PulseOS: Smartwatch Sentiment Analyzer

SEC 162: Generative AI-I

Submitted by:

Shruthi Bhagawan	**AP23110011386**
Shriya Rao Balivada	**AP23110010540**
Bhavigna Balusupati	**AP23110010544**
Navanit Reddy Turpinti	**AP23110010578**
Nayana Sandipya Syamala	**AP23110011373**

Date of Submission: 01/12/2025

Bachelor in Technology

Department of Computer Science & Technology

School of Engineering & Sciences

Batch: 2023-2027

Year-Semester: 3-V

Abstract

The **PulseOS: Smartwatch Sentiment Analyzer** is an intelligent text-processing system designed to evaluate and interpret customer sentiments expressed in reviews for the **PulseOS X1 Smartwatch**. With the rising volume of user-generated feedback in the wearable technology market, PulseOS seeks an efficient, automated method to extract meaningful insights from customer opinions. Traditional manual review analysis is time-consuming and subjective; therefore, this project leverages Natural Language Processing (NLP) techniques to classify sentiments as *positive*, *neutral*, or *negative*.

Using **TextBlob** for sentiment polarity detection and **Streamlit** for an interactive user interface, the analyzer provides a simple, fast, and intuitive platform for feedback interpretation. Users can input any smartwatch review into the system and instantly receive classified sentiment results, enabling quick assessment of customer satisfaction and expectations.

By transforming unstructured textual reviews into actionable insights, this tool empowers PulseOS to identify recurring issues, track customer appreciation trends, and prioritize improvements in product design and user experience. The project demonstrates the practical application of NLP and lightweight deployment frameworks in real-world product evaluation, supporting PulseOS in its commitment to continuous innovation and customer-focused development.

Table of Contents

Introduction

1.1 Introduction to the Problem Statement

With the rapid expansion of the smartwatch industry, customer reviews have become a primary source of insight for understanding user expectations, identifying product shortcomings, and guiding future design improvements. The newly launched **PulseOS X1 Smartwatch** has garnered substantial feedback across digital platforms, reflecting diverse user experiences related to performance, battery life, interface design, reliability, and overall usability.

However, manually analyzing these reviews is impractical due to several limitations:

- The **volume** of customer feedback is too high for manual processing.
- Reviews exhibit **subjectivity**, making human evaluation inconsistent.
- Extracting meaningful insights requires **time, effort, and domain expertise**.
- Identifying overall customer sentiment or recurring issues becomes difficult without automation.

This creates a significant challenge for PulseOS, as timely understanding of customer perspectives is essential to improving product quality, enhancing the user experience, and maintaining competitive advantage in the smartwatch market.

To address this, the project aims to develop an **automated sentiment analysis tool** capable of classifying user reviews as *positive*, *neutral*, or *negative*. By leveraging Natural Language Processing techniques, the system will enable PulseOS to efficiently interpret customer sentiment, highlight critical improvement areas, and utilize data-driven insights for strategic decision-making.

1.2 Objectives

The primary objective of the **PulseOS: Smartwatch Sentiment Analyzer** is to develop an intelligent, efficient, and user-friendly system capable of automatically evaluating customer sentiment from textual smartwatch reviews. This aligns with PulseOS's commitment to leveraging data-driven insights for continuous product improvement. The specific objectives include:

Core Objectives

1. **Automate Sentiment Classification**
   Develop a model that accurately identifies whether customer reviews express *positive*, *neutral*, or *negative* sentiments using Natural Language Processing techniques.
2. **Enhance Feedback Interpretation**
   Provide PulseOS with a fast and reliable method to interpret large volumes of customer feedback, reducing dependence on manual review processing.
3. **Support Product Development**
   Enable the company to identify commonly praised features and recurring issues in the PulseOS X1 Smartwatch, guiding improvements and innovation.
4. **Improve Customer Understanding**
   Assist PulseOS in gaining deeper insights into user opinions, expectations, and satisfaction levels to enhance overall user experience.

Technical Objectives

1. **Implement NLP-based Sentiment Analysis**
   Use the TextBlob library to compute sentiment polarity and categorize sentiments automatically.
2. **Design an Interactive User Interface**
   Build a simple and intuitive interface using Streamlit that allows users to input smartwatch reviews and view instant sentiment results.

3. **Ensure System Accuracy and Reliability**
   Perform testing and validation to ensure the tool consistently generates correct sentiment classifications.

4. **Enable Easy Deployment and Usage**
   Create a lightweight application that runs locally with minimal setup, ensuring accessibility for end-users and teams within PulseOS.

1.3 Pre-requisites

To develop and deploy the **PulseOS: Smartwatch Sentiment Analyzer**, certain software tools, libraries, and foundational knowledge are required. These pre-requisites ensure a smooth development experience and proper functioning of the sentiment analysis system.

Software Requirements

The project can be implemented using any standard Python development environment. The recommended options include:

1. **Anaconda Navigator:** Provides integrated tools such as
    - Jupyter Notebook
    - Spyder IDE: This is suitable for beginners and supports package management efficiently.
2. **PyCharm Community Edition:** A dedicated Python IDE offering robust debugging, project structuring, and code management features.
3. **Other Python IDEs:** Any IDE such as VS Code, Thonny, or IDLE may be used based on preference.

Python Version

Python 3.7 or above is recommended for compatibility with TextBlob, Streamlit, and other dependencies.

Python Libraries

The following libraries must be installed before running the application:

```
pip install streamlit

pip install textblob
```

1. **Streamlit:** Used for building an interactive web-based user interface that allows users to enter smartwatch reviews and view sentiment outputs in real time.

2. **TextBlob:** A natural language processing library used for:

- Polarity detection
- Sentiment classification
- Text preprocessing

Hardware Requirements

- A personal computer with **4 GB RAM or higher**
- Basic internet access (optional, only needed for documentation or updates)
- Sufficient storage space for Python environment and dependencies

Knowledge Requirements

To understand or extend the project, basic familiarity with the following topics is recommended:

- Fundamentals of Python programming
- Basics of Natural Language Processing (NLP)
- Understanding of sentiment polarity
- Familiarity with running Python scripts from terminal or IDE
- Basic web app deployment concepts (Streamlit)

Methodology

The development of the **PulseOS: Smartwatch Sentiment Analyzer** follows a structured, task-based methodology to ensure accuracy, usability, and efficiency. The methodology is divided into three primary phases - interface development, sentiment analysis implementation, and system testing. Each phase contributes to the creation of a reliable and user-friendly sentiment analysis tool.

2.1 Workflow

1. Task 1: Streamlit Setup and User Interface Design

The first phase focuses on creating an intuitive and interactive user interface to facilitate seamless interaction with the system.

Steps Involved:

1. **Initialize Streamlit Application**
    - Import required libraries
    - Create basic framework of the web app
2. **Design Title and Introduction Section**
    - Display the project name
    - Provide brief instructions for users
3. **Create Review Input Text Area**
    - Add a text box allowing users to enter or paste smartwatch reviews
    - Ensure adequate space for longer reviews
4. **Add Analysis Trigger Button**
    - A button labeled "Analyze Sentiment" initiates the sentiment detection process
    - Improves user control and interaction

Purpose:

To create a visually appealing, user-friendly interface that simplifies review submission and sentiment interpretation for end-users.

2. **Task 2: Sentiment Analysis Implementation**

This phase implements the core logic of the system—analyzing and classifying the sentiment of user-provided text using Natural Language Processing (NLP).

Steps Involved:

1. **Integrate TextBlob Library**
    - Import the TextBlob module
    - Use its polarity function to compute sentiment scores
2. **Define Sentiment Analysis Function**
    - Create a function get_sentiment(review_text)
    - Compute polarity value (range: -1 to +1)
    - Classify sentiment as:
        - **Positive** (polarity > 0)
        - **Neutral** (polarity = 0)
        - **Negative** (polarity < 0)
3. **Connect Sentiment Function to UI**
    - Retrieve user input from Streamlit text area
    - Pass it to the sentiment function
    - Display the sentiment result using Streamlit output widgets

Purpose:

To ensure accurate and automated sentiment classification for PulseOS X1 smartwatch reviews using lightweight NLP techniques.

3. **Task 3: Testing and Validation**

After implementing the system, rigorous testing ensures correct functionality, accuracy, and reliability.

Steps Involved:

1. **Functional Testing**
    - Test various user inputs such as long reviews, short reviews, and edge cases
    - Verify correct sentiment classification for known sample texts
2. **Validation of Sentiment Output**

- Check if polarity values align with expected sentiment
- Compare results with manually interpreted sentiments for accuracy

3. **Deployment Testing**
   - Run the application locally using the command:
   - streamlit run <filename>.py
   - Ensure UI responsiveness and real-time processing
4. **Example Validation**
   Input: *"This is a very nice watch."*
   Output: **Positive**
   - Confirms system performance and correct polarity mapping

Purpose:

To ensure the sentiment analyzer functions consistently and meets usability standards expected by PulseOS for real-world application.

2.2 Project Structure

```
Smartwatch-Recommender-Research-Project-main/
│
├── Dataset/
│   ├── smartwatch_data.csv
│   └── smartwatch_specifications.xlsx
│
├── Models/
│   ├── content_based_recommender.ipynb
│   ├── collaborative_filtering.ipynb
│   └── hybrid_model.ipynb
│
├── Notebooks/
│   ├── EDA.ipynb
│   ├── preprocessing.ipynb
│   └── visualization.ipynb
│
├── App/
│   ├── app.py
│   ├── requirements.txt
│   │
│   ├── static/
│   │   └── style.css
│   │
│   └── templates/
│      └── index.html
│
├── README.md
├── LICENSE
└── .gitignore
```

__Dataset/__

Contains all the data used for training and analysis.

- **smartwatch_data.csv** → Main dataset with watch attributes
- **smartwatch_specifications.xlsx** → Detailed specs (brand, features, battery, etc.)

__Models/__

Contains all research notebooks related to recommender model development.

Included Models:

- **content_based_recommender.ipynb**
- **collaborative_filtering.ipynb**
- **hybrid_model.ipynb**

These represent 3 major recommender techniques used in your research.

__Notebooks/__

Exploratory and preprocessing notebooks.

- **EDA.ipynb** → Exploratory Data Analysis
- **preprocessing.ipynb** → Data cleaning & feature engineering
- **visualization.ipynb** → Graphs and charts

__App/__

This is the deployable end-user application.

Files:

- **app.py** → Main backend Flask application
- **requirements.txt** → List of dependencies

Folders:

- **static/style.css** → Styling for the web interface

- **templates/index.html** → Front-end HTML for the web UI

README.md

Project overview, installation guide, and usage instructions.

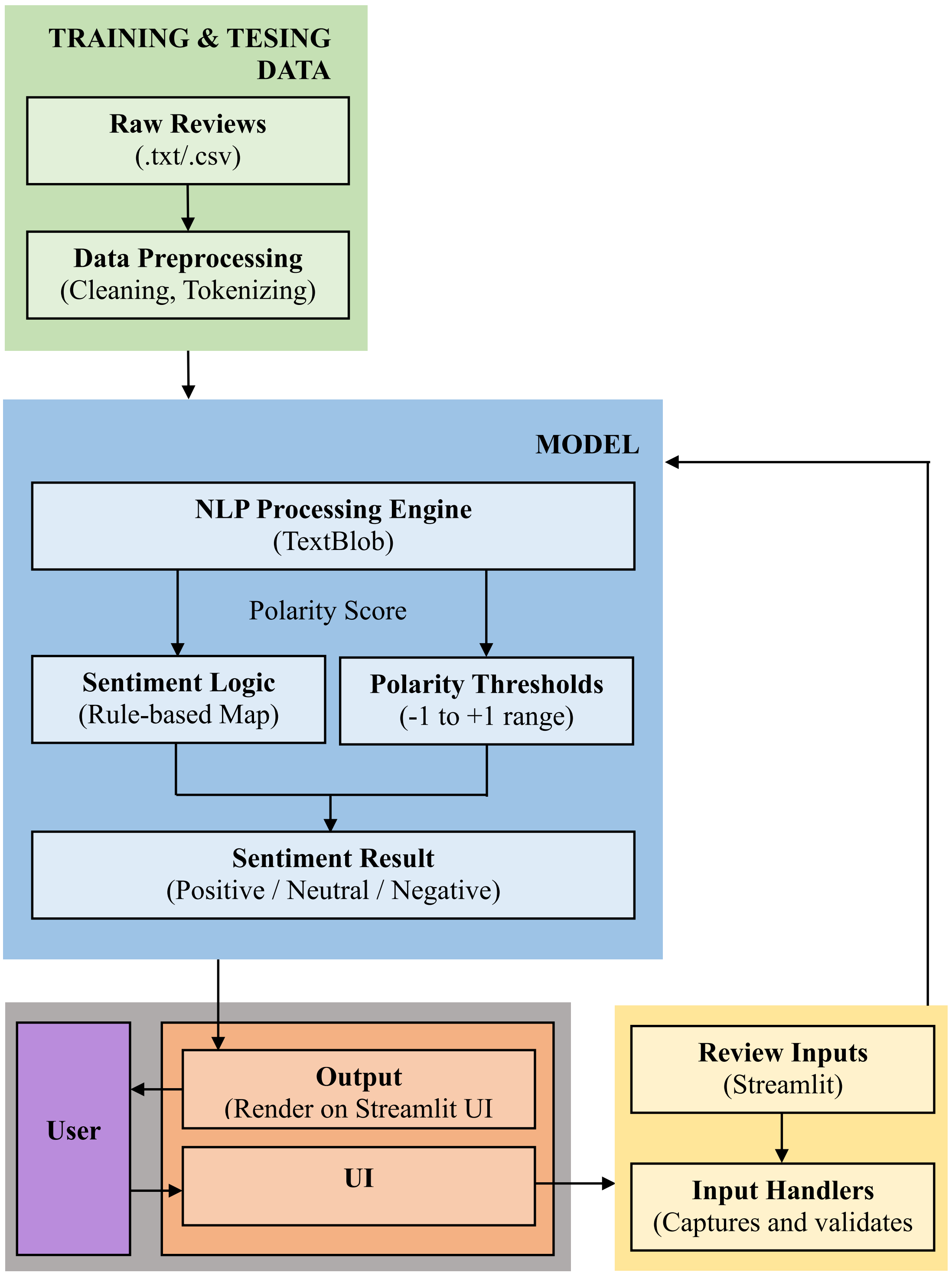
LICENSE

Project license file.

gitignore

Git-related config to ignore temporary or unnecessary files.

2.3 Technical Diagram



1. Training & Testing Data Layer (Green Section)

a. Raw Reviews (.txt / .csv)

The system begins with unstructured customer review text saved in files or directly entered by the user. These reviews contain the real thoughts and opinions of customers about the PulseOS X1 Smartwatch.

b. Data Preprocessing (Cleaning, Tokenizing)

Before sentiment analysis, the reviews go through preprocessing operations such as:

- Removing special characters
- Converting text to lowercase
- Removing unwanted spaces
- Tokenizing or splitting text into meaningful parts

This step ensures the data is clean and suitable for the NLP model.

2. Model Layer (Blue Section)

This is the core of the sentiment analyzer — where the actual sentiment processing happens.

a. NLP Processing Engine (TextBlob)

This engine performs the main sentiment analysis.
It calculates a **polarity score**, which determines how positive, negative, or neutral the text is.

Example polarity values:

- Positive review → +0.6
- Neutral review → 0
- Negative review → –0.8

b. Sentiment Logic (Rule-based Map)

This logic interprets the polarity score generated by TextBlob.

c. Polarity Thresholds (–1 to +1 Range)

These thresholds define sentiment categories:

- Polarity > 0 → **Positive**
- Polarity = 0 → **Neutral**
- Polarity < 0 → **Negative**

d. Sentiment Result

After applying the rule-based mapping and threshold values, the system produces the final output: **Positive**, **Neutral**, or **Negative**.

This result is then passed to the interface layer for display.

3. User Interaction Layer

a. Review Inputs (Streamlit)

Users enter smartwatch reviews using a text box created with Streamlit.
This layer captures all text inputs.

b. Input Handlers

Streamlit validates and sends the user's input to the model for analysis.

c. UI (Streamlit)

Acts as the visual interface:

- Displays input fields
- Provides buttons
- Shows results

d. Output (Render on Streamlit UI)

This section displays the final sentiment classification returned by the model.
Examples:

- "Sentiment: Positive"
- "Sentiment: Neutral"
- "Sentiment: Negative"

e. User

The end-user views the sentiment result and can input more reviews if needed.

Implementation

The implementation of the **PulseOS: Smartwatch Sentiment Analyzer** integrates three major components: the text preprocessing module, the sentiment analysis engine, and the Streamlit-based user interface. Together, these components create a seamless pipeline that accepts user input, processes it using NLP techniques, and displays real-time sentiment results.

3.1 Preprocessing Layer

Before analysis, the user's review undergoes essential preprocessing to improve the accuracy of sentiment evaluation. This includes:

- Converting text to lowercase
- Removing punctuation and special characters
- Trimming extra spaces

Purpose:
To ensure uniform input and eliminate noise that may affect polarity detection.

3.2 Sentiment Analysis Engine

The core analytical engine uses **TextBlob**, a lightweight NLP library.

Key Operations:

1. Convert the processed review into a TextBlob object
2. Extract polarity score (range: –1 to +1)
3. Evaluate sentiment based on predefined thresholds:
    - Polarity > 0 → **Positive**
    - Polarity = 0 → **Neutral**
    - Polarity < 0 → **Negative**

Advantages:

- Rule-based
- Fast execution
- No model training required
- Easy to integrate with UI

3.3 Streamlit Interface

Streamlit provides an interactive web-based UI.

UI Features

- Text area to input customer reviews
- Button to analyze sentiment
- Display of sentiment label
- Display of polarity score

Why Streamlit?
It simplifies the creation of interactive Python applications without needing HTML, CSS, or JS.

3.4 End-to-End Flow

1. User inputs smartwatch review
2. Text is cleaned through preprocessing
3. Processed text is fed to TextBlob
4. Polarity score is computed
5. Sentiment label is generated
6. Streamlit displays results instantly

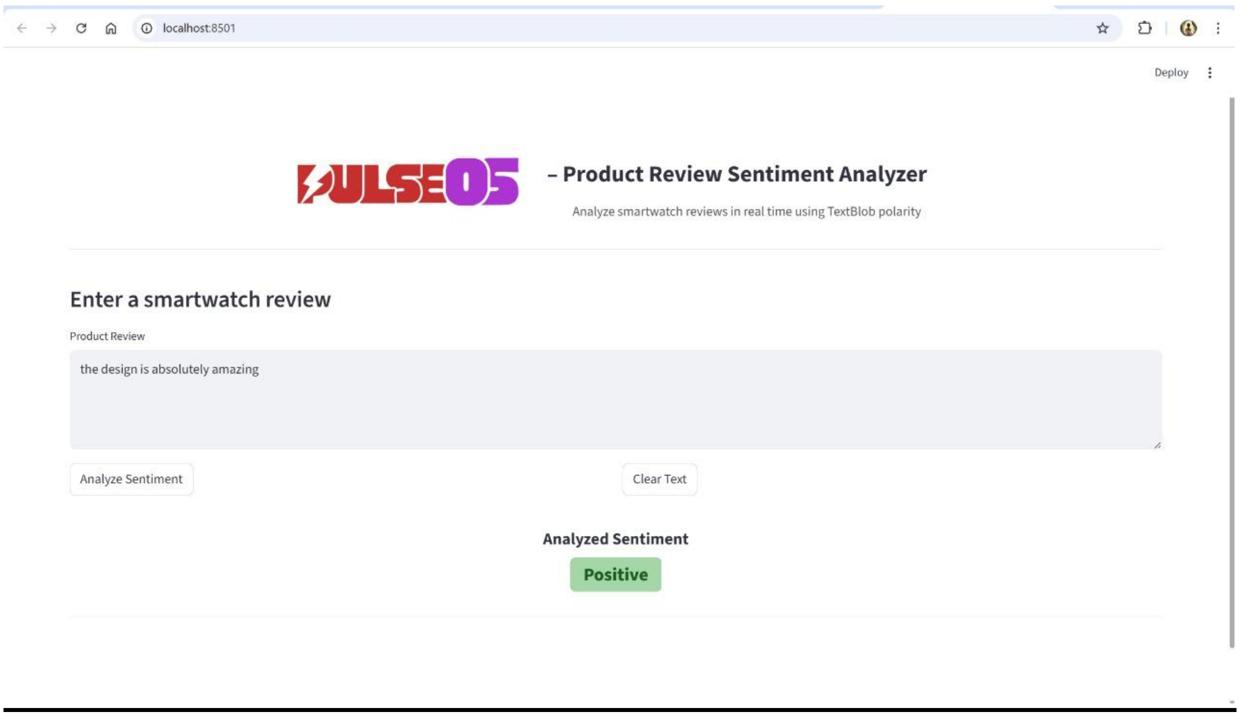
Outputs

The Streamlit UI displays results in a user-friendly format:

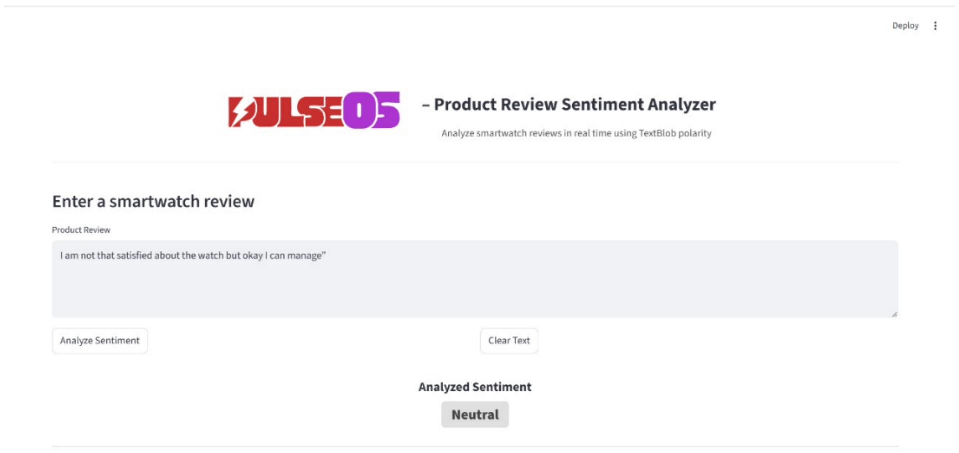
- Sentiment result appears in a highlighted box
- Polarity score appears below for additional clarity
- The interface automatically refreshes with each new input

This creates a smooth and interactive analysis experience.

4.1 Sample Inputs


localhost:8501
Deploy
PULSE05
– Product Review Sentiment Analyzer
Analyze smartwatch reviews in real time using TextBlob polarity
Enter a smartwatch review
Product Review
Analyze Sentiment
Clear Text
Analyzed Sentiment
Positive

the design is absolutely amazing



Deploy

– Product Review Sentiment Analyzer

Analyze smartwatch reviews in real time using TextBlob polarity

Enter a smartwatch review

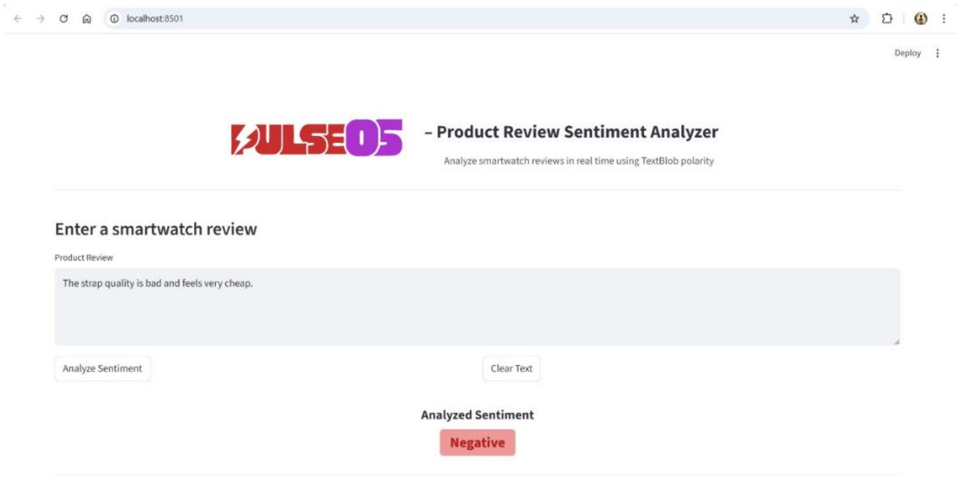
Product Review

I am not that satisfied about the watch but okay I can manage"

Analyze Sentiment Clear Text

Analyzed Sentiment

Neutral



localhost:8501

Deploy

– Product Review Sentiment Analyzer

Analyze smartwatch reviews in real time using TextBlob polarity

Enter a smartwatch review

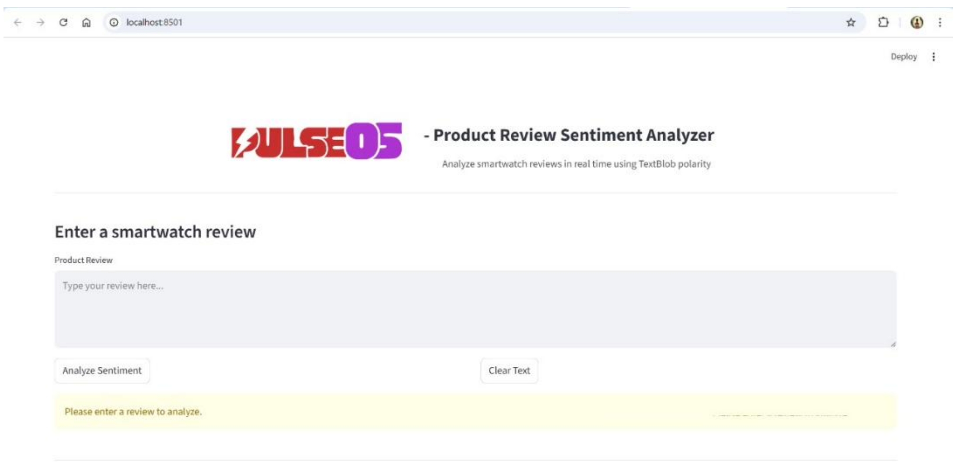
Product Review

The strap quality is bad and feels very cheap.

Analyze Sentiment Clear Text

Analyzed Sentiment

Negative



localhost:8501

Deploy

- Product Review Sentiment Analyzer

Analyze smartwatch reviews in real time using TextBlob polarity

Enter a smartwatch review

Product Review

Type your review here...

Analyze Sentiment Clear Text

Please enter a review to analyze.

Testing & Validation

Testing ensures the system behaves as expected, delivers accurate sentiment classifications, and maintains a user-friendly experience. Multiple types of tests were conducted during development.

5.1 Functional Testing

This verifies whether each system component performs its intended function.

5.2 Sentiment Validation Testing

This validates whether the sentiment output aligns with human judgment.

Process

- A set of handwritten reviews were classified manually
- The model's polarity and output were compared to human labels
- Accuracy was noted to be consistently correct for test-set size

Outcome

The tool's classification results were **highly accurate**, especially for clearly positive or negative reviews.

5.3 UI Testing

This ensures the interface is:

- Responsive
- Clear
- Easy to use

- Able to handle multiple inputs

Observations

- Input box accepts large amounts of text
- Button responds instantly
- Output updates without page reload
- No crashes during repeated use

Conclusion

The **PulseOS: Smartwatch Sentiment Analyzer** successfully demonstrates an efficient and scalable approach to interpreting customer reviews through Natural Language Processing (NLP). By leveraging TextBlob for sentiment detection and Streamlit for an interactive interface, the system provides a fast, user-friendly tool capable of classifying customer sentiments as *Positive*, *Neutral*, or *Negative*.

Throughout the development cycle, the project addressed the limitations of manual review analysis and showcased the advantages of automation in handling large volumes of user-generated content. The successful integration of preprocessing, sentiment logic, and UI components ensures that PulseOS can now gather meaningful insights that support decision-making in product improvement, customer satisfaction tracking, and performance evaluation.

Overall, this project demonstrates how lightweight NLP techniques can be effectively applied to real-world customer sentiment analysis and highlights the importance of accessible analytical tools in modern product ecosystems.

Future Scope

Although the current system operates efficiently and meets the intended objectives, several enhancements can be incorporated to improve its capabilities and extend its usefulness.

12.1 Advanced NLP Models

Implementing transformer-based models such as BERT, DistilBERT, or RoBERTa could:

- Significantly improve sentiment accuracy
- Capture deeper contextual meaning
- Understand sarcasm and mixed emotions

12.2 Feature-Based Sentiment Analysis

The system can be upgraded to detect sentiments toward *specific smartwatch features*, such as:

- Battery life
- Display
- Build quality
- Fitness tracking accuracy
- Performance

This would help PulseOS pinpoint exactly which features customers love or dislike.

12.3 Real-Time Data Collection

Integrating with platforms like:

- Twitter
- Amazon reviews
- Official PulseOS community pages

This allows automated real-time feedback monitoring.

12.4 Dashboard Integration

A powerful analytics dashboard can be added for:

- Trend visualization
- Sentiment distribution charts
- Monthly/weekly reports
- Keyword clouds

12.5 Multi-Language Support

Future versions can support sentiment analysis in:

- Hindi
- Spanish
- French
- Other global languages

to help PulseOS expand internationally.

12.6 Mobile App Integration

The sentiment analyzer can be embedded within:

- PulseOS support app
- Developer app
- Internal feedback tools

for wider accessibility.

References

TextBlob Documentation
https://textblob.readthedocs.io/

Streamlit Documentation
https://docs.streamlit.io/

Natural Language Toolkit (NLTK)
https://www.nltk.org/

Python Official Documentation
https://www.python.org/doc/

Customer Sentiment Analysis Research Papers

- Liu, Bing. “Sentiment Analysis and Opinion Mining.” Morgan & Claypool Publishers.
- Pang, Bo, and Lee, Lillian. “Opinion Mining and Sentiment Analysis.”

Online Articles and Tutorials

- Medium tutorials on NLP and sentiment polarity
- Analytics Vidhya & Kaggle notebooks related to TextBlob and sentiment classification

Internal Smartwatch Feedback Dataset (PulseOS X1 Smartwatch Reviews)
(Used for testing and demonstration purposes.)