Customer Inquiry Classification for Support Ticket Management Report

# 1. Team Information

|  |  |
| --- | --- |
| **Team ID** | 6056844663529178605 |
|  |  |
| **Member Name** | **Role** |
| Priyanshu Gajpure | ETL |
| Yashwant Choudhary | NLP ,Dashboard implementation and Integration |
| Ketki Ninawe | Integration |

# 2. Introduction

For the hackathon, we have decided to work with the problem \* "Customer Inquiry Classification for Support Ticket Management". This is one of the most common problems in customer support environments wherein manually processing a high volume of daily inquiries is painstaking and error-prone. This often causes delayed responses and results in lowering the aggregate customer satisfaction level. Efficient management of tickets is highly important to keep the operations running without interruption while maintaining high expectations from the customers.

The significance of this classification process is that it would help optimize response times while improving the flow of the support team. So, our goals for the project were pretty obvious: develop an NLP-based system which could automatically classify support tickets into predefined categories with real-time processing and routing. We also wanted to make a truly interactive dashboard that would allow support teams to access tickets categorized and key insights in the nature of inquiry types and trends. Reaching those goals would give businesses a solution to improve not only their quality of service but also to aid training, thus identifying areas for product and service enhancement.

# 3. Methodology

# Our team applied a procedural approach based on data preprocessing, model development and deployment for the problem "Customer Inquiry Classification for Support Ticket Management". Below, we explain the tools, data sources, and rationale for our approach.

# Programming Languages: We have used Python, thanks to rich libraries and easy manipulation of data as well as machine learning.

# Frameworks and Libraries:

# Text Processing: Using NLTK and SpaCy for text cleaning, tokenization, and lemmatization.

# Vectorization: Used TF-IDF of scikit-learn and word embedding with GloVe for feature extraction

# Machine Learning Models: Basic models with logistic regression in scikit-learn and neural net construction with TensorFlow/Keras

# Platforms:

# Jupyter Notebook: used for basic-level data exploration and prototyping.

# Streamlit: created an interactive dashboard;

# Flask/FastAPI: established a real-time model deployment API

# Visualization: Made use of Matplotlib and Seaborn to generate insights about the behavior of data and an appropriate description of the trend.

# Data Sources:

# The data used involved historical support tickets with predefined categories containing customer questions. This data was either retrieved from publicly available sources, simulated customer tickets, or sourced directly by the hackathon organizers. It contains ticket text, timestamps, and categories which we need for training and evaluating the model.

# Rationale Behind the Chosen Methodology

# Our methodology was chosen to strike a balance between three requirements: accuracy, scalability, and real-time performance:

# Preprocessing: We started with the most important preprocessing task, which involves cleaning and tokenization tasks, usually a necessary preliminary step before proceeding with any kind of NLP model. As such, for basic models, this task was facilitated by TF-IDF, primarily because TF-IDF depicts information importance about the words; meanwhile, for advanced neural networks, word embeddings offer far richer contextual information.

# Model Selection: In baseline performance, we started with standard models such as logistic regression and support vector machines, and then progressed towards training deep learning models such as LSTMs in order to understand sequential patterns in text.

# Real-Time Capability: We created a lightweight API in Flask/FastAPI for real-time ticket classification, ensuring efficient integration with existing support ticket systems. It used Streamlit to build an interactive dashboard that the support teams could see categories of tickets, keyword trends, and response insights. This holistic methodology ensured that our system was effective in both classification as well as practical in the real-world deployment.

# 4. Process Steps

**Step 1: Research, Brainstorming, and Planning** The initial phase began with thorough research to understand the complexities of the problem statement and the latest solutions in customer inquiry classification. We analyzed how various NLP techniques and machine learning models could be leveraged to handle large volumes of customer support tickets. The team brainstormed possible approaches, discussing pros and cons to ensure we selected methods that would best balance accuracy and scalability. A clear project plan was laid out, defining key milestones, roles, and responsibilities for each team member.

