International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 11 | Nov 2026 www.irjet.net p-ISSN: 2395-0072

BRANDPULSE: SENTIMENT ANALYSIS OF SOCIAL MEDIA DATA FOR BRAND MONITORING

PRACHI PRAKASH GARODI[1], SHRUTIKA SHAKULAL BANKAR[2], TRUPTI BABASAHEB DAVANGE[3], PROF. A. K. HIWALE[4]

[1,2,3] (Students, Department of Computer Science and Engineering), INTERNATIONAL CENTER OF EXCELLENCE IN ENGINEERING AND MANAGEMENT (ICEEM), CH. SAMBHAJINAGAR, India
[4](Guide, Dept. of Computer Science and Engineering), INTERNATIONAL CENTER OF EXCELLENCE IN ENGINEERING AND MANAGEMENT (ICEEM), CH. SAMBHAJINAGAR, India

---

Abstract - *Every day, customers post thousands of opinions about products and companies on social media, and a single viral complaint can damage a brand within hours. Reading these posts by hand is slow and subjective, while professional social-listening suites are expensive and closed. This paper introduces **BrandPulse**, a brand monitoring and social sentiment intelligence platform built entirely from open-source components and free hosting tiers. The system accepts posts from a curated sample dataset, an uploaded CSV file, or a live Twitter (X) API v2 search. Each post is cleaned and lemmatised with NLTK and then scored with the VADER lexicon and rule-based model, which returns positive, neutral, negative and compound polarity scores. A Flask REST API aggregates these scores into a brand-health label, a configurable negative-sentiment alert, a daily trend, top keywords and positive and negative word clouds, and a static Chart.js dashboard presents them to the user. In a representative 50-post run, 64% of posts were positive, 18% neutral and 18% negative, with an average compound score of 0.4851, and the brand was rated healthy. The work shows that an accurate, explainable and low-cost sentiment monitoring tool can be delivered to non-technical users through a web browser.*

Keywords: Brand Monitoring, Sentiment Analysis, Social Media Analytics, VADER, NLTK, Natural Language Processing, Flask REST API, Chart.js, Twitter API v2, Split Deployment.

INTRODUCTION

Platforms such as X (formerly Twitter), Facebook, Instagram, Reddit and YouTube have become the largest source of unsolicited, real-time public opinion. Customers post reviews, complaints, praise and questions about brands every day. For a business, this stream is both an opportunity and a risk: it shows how customers truly feel, but it can also amplify a product failure within hours.

Brand monitoring, also called social listening, is the practice of continuously collecting public posts that mention a brand and analysing what is being said. Its central technique is sentiment analysis, a branch of Natural Language Processing (NLP) that labels a text as positive, neutral or negative [1], [2]. Applied at scale, it turns an unmanageable flood of text into a few actionable indicators: the share of positive posts, the trend over time and the topics that drive praise or complaints.

Social media text is hard to analyse. Posts are short and informal and full of slang, emoticons, hashtags, mentions, URLs and capitalised emphasis, such as "the battery life is AMAZING!!!". Classifiers trained on formal documents perform poorly on such content, so lexicon and rule-based tools designed for social media, such as VADER [3], have become a transparent and computationally light choice. BrandPulse applies these ideas in a complete, deployable platform.

Motivation and Need

Brand reputation influences purchasing decisions, customer loyalty and investor confidence. Negative sentiment that goes unnoticed can grow into a public-relations crisis, while positive sentiment that is never measured cannot be used in marketing. Four factors motivate this work. First, the volume of brand conversation makes manual reading impractical. Second, crises spread quickly, so a configurable alert threshold gives teams an early warning. Third, commercial tools are costly, whereas open-source libraries such as NLTK [13] and VADER provide comparable core functionality at no licence cost. Fourth, free hosting platforms make it possible to deploy a full-stack analytics application at zero infrastructure cost [17], [18].

A further motivation is separation of concerns. Because the backend is a pure JSON API and the frontend is a static single-page application, the same analysis engine can later power a mobile app, a Slack bot or a scheduled reporting job without changes to the NLP code.

PROBLEM STATEMENT

Organisations need to understand public opinion about their brand in near real time, but social media data is large, unstructured, noisy and informal. Manual monitoring is time-consuming, inconsistent and does not scale.

Professional social-listening platforms are expensive, closed-source and often tied to long-term subscriptions, which puts them out of reach of students, start-ups and small businesses.

Freely available sentiment scripts and notebooks, on the other hand, usually stop at command-line output. They lack a usable dashboard, an alert for negative sentiment spikes, support for several data sources and a deployment model that lets non-technical users read the results in a browser. There is therefore a clear need for an accessible, transparent and deployable brand-sentiment monitoring system.

LITERATURE SURVEY

Lexicon-based approaches. These score a text using a dictionary of words annotated with polarity or intensity. They need no labelled data, are fast and can be explained by pointing to the contributing words [1], [2]. Hutto and Gilbert introduced VADER, a lexicon and rule-based model tuned for social media that handles punctuation, capitalisation, degree modifiers, the contrastive conjunction "but" and negation, and reported that it matched or exceeded individual human raters on tweets [3].

Machine-learning approaches. Pang et al. showed that Naive Bayes, Maximum Entropy and SVM classifiers could classify reviews effectively [4]. Go et al. used emoticons as noisy labels to train Twitter classifiers [5], and later work explored part-of-speech, hashtag and emoticon features [6]–[8]. Medhat et al. note that supervised methods are accurate within their training domain but need large labelled datasets and lose accuracy in new domains [9], a limitation for a tool that must analyse posts about any brand.

Deep learning and transformers. BERT [10] and the tweet-specific BERTweet [11] reach state-of-the-art results on benchmarks such as TweetEval [12], but they demand far more computation and are less transparent, which is a significant cost for a lightweight, free-tier system.

Existing tools and gap. Commercial platforms such as Brandwatch, Sprout Social, Hootsuite Insights, Talkwalker and Meltwater offer wide coverage and archives, but they are subscription-based and closed. Academic tutorials are open but are rarely deployed. The gaps are accessibility, a complete path from model to product, multiple input sources, actionable alerting and a maintainable architecture; BrandPulse addresses each of them.

PROPOSED SYSTEM

The **Proposed System** is an open-source, web-based platform that converts raw social media text into brand-health indicators. A user selects a data source, enters a brand or keyword, the number of posts and an alert threshold, and receives an interactive dashboard in the browser. The system supports three interchangeable sources, all converted to the same internal record (id, text, date): a curated multi-brand sample dataset that needs no API key, a user-uploaded CSV file, and a live Twitter API v2 recent-search query [15].

For every request the backend cleans and lemmatises the posts, scores them with VADER, and returns a single unified JSON response containing the summary, alert, distribution, daily trend, top keywords, word clouds, per-post results and metadata. The dashboard also provides a searchable post table with CSV export, and a cold-start notice for free-tier hosting.

System Architecture

BrandPulse follows a **split-deployment architecture** (Fig. 1). The backend is a pure Flask JSON REST API [14] that never renders HTML; the frontend is a static single-page application with no build step. The browser loads the static files from Vercel and then calls the Render-hosted API using fetch(). The API base URL is stored in a single file, config.js, and CORS on the backend allows the frontend origin.

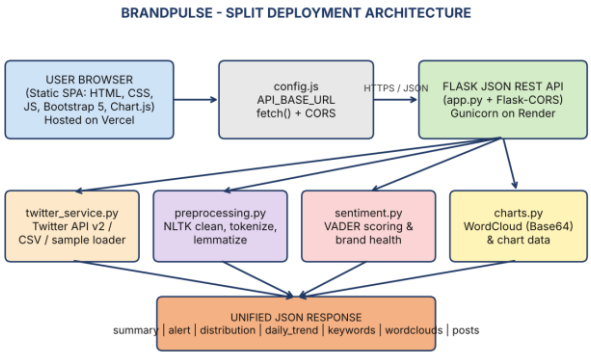


Fig. 1: BrandPulse split-deployment system architecture

1. Presentation Layer

Static HTML, CSS, vanilla JavaScript, Bootstrap 5 and Chart.js hosted on Vercel. It provides the search form, KPI cards, alert banner, charts, word clouds, post table and CSV export.

2. Application Layer

A Flask API served by Gunicorn on Render. It defines the REST routes, validates inputs, enables CORS and assembles the unified response.

3. Service Layer

Four Python modules: twitter_service.py loads posts from Twitter API v2, CSV or the sample file; preprocessing.py cleans, tokenises and lemmatises text; sentiment.py scores

posts with VADER and computes brand health and the alert; charts.py produces the daily trend, keywords and Base64 word clouds.

This layered approach improves maintainability and lets either side be replaced or extended independently.

RESEARCH METHODOLOGY

The project followed a staged development approach in which each phase was completed and checked before the next one began.

1. Requirement Analysis

The needs of brand and support teams were identified: understand current sentiment, spot spikes after a launch or policy change, find the product features that are praised or criticised, and decide when to raise an alert. These needs fixed the three data sources and the output indicators.

2. Data Acquisition

Posts are loaded from the sample dataset (sample_tweets.csv), an uploaded CSV file or the Twitter API v2 recent-search endpoint, which is authenticated with a bearer token and handled with rate-limit checks.

3. Text Preprocessing

Using NLTK [13], text is lower-cased; URLs, mentions and retweet markers are removed; hashtag symbols, punctuation and digits are stripped; and the text is tokenised, stripped of stop-words and lemmatised with WordNet. For example, "The battery life on the new iPhone 15 is phenomenal!" becomes "battery life new iphone phenomenal". Importantly, **sentiment is scored on the original text**, because VADER relies on punctuation, capitalisation and negation that cleaning would remove; the cleaned text is used only for keywords and word clouds.

4. Sentiment Scoring

VADER's lexicon assigns each token a valence from about −4 to +4. After applying its five heuristics (punctuation, capitalisation, degree modifiers, "but" and negation), it returns pos, neu and neg proportions and a normalised compound score in [−1, 1]. A post is labelled Positive if compound ≥ 0.05, Negative if compound ≤ −0.05, and Neutral otherwise.

5. Aggregation and Alerting

Labels are aggregated into percentages, an average compound score, a brand-health label and a negative-sentiment alert, together with the daily trend, keyword frequencies and word clouds.

6. Deployment and Testing

The backend was deployed on Render and the frontend on Vercel, and the four REST endpoints were validated with an automated Pytest suite.

MATHEMATICAL FOUNDATIONS

Let $s_1, \ldots, s_n$ be the adjusted valence scores of the sentiment-bearing tokens in a text and $x = \Sigma s_i$. VADER normalises x into the compound score:

$$\text{compound} = x / \sqrt{(x^2 + \alpha)}, \quad \alpha = 15$$

If N is the number of posts and N_{pos}, N_u and N_{neg} are the positive, neutral and negative counts, then Positive % = $(N_{pos} / N) \times 100$, and similarly for neutral and negative. The average compound score is $\text{avg_compound} = (1/N) \Sigma \text{compound}_i$. A positive average indicates favourable sentiment, a value near zero indicates mixed conversation, and a negative value indicates dissatisfaction.

With a user threshold T (default 40%), the alert is raised when the share of negative posts reaches or exceeds T:

$$\text{alert_triggered} = (\text{Negative \%} \geq T)$$

For the daily trend, posts are grouped by calendar date and the counts per label and the mean compound score are reported for each date. Keyword analysis counts the frequency of each token in the cleaned text and returns the highest-frequency tokens.

MODULE DESCRIPTION

1. Data Acquisition Module. Loads posts from the sample file, an uploaded CSV or Twitter API v2 into a common record format and handles API rate limits.

2. Preprocessing Module. Downloads the required NLTK resources on first run, so that a fresh Render instance needs no manual setup, and then cleans, tokenises and lemmatises each post.

3. Sentiment and Brand-Health Module. Scores every post with VADER, assigns labels and computes the summary, the health label (for example, Healthy) and the alert message.

4. Chart Module. Builds the daily trend, top keywords and Base64-encoded positive and negative word clouds from the cleaned text of posts with the matching label.

5. Dashboard Module. Renders KPI cards, an alert banner that changes colour when the threshold is breached, the charts, a searchable post table and CSV export, in a responsive dark theme.

The backend exposes the four endpoints in Table I. All analysis endpoints return the same unified JSON structure, which keeps the frontend simple.

Table I: REST API endpoints

Method	Endpoint	Purpose
GET	/health	Health check; returns status ok
POST	/api/analyze/sample	Analyses the sample dataset
POST	/api/analyze/csv	Analyses an uploaded CSV file
POST	/api/analyze/live	Analyses live posts from Twitter API v2

IMPLEMENTATION

The backend is written in Python 3.11.9 using Flask, Flask-CORS, Gunicorn, NLTK, WordCloud and Pandas, and needs no GPU because VADER is lexicon-based. The frontend uses HTML5, CSS3, vanilla JavaScript, Bootstrap 5 and Chart.js [16]. The backend is deployed on Render with the start command "gunicorn app:app", and the frontend on Vercel with no build command. The Twitter token and the allowed frontend origin are supplied as environment variables.

Cold-start handling. Render free instances sleep after 15 minutes of inactivity. When a request takes longer than expected, the frontend shows the notice "Waking up the server, this can take up to 60 seconds on first use", so that users do not think the application has failed. The system is deliberately stateless: results are computed per request and no database is used.

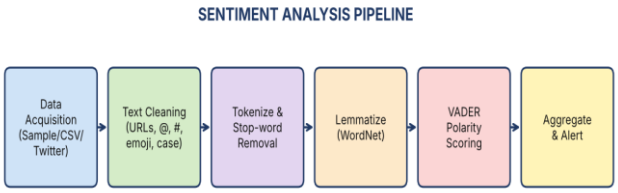


Fig. 2: Sentiment analysis pipeline

RESULTS AND DISCUSSION

The built-in sample dataset was analysed for the brand "Apple" with 50 posts and an alert threshold of 40%. The output is summarised in Table II.

Table II: Sample run (brand: Apple, 50 posts, threshold 40%)

Metric	Value
Total posts analysed	50
Positive posts	32 (64.0%)
Neutral posts	9 (18.0%)
Negative posts	9 (18.0%)
Average compound score	0.4851
Brand health	Healthy
Alert triggered	No (18% < 40%)
Top keywords	camera (18), battery (14)

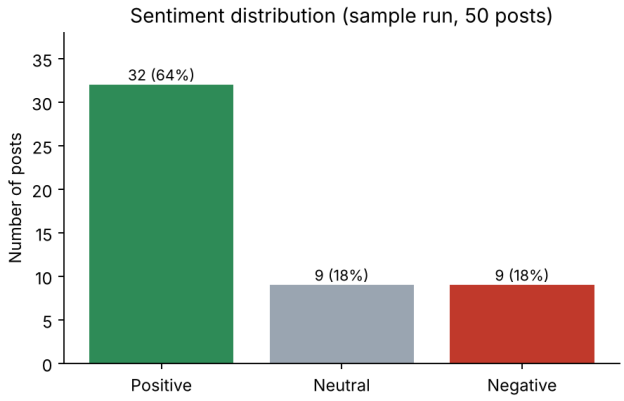


Fig. 3: Sentiment distribution for a 50-post sample run

Favourable posts clearly dominate, with positive posts about 3.5 times as frequent as negative posts. The average compound score of 0.4851 confirms an overall favourable tone, and the health indicator reports the brand as healthy.

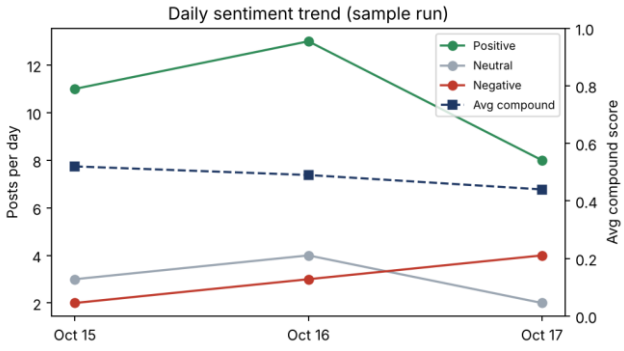


Fig. 4: Daily sentiment trend for a sample run

The trend for 15–17 October shows positive posts peaking on 16 October (13 posts) and falling to 8 on 17 October, while negative posts rose from 2 to 4 and the average compound score dropped from 0.52 to 0.44. Although the brand remains healthy, this gradual drift is exactly the early signal that a daily trend view is designed to reveal and that a single overall percentage would hide.

Keywords. "Camera" (18 mentions) and "battery" (14 mentions) were the most frequent terms, showing which product aspects dominate the conversation. The positive and negative word clouds help a team decide which features to promote and which to investigate.

Functional testing. The Pytest suite validates all four endpoints: /health returns status ok; the sample, CSV and live endpoints return the unified JSON response or handle the request correctly. All four tests passed. Manual testing covered each data-source tab, empty-input validation, the alert banner colour change when the threshold is lowered below the negative percentage, chart rendering, word-cloud display, table search and CSV export.

Comparison. BrandPulse trains no model, so its classification quality is that of VADER, which its authors validated on social media text [3]. Compared with Naive Bayes or SVM, VADER needs no labelled corpus; compared with LSTM or BERT-class models, it has very low compute cost and is explainable. It therefore suits 512 MB free-tier instances, where transformer models are a poor fit.

Advantages of the Proposed System

1. **Low Cost and Open Source:** built from open-source libraries and deployable on free hosting tiers.

2. **Explainable Scoring:** each label can be traced to the words and rules that produced it.

3. **Multiple Data Sources:** sample data, CSV upload and live Twitter search share one response format.

4. **Actionable Alerting:** a configurable negative-sentiment threshold and a brand-health label tell a team when to act.

5. **Maintainable Architecture:** the pure JSON API can be reused by other clients without changing the NLP code.

LIMITATIONS

VADER is English-centric and may misclassify sarcasm, irony and domain-specific slang. Live Twitter search depends on API access level, rate limits and token availability. Free-tier hosting causes cold starts and limited memory, which restricts the number of posts per request. The system is stateless, so historical comparisons are not stored, and accuracy against human-labelled ground truth for specific brands has not yet been measured.

CONCLUSION AND FUTURE SCOPE

BrandPulse shows that a transparent and deployable brand-sentiment monitoring tool can be built with open-source libraries and free hosting. Its split-deployment design, unified response schema, configurable alert and cold-start-aware interface turn raw posts into clear indicators, a daily trend, keywords and word clouds, and an automated test suite validates all four REST endpoints.

Future work includes transformer-based models such as BERTweet with VADER as a lightweight fallback, multilingual support including Hindi and Marathi, sarcasm and emoji-aware analysis, additional platforms such as Reddit and YouTube, a database for historical analyses, scheduled e-mail or Slack alerts, and aspect-based sentiment analysis for features such as camera, battery and price.

REFERENCES

[1] B. Pang and L. Lee, "Opinion mining and sentiment analysis," Foundations and Trends in Information Retrieval, vol. 2, no. 1–2, pp. 1–135, 2008.

[2] B. Liu, Sentiment Analysis and Opinion Mining. San Rafael, CA, USA: Morgan & Claypool, 2012.

[3] C. J. Hutto and E. Gilbert, "VADER: A parsimonious rule-based model for sentiment analysis of social media text," in Proc. 8th Int. AAAI Conf. Weblogs and Social Media (ICWSM), Ann Arbor, MI, USA, 2014.

[4] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," in Proc. EMNLP, 2002, pp. 79–86.

[5] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," CS224N Project Report, Stanford University, 2009.

[6] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in Proc. 7th Int. Conf. Language Resources and Evaluation (LREC), Valletta, Malta, 2010.

[7] E. Kouloumpis, T. Wilson, and J. Moore, "Twitter sentiment analysis: The good the bad and the OMG!," in Proc. 5th Int. AAAI Conf. Weblogs and Social Media (ICWSM), 2011.

[8] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment analysis of Twitter data," in Proc. Workshop on Languages in Social Media, ACL, 2011, pp. 30–38.

[9] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," Ain Shams Engineering Journal, vol. 5, no. 4, pp. 1093–1113, 2014.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc.

NAACL-HLT, Minneapolis, MN, USA, 2019, pp. 4171–4186.

[11] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "BERTweet: A pre-trained language model for English tweets," in Proc. EMNLP: System Demonstrations, 2020, pp. 9–14.

[12] F. Barbieri, J. Camacho-Collados, L. Espinosa Anke, and L. Neves, "TweetEval: Unified benchmark and comparative evaluation for tweet classification," in Findings of EMNLP, 2020.

[13] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. Sebastopol, CA, USA: O'Reilly Media, 2009.

[14] Pallets Projects, "Flask documentation." [Online]. Available: https://flask.palletsprojects.com

[15] Twitter (X) Developer Platform, "Twitter API v2 documentation." [Online]. Available: https://developer.x.com/en/docs/twitter-api

[16] Chart.js Contributors, "Chart.js documentation." [Online]. Available: https://www.chartjs.org/docs

[17] Render, "Render documentation: Web services." [Online]. Available: https://render.com/docs

[18] Vercel, "Vercel documentation: Static deployments." [Online]. Available: https://vercel.com/docs

BIOGRAPHIES

MS. PRACHI PRAKASH GARODI

Pursuing B.E. (Computer Science and Engineering)
INTERNATIONAL CENTER OF EXCELLENCE IN ENGINEERING AND MANAGEMENT (ICEEM), CH. SAMBHAJINAGAR

MS. SHRUTIKA SHAKULAL BANKAR

Pursuing B.E. (Computer Science and Engineering)
INTERNATIONAL CENTER OF EXCELLENCE IN ENGINEERING AND MANAGEMENT (ICEEM), CH. SAMBHAJINAGAR

MS. TRUPTI BABASAHEB DAVANGE

Pursuing B.E. (Computer Science and Engineering)
INTERNATIONAL CENTER OF EXCELLENCE IN ENGINEERING AND MANAGEMENT (ICEEM), CH. SAMBHAJINAGAR

PROF. A. K. HIWALE

Guide, Dept. of Computer Science and Engineering
INTERNATIONAL CENTER OF EXCELLENCE IN ENGINEERING AND MANAGEMENT (ICEEM), CH. SAMBHAJINAGAR