Customer Feedback Sentiment Analyzer

Arvind Kumar S

1. Abstract

Customer feedback plays a crucial role in understanding business performance and customer satisfaction. Traditional methods of analyzing customer feedback are slow, manual, and often biased. The proposed **Customer Feedback Sentiment Analyser** automates this process using **Natural Language Processing (NLP)** methods.

Developed using **Python (OOP + Modular Programming)**, the system ensures that every functionality is encapsulated into well-defined classes and modules for better maintainability and scalability. The system allows users to input customer feedback through a **tkinter GUI**, stores them securely in **MySQL**, and classifies sentiments into categories (*Positive, Negative, Neutral*). Reports are generated in the form of textual summaries and simple charts to provide actionable insights.

2. Objectives

1. **Automated Sentiment Analysis:** Enable quick categorization of customer sentiments from textual feedback.
2. **Modular OOP Design:** Organize the project using object-oriented design principles (Encapsulation, Inheritance, Polymorphism, Abstraction).
3. **User-Friendly GUI:** Provide an intuitive tkinter-based desktop interface for both administrators and analysts.
4. **Structured Storage:** Maintain customer feedback and analysis results in an organized **MySQL** database.
5. **Decision Support:** Generate visual and statistical reports to help businesses take corrective actions based on customer feedback trends.

3. Module Overview

The system is divided into modules (each implemented as a Python class inside separate modules/packages):

3.1 Feedback Management Module (`feedback.py`)

- Manages feedback CRUD operations.
- Encapsulated in a `FeedbackManager` class.

3.2 Sentiment Analysis Module (`sentiment.py`)

- Handles NLP-related tasks.
- Implemented through a `SentimentAnalyser` class using NLP libraries (`NLTK/TextBlob/spaCy`).

3.3 User Management Module (`user.py`)

- Provides authentication and role-based access.
- Encapsulated in a `UserManager` class.

3.4 Reporting and Visualization Module (`report.py`)

- Generates sentiment distribution charts.
- Implemented by `ReportGenerator` class.

3.5 GUI Module (`gui.py`)

- Provides tkinter-based user interface.
- Encapsulated in `AppGUI` class which interacts with other modules.

4. Architecture Overview

4.1 Architectural Style

- **Frontend (Presentation Layer):** `tkinter GUI` classes.
- **Business Logic Layer:** Python modules (`FeedbackManager`, `SentimentAnalyser`, `UserManager`, `ReportGenerator`).
- **Database Layer:** Encapsulated in `DatabaseHandler` (`db.py`).

4.2 Component Interaction

1. User interacts with **GUI (AppGUI Class)**.
2. GUI invokes respective managers (`FeedbackManager`, `SentimentAnalyser`, etc.).
3. `DatabaseHandler` (using OOP wrapper for MySQL) manages persistence.
4. NLP computations occur inside `SentimentAnalyser`.
5. Analysis results are sent back to GUI for display in both tabular and chart formats.

5. Detailed Module Design

5.1 Feedback Management Module

Class: `FeedbackManager`

- Methods:
  - `add_feedback(customer_name, text)`
  - `update_feedback(feedback_id, text)`
  - `delete_feedback(feedback_id)`
  - `get_feedback(feedback_id)`
  - `list_feedback(limit=100)`

5.2 Sentiment Analysis Module

Class: `SentimentAnalyser`

- Methods:
    - `preprocess(text)` → tokenize, remove stopwords, lemmatize.
    - `analyze_feedback(text)` → returns sentiment category + confidence score.
    - `store_analysis(feedback_id, sentiment, score)`

Uses OOP principle of **Abstraction** (user doesn't need to know the NLP pipeline, only result).

5.3 User Management Module

Class: `UserManager`

- Methods:
    - `register_user(username, password, role)`
    - `authenticate(username, password)`
    - `get_user_role(user_id)`

Passwords are hashed before DB storage.

5.4 Reporting & Visualization Module

Class: `ReportGenerator`

- Methods:
    - `generate_sentiment_summary()` → returns sentiment percentages.
    - `plot_sentiment_distribution()` → tkinter + matplotlib chart.
    - `filter_feedback_by_date(start, end)`

5.5 Database Handler Module

Class: `DatabaseHandler`

- Wrapper class around MySQL operations.
- Methods like `execute_query(query)`, `fetch_results(query)` encapsulate DB logic.

5.6 GUI Module

Class: `AppGUI`

- Methods:
  - `show_login_screen()`
  - `show_dashboard()`
  - `show_feedback_form()`
  - `show_sentiment_results()`
  - `show_reports()`

Handles **event-driven interaction** with all other managers.

6. Database Schema

Entities

- **User**
  - UserID (PK)
  - Username
  - Password (hashed)
  - Role
- **Feedback**
  - FeedbackID (PK)
  - UserID (FK)
  - CustomerName
  - FeedbackText
  - DateSubmitted
- **SentimentResult**
  - ResultID (PK)

- FeedbackID (FK)
- SentimentCategory (Positive/Negative/Neutral)
- ConfidenceScore

7. User Interface Design (tkinter Wireframes)

1. **Login Window:** Authentication form.
2. **Dashboard:** Buttons - "Feedback Management", "Analysis", "Reports".
3. **Feedback Form:** Text input + Save button.
4. **Analysis View:** Shows feedback with classified sentiment table.
5. **Reports View:** Pie chart/Bar chart showing Positive/Negative/Neutral counts.

8. Non-Functional Requirements

- **Performance:** Process 200 feedback entries within ~5 seconds locally.
- **Scalability:** Modular OOP design → can integrate ML models later.
- **Security:** Encrypted passwords & restricted access.
- **Usability:** Simple desktop GUI for admins/analysts.

9. Assumptions & Constraints

- Only **English** text supported (for first release).
- Must run in a **local environment**.
- Python OOP + MySQL → **no web/cloud deployment**.