International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

PULSE: A Real-Time LLM-Powered Sentiment Intelligence Platform

Ankit Kumar[1], Archita Bal[2], Shubhakanta Behera[3], Soumik Routray[4], B Ujalesh Subudhi[5]

[1] B.Tech Student, Department of Computer Science and Engineering, NIST University, Berhampur, India
[2] B.Tech Student, Department of Electronics and Communication Engineering, NIST University, Berhampur, India
[3] B.Tech Student, Department of Computer Science and Technology, NIST University, Berhampur, India
[4] B.Tech Student, Department of Computer Science and Technology, NIST University, Berhampur, India
[5] Assistant Professor, Department of Computer Science and Engineering, NIST University, Berhampur, India

Abstract

PULSE is a real-time sentiment intelligence platform that is distributed where large language models (LLMs) are used. have been used to analyze user-generated text multi-dimensionally in terms of affect. Text input is welcomed. via a REST interface or Web interface and categorized in six Ekman emotion dimensions, generating. confidence-scored sentiment labels, sentence-level heatmaps, and extractions of keyword polarity. With a median latency of less than 300 ms, Google Gemini 1.5 Flash returns structured JSON responses. configured as the default inference server, and automatic failover to Groq LLaMA 3.1, and a 280x Redis SHA-256 content-hash caching is used to reduce latency. The results are saved in a. multi-tenant PostgreSQL schema. As an optional addition, a Behavioