Automating Customer Service Using Sentiment Analysis

- VAMSHI SAINI.
- SWARNA DOPPA.
- HARSH KUMAR PATEL.
- PAVAN KALYAN RAGALA.

Content

DATA OVERVIEW

DATA CLEANING AND PREPROCESSING

DATABASE SETUP AND USAGE

MACHINE LEARNING PROCESS

FLASK APPLICATION OVERVIEW

FINDINGS AND RESULTS

CONCLUSION AND FUTURE WORK

ACKNOWLEDGEMENTS AND QUESTIONS

Data Used

- **Dataset**: Product reviews dataset.
- **Source**: Kaggle
- **Raw Data Characteristics**:
  - Contains approximately 5,000 product reviews.
  - Includes user comments, ratings, and metadata.
  - Data is unstructured, noisy, and diverse in sentiment.

Motivation

01

Understanding customer sentiments from product reviews.

02

Automating reply generation to streamline customer service.

03

Tracking and resolving user issues efficiently with ticket numbers.

04

Enabling businesses to gain insights and improve user experience.

Data Cleaning and Preprocessing

- **Why Clean the Data**:
  - Raw data contains duplicates, noise (special characters, emojis), and irrelevant content.
  - Inconsistent formatting (mixed cases, hyperlinks, punctuation).
- **Cleaning Techniques**:
  - Removed duplicates and empty comments.
  - Tokenized text into individual words.
  - Standardized text (lowercased, removed punctuation).
  - Removed stop words and special characters.
- **Tools Used**:
  - Python libraries: pandas, NLTK, re (regular expressions).

Before Cleaning

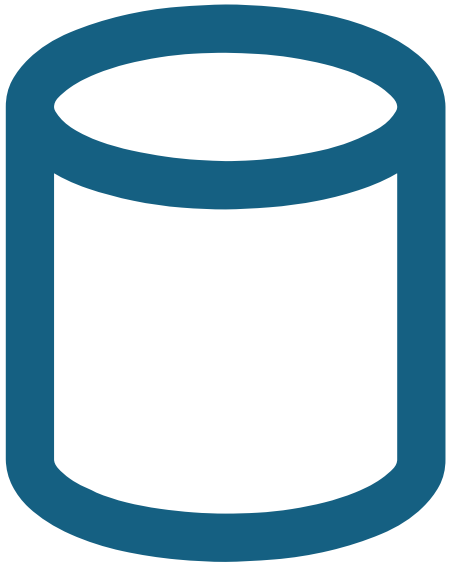
Raw Data Example:

- **User Comment**: "I REALLY LOOOOVE this product!!! ❤️❤️ www.example.com"
- **Issues**:
  - Mixed case: "I REALLY LOOOOVE"
  - Redundant characters: "LOOOOVE"
  - Special characters: "❤️❤️"
  - Link: "www.example.com"

After Cleaning

Cleaned Data Example:

- **User Comment**: "i really love this product"
- **Fixes**:
  - Converted to lowercase: "I REALLY LOOOOVE" → "i really loooove".
  - Reduced redundant characters: "loooove" → "love".
  - Removed special characters: "❤️❤️" → (removed).
  - Removed links: "www.example.com" → (removed).



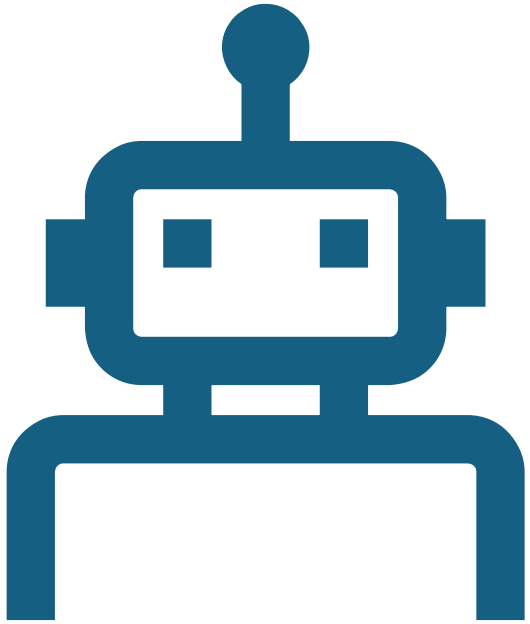
Database Setup and Usage

- **Why Use a Database**:
  - Efficient storage and retrieval of processed data.
  - Enables integration with the web application.
  - Ensures data consistency and scalability.
- **Database Setup**:
  - MySQL database configured to store cleaned and structured data.

Tables and Fields:

- **Table**: Comments
- **Fields**:
  - comment_id: Primary key.
  - product_id: Foreign key linking to products.
  - user_comment: Original user comment.
  - reply_comment: Generated reply.
  - sentiment: Sentiment classification (Positive, Neutral, Negative).
  - ticket_number: Assigned to negative comments.
  - created_at: Timestamp.

- **Data Storage**:
  - User comments and generated replies are stored with metadata.
  - Ticket numbers are assigned to negative comments for issue tracking.



Machine Learning Process Overview

- **Objective:**
  - Analyze user comments for sentiment classification.
  - Generate replies and assign ticket numbers for negative comments.
- **Process Overview:**
  - Data Preprocessing: Cleaning, tokenization, and vectorization.
  - Model Selection: Tested Logistic Regression, SVM, Random Forest.
  - Custom Training: Fine-tuned BERT for optimal results.
- **Why Machine Learning:**
  - Automates sentiment analysis and reply generation.
  - Ensures scalability for large datasets.

Custom Training of Different Models

- **Models Tested**:
  - Logistic Regression: Simple, quick, moderate F1-score (~0.65).
  - SVM: Robust, good performance, slightly lower F1-score (~0.64).
  - Random Forest: Better for imbalanced datasets, highest F1-score (~0.68).
- **Why Move Beyond Traditional Models**:
  - Limited in handling complex language semantics.
  - Struggle with ambiguous or context-dependent comments.
- **Decision to Fine-Tune BERT**:
  - BERT excels at capturing contextual language representations.

Why Fine-Tuned BERT?

- **Advantages of BERT**:
  - Pre-trained on massive datasets, offering state-of-the-art NLP performance.
  - Handles complex language semantics and context effectively.
- **Fine-Tuning Process**:
  - Used a pre-trained BERT model with a classification head.
  - Fine-tuned on cleaned, labeled data (positive, neutral, negative).
  - Trained for 3 epochs with learning rate adjustments for stability.
- **Key Results**:
  - Accuracy: ~91%
  - Macro F1-Score: ~0.71 (Neutral class needs improvement).

ML Challenges and Achievements

- **Challenges**:
  - Handling ambiguous or mixed-sentiment comments.
  - Balancing the dataset for neutral sentiment classification.
- **Achievements**:
  - High precision and recall for positive and negative sentiments.
  - Seamless integration with the web app for real-time analysis.
  - Automated reply generation and issue tracking improved customer interaction.

Flask Application Overview

- **Purpose**:
  - Backend framework for handling API requests and serving web pages.
- **Features Implemented**:
  - Displays product list and comments dynamically.
  - Allows users to add comments through a web interface.
  - Integrates with the database to store user comments, replies, and ticket numbers.

Conclusion

- Sentiment analysis and automated reply generation enhance customer experience.
- Ticket numbers provide efficient issue tracking for negative comments.
- Fine-tuned BERT delivers high accuracy for positive and negative sentiments.
- Neutral sentiment classification remains a key area for improvement.

Future potential work

- **Focus on improving neutral sentiment classification**:
  - Expanding training data with more examples.
  - Incorporating contextual enhancements for ambiguous comments.
- **Deploy the application in the cloud:** for scalability and global access.
- **Explore multilingual support:** to cater to a diverse user base.
- **Integrate advanced analytics:** to gain deeper insights into user feedback.
- **Extend functionality:** to handle product-specific queries or complaints.

THANK-YOU