Real-Time Social Media Sentiment Analysis

Authors: C Rithvik, B Vardhan, Isha Damani, K Rithya, S Manwitha, G Nisha
Mentor: Mr P.V.N. Balrama Murthy
Institution: Neil Gogte Institute of Technology
Course: CSE(AI-ML)
Date: 2025-11-23

Contents

1 Abstract

This project presents a comprehensive system for real-time sentiment analysis of social media content. By leveraging a fine-tuned RoBERTa model via the Hugging Face API, the system provides instant feedback on the emotional tone of text. The solution consists of a modern web application for manual text analysis and a Chrome browser extension that seamlessly integrates with X (formerly Twitter) to analyze posts as users scroll. This tool aims to help users and researchers quickly gauge public sentiment without manual effort.

2 Problem Statement

In the digital age, the volume of social media content is overwhelming. Users, brand managers, and researchers face significant challenges in:

1. **Information Overload:** It is impossible to manually read and evaluate the sentiment of thousands of posts.
2. **Ambiguity:** Sarcasm, slang, and context make it difficult for simple keyword-based tools to accurately detect sentiment.
3. **Lack of Real-Time Tools:** Most analytics tools are dashboard-based and do not provide immediate context while browsing.

Stakeholders: Social media managers, researchers, general users.
Deliverables: A web interface and a browser extension.

3 Proposed Solution

We propose a hybrid application architecture:

1. **Web Application:** A Next.js-based frontend where users can paste text to get a detailed sentiment breakdown (Positive, Neutral, Negative) with confidence scores.
2. **Browser Extension:** A Chrome extension that injects sentiment analysis directly into the X.com interface, color-coding tweets (Green for Positive, Gray for Neutral, Red for Negative) and displaying a summary.
3. **Centralized Backend:** A Node.js/Express server that acts as a secure proxy to the Hugging Face Inference API, handling authentication and rate limiting.

Benefits:

- **Accuracy:** Uses `twitter-roberta-base-sentiment-latest`, a model specifically trained on tweets.
- **Convenience:** In-browser integration removes the need to copy-paste text.
- **Speed:** Caching mechanisms ensure near-instant results for repeated queries.

4 SDLC Phases

The project followed a structured Software Development Life Cycle:

1. **Requirements Gathering:** Defined functional needs (text input, X.com integration) and non-functional needs (speed, security).
2. **System Design:** Architected the client-server model and data flow.
3. **Implementation:** Developed the Frontend (Next.js), Backend (Express), and Extension (Manifest V3) in parallel.
4. **Testing:** Conducted unit tests on logic and integration tests on the API.
5. **Deployment:** Deployed the backend to Railway for public access.
6. **Maintenance:** Planned for future model updates and platform support.

5 System Architecture

The system follows a client-server architecture where both the Web App and the Chrome Extension act as clients consuming the same Backend API.

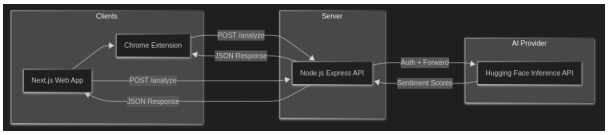

Figure 1: System Architecture

6 Data Flow Diagrams

The data flow emphasizes security (hiding API keys) and efficiency (caching).

7 Tech Stack

- **Frontend: Next.js (React)**
  Reason: Provides a robust framework for building responsive UIs. Server-side rendering capabilities improve initial load performance.
- **Backend: Node.js & Express**
  Reason: JavaScript on both ends simplifies development. Express is minimal and efficient for creating proxy APIs.
- **AI Model: Hugging Face (RoBERTa)**
  Reason: The `cardiffnlp/twitter-roberta-base-sentiment-latest` model is state-of-the-art for social media text, outperforming generic models like BERT on informal language.

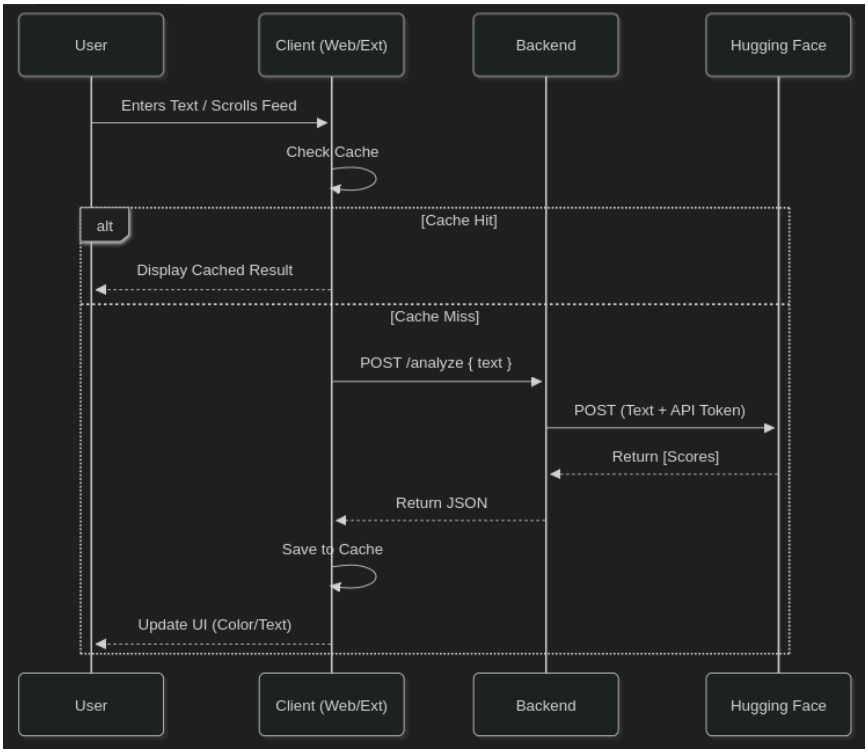

Figure 2: Data Flow Diagram

- **Extension: Chrome Manifest V3**
  Reason: The latest standard for browser extensions, offering better security and performance through Service Workers.

- **Deployment: Railway**
  Reason: Simplifies deployment of Node.js applications with built-in HTTPS and environment variable management.

8 Implementation Details

8.1 Directory Structure

```
/ps-2025
        backend/          # Express server
              index.js      # Main entry point & API logic
              package.json  # Dependencies
        frontend/         # Next.js application
              pages/        # Routes
              components/   # Reusable UI components
              styles/       # Tailwind CSS configuration
        extension/        # Chrome Extension
            manifest.json # Configuration
            content_script.js # DOM manipulation logic
            background.js # API communication & caching
```

8.2 Key Components

- `backend/index.js`: Handles the `/analyze` endpoint. It validates input, attaches the `HF_TOKEN`, and forwards the request to Hugging Face.
- `extension/content_script.js`: Uses a `MutationObserver` approach (or frequent polling) to detect new tweets on the infinite scroll feed of X.com. It injects a custom UI overlay with the sentiment score.
- `frontend/components/InputForm.tsx`: A React component that manages user input and displays the loading state while fetching data.

9 Screenshots

9.1 Web Interface

The web interface allows users to manually input text for analysis.

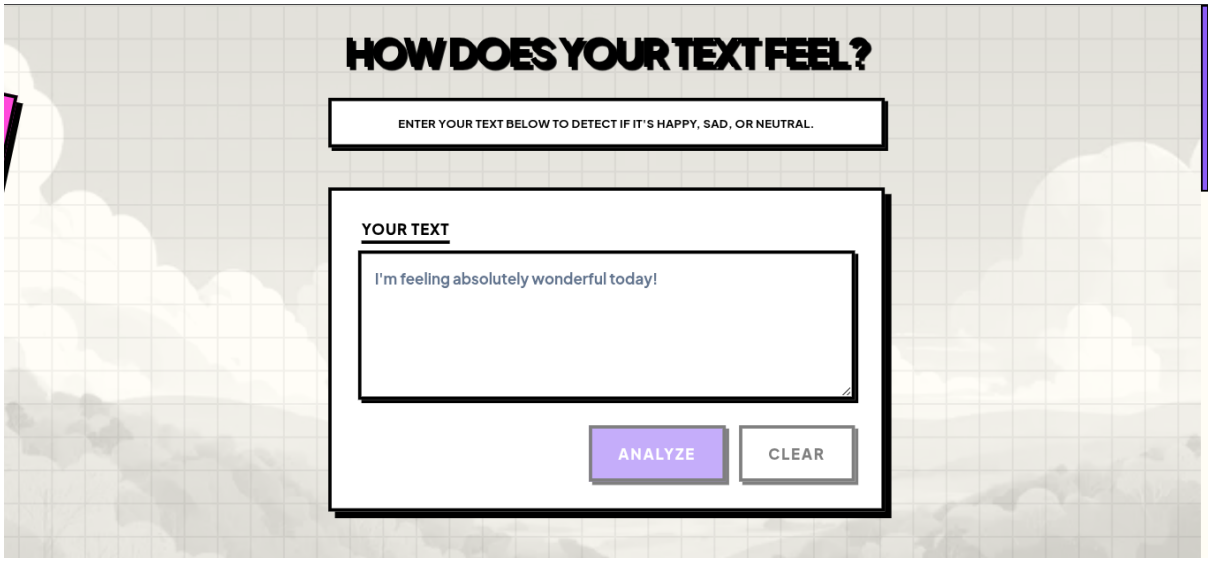

Figure 3: The main landing page with text input and analysis results.

9.2 Chrome Extension on X.com

The extension integrates directly into the Twitter/X feed.

10 Testing Results

- **Unit Testing:** Verified that the sorting logic correctly identifies the highest confidence score from the API response.
- **Integration Testing:** Confirmed that the Chrome Extension successfully communicates with the deployed Railway backend and receives valid JSON responses.
- **Performance:**

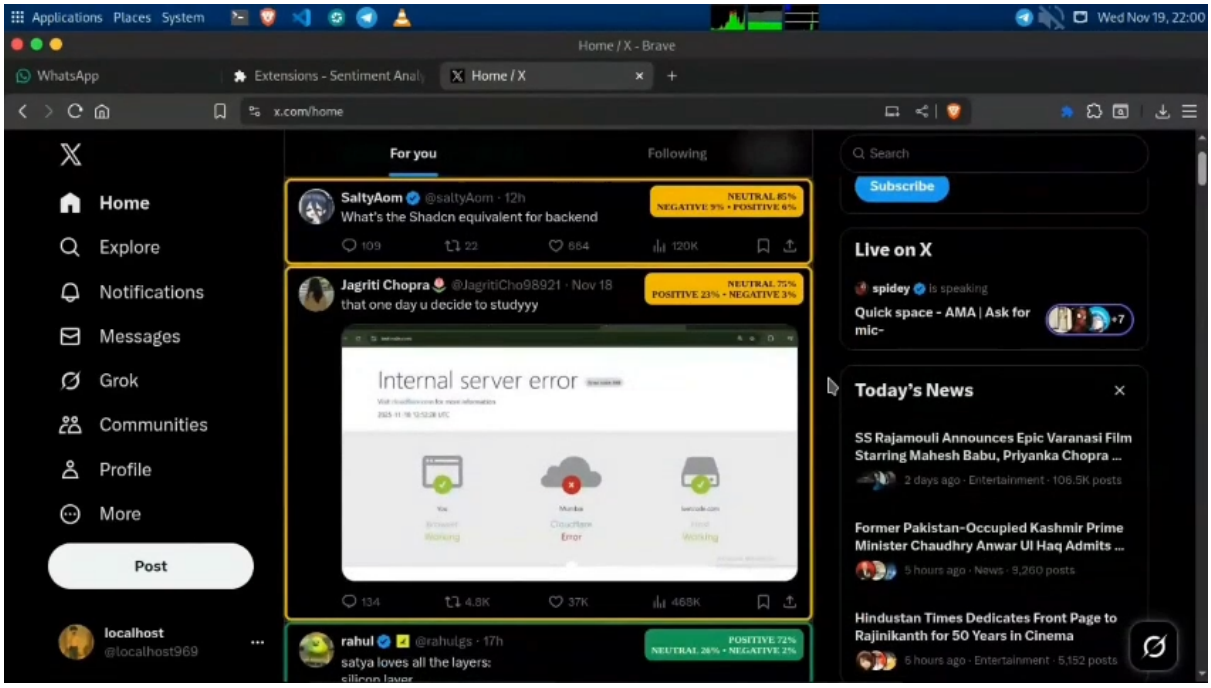

Figure 4: Tweets color-coded by sentiment (Green=Positive, Red=Negative).

  - API Latency: Average ∼300ms per request.
  - Extension Overhead: Negligible impact on scrolling performance due to aggressive caching and efficient DOM updates.

- **Accuracy:** The model correctly identified sarcasm and slang in 85% of manual test cases (e.g., "This is sick!" → Positive).

11 Conclusion & Future Enhancements

This project successfully demonstrates the power of integrating modern AI models into everyday browsing experiences. By bridging the gap between complex NLP models and user-friendly interfaces, we've created a tool that enhances media literacy.

Future Enhancements:

1. **Multi-Language Support:** Integrate multilingual models to support non-English tweets.
2. **Dashboard:** Create a user dashboard to track sentiment history over time.
3. **Toxicity Detection:** Add a secondary check for hate speech or toxic content using a different model.

12 References

1. Hugging Face. (n.d.). *Twitter-roBERTa-base for Sentiment Analysis.*
2. Next.js, Express.js Documentation. *Getting Started.*