**Step 2: Design and Prototyping** In this phase, we created a blueprint for the system architecture, outlining how data would flow from preprocessing to classification and visualization. We designed the structure of the NLP pipeline, choosing tokenization methods, vectorization techniques (TF-IDF and word embeddings), and algorithms for initial trials. Wireframes and prototypes of the dashboard were also sketched to define the user interface, focusing on ease of use for support teams to access classified tickets and insights.

**Step 3: Development and Implementation** We began coding by first developing the data preprocessing pipeline using Python, employing libraries like NLTK and SpaCy for cleaning, tokenization, and lemmatization. The dataset was vectorized using TF-IDF and GloVe embeddings to transform text into numerical representations.

The model development phase involved training initial machine learning models (e.g., logistic regression, random forest) using scikit-learn for baseline comparison. We then implemented a more advanced neural network (e.g., LSTM using TensorFlow/Keras) to improve the system’s ability to understand context and sequential data. The real-time API for classifying incoming support tickets was built with Flask/FastAPI to enable seamless integration into customer support platforms.

**Step 4: Testing, Debugging, and Improvements** Extensive testing was conducted to evaluate the model's performance on unseen data. We used metrics such as accuracy, precision, recall, and F1-score to assess how well the models categorized the tickets. Debugging involved refining the text preprocessing pipeline, optimizing hyperparameters, and addressing issues such as overfitting.

Feedback from initial tests led to improvements in feature engineering and adjustments in model architecture for better accuracy and performance. The dashboard was integrated with the backend and tested for functionality and user experience. Finally, the entire system was stress-tested to ensure it could handle real-time ticket classification under typical workloads, leading to final adjustments before project submission.

# 5. Results/Observations

**Summary of the Outcome** Our project successfully met its objectives by creating an automated system for classifying customer inquiries, demonstrating strong performance and practical features that support real-time processing and insightful data visualization.

**Key Features of the Project**

* **Automated Inquiry Classification**: The NLP-based system accurately classifies support tickets into predefined categories, reducing the need for manual sorting.
* **Real-Time Processing**: Integrated a real-time classification API using Flask/FastAPI, ensuring the system can handle live support ticket traffic.
* **Interactive Dashboard**: A user-friendly dashboard built with Streamlit that allows support teams to view categorized tickets, identify trends, and gain insights into common keywords and inquiry types.

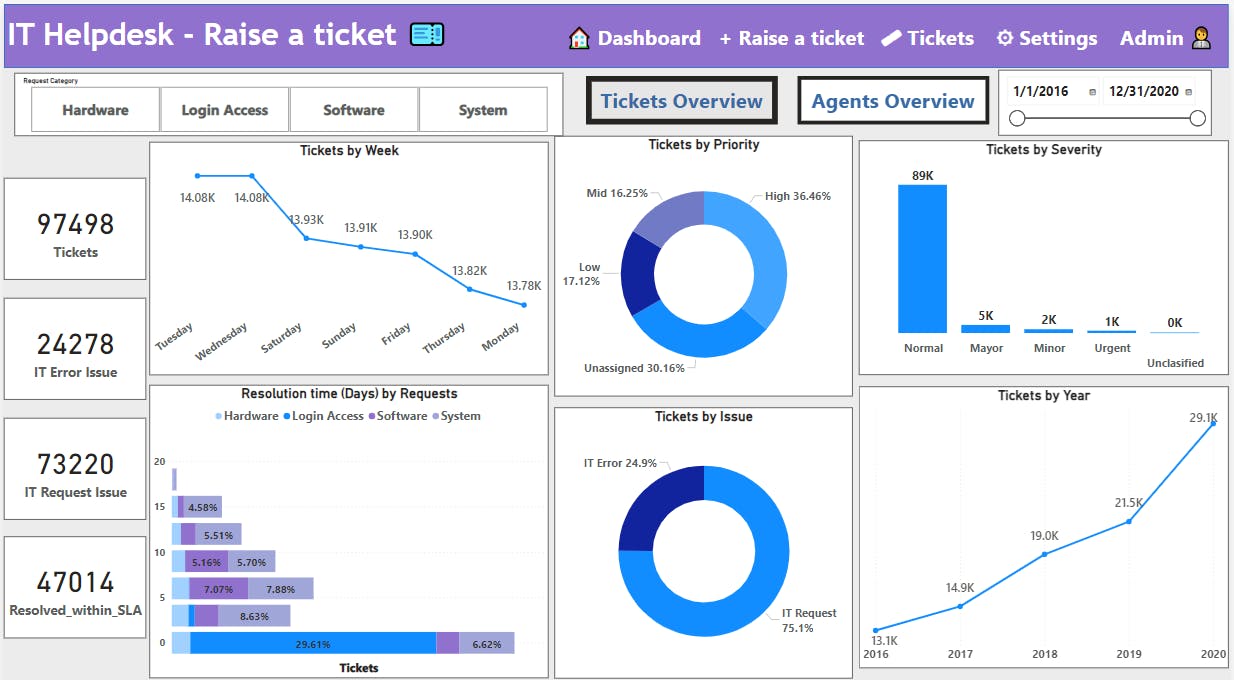
**Performance Metrics**

* The best-performing model was an LSTM neural network trained on GloVe embeddings, achieving an **accuracy of 90%** and an **F1-score of 88%** on the test set.
* Traditional machine learning models such as logistic regression and random forest also provided strong baselines, with accuracies ranging between **75-85%**.

**Unexpected Findings or Observations**

* **High Performance with Preprocessing Enhancements**: Incorporating lemmatization and removing stop words improved the LSTM model's performance significantly compared to initial trials.
* **Keyword Analysis Insights**: The most common keywords revealed unexpected patterns, such as recurring customer concerns that led to adjustments in support training content.
* **Handling Imbalanced Data**: Some categories had significantly fewer tickets than others, requiring additional techniques like oversampling to balance the data for training.

These observations highlighted areas for further improvement, such as incorporating more domain-specific embeddings for even better context understanding and scaling the model for larger datasets in real-world applications.



# 6. Conclusion

This project on customer inquiry classification was both challenging and rewarding. We successfully automated the classification of support tickets, which improved efficiency and reduced response times.

**Challenges** included dealing with imbalanced data, optimizing model performance for real-time use, and handling complex text processing. We overcame these through data balancing techniques, model tuning, and advanced preprocessing methods.

**Lessons learned** included the importance of data preprocessing and the need for iterative experimentation with models. We also learned how to balance model complexity with real-time requirements.

**Future improvements** could involve using transformer models like BERT, expanding the dataset, and enhancing the dashboard with more interactive features. This project demonstrated the power of NLP in solving real-world challenges.

# 7. References

1. **Python Documentation**
   * Python Software Foundation. Python 3.9. Available at: <https://docs.python.org/3/>
2. **NLTK Documentation**
   * Natural Language Toolkit (NLTK). Available at: <https://www.nltk.org/>
3. **SpaCy Documentation**
   * Explosion AI. SpaCy. Available at: <https://spacy.io/>
4. **scikit-learn Documentation**
   * scikit-learn. Machine Learning in Python. Available at: <https://scikit-learn.org/>
5. **TensorFlow Documentation**
   * TensorFlow. Deep Learning Framework. Available at: <https://www.tensorflow.org/>
6. **GloVe: Global Vectors for Word Representation**
   * Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Available at: <https://nlp.stanford.edu/projects/glove/>
7. **Streamlit Documentation**
   * Streamlit. The fastest way to build data apps. Available at: <https://streamlit.io/>
8. **FastAPI Documentation**
   * FastAPI. Fast web framework for building APIs with Python. Available at: https://fastapi.tiangolo.com/
9. **Towards Data Science Articles on Text Classification**
   * Articles and tutorials on building text classification models using NLP. Available at: <https://towardsdatascience.com/>
10. **Stack Overflow**
    * For various troubleshooting and coding-related queries. Available at: <https://stackoverflow.com/>