical Support* and *Customer Service* containing substantially more samples than others. The impact of this imbalance was carefully analyzed during exploratory data analysis.

Although class weighting was considered as a potential mitigation strategy, the final model was trained **without applying class weights**, as preliminary experiments showed limited improvement in validation performance and occasional instability during training. Instead, the model relied on stratified dataset splitting, dropout regularization, and hyperparameter tuning to achieve stable convergence and reasonable generalization.

---

8. Model Training

Training Configuration

- Optimizer: Adam
- Loss Function: Sparse Categorical Crossentropy
- Epochs: 10
- Batch Size: Tuned experimentally
- Callbacks:
    - EarlyStopping
    - ModelCheckpoint (saving the best validation model)
    - ReduceLROnPlateau

The best-performing model was saved as `best_lstm_model.keras`.

---

9. Hyperparameter Tuning

Hyperparameter tuning was conducted to identify the optimal model configuration. The following parameters were varied:

- Embedding dimension: 100, 200
- Number of LSTM units: 128, 256
- Batch size: 32, 64

Tuning Results

Embedding Dim	LSTM Units	Batch Size	Validation Accuracy
200	128	32	**0.5364**
200	128	64	0.5252
200	256	64	0.5020
200	256	32	0.4951
100	128	32	0.4881
100	256	32	0.4836
100	128	64	0.4541
100	256	64	0.4271

Observations

- Larger embedding dimensions provided richer semantic representations
- Moderate LSTM capacity (128 units) generalized better than larger models
- Smaller batch sizes improved validation performance

The configuration **Embedding = 200, LSTM Units = 128, Batch Size = 32** achieved the highest validation accuracy and was selected as the final model.

---

10. Model Evaluation

The selected model was evaluated on the held-out test dataset.

Evaluation Metrics

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

Results

- Test Accuracy: approximately 54%
- Macro-average F1-score: approximately 0.71
- Weighted-average F1-score: approximately 0.54

The model demonstrated strong performance on many specific categories, while broader and overlapping classes such as *Customer Service* and *Technical Support* remained more challenging.

---

11. Gemini LLM Integration

Model Used

- **Google Gemini 2.5 Flash**

Integration Workflow

1. Customer ticket text is classified using the trained LSTM model
2. The predicted queue and ticket text are passed to Gemini
3. Gemini generates a polite, empathetic acknowledgment reply

Prompt design ensured responses were professional and avoided placeholders or signatures.

---

12. Streamlit Application

A Streamlit-based web application was developed to demonstrate the system:

- User enters ticket text
- The model predicts the appropriate queue
- Gemini generates an automated response in real time

This application serves as a functional prototype for real-world deployment.

---

13. Project Deliverables

- Trained LSTM model
- Tokenizer and label encoder files
- Model training notebook
- Streamlit application

- Project documentation

---

14. Limitations and Future Work

Limitations

- Moderate accuracy due to overlapping queue semantics
- No use of additional metadata such as priority or tags
- Sequential models may struggle with very long dependencies

Future Enhancements

- Transformer-based models (BERT, RoBERTa)
- Hierarchical classification strategies
- Incorporation of ticket metadata
- Continuous learning with user feedback

---

15. Conclusion

This project demonstrates a complete NLP pipeline for automatic customer support ticket classification and response generation. By combining deep learning-based text classification with generative AI, the system improves ticket routing efficiency while enhancing customer experience through immediate, empathetic responses. The project highlights the practical application of RNNs and LLMs in modern customer support automation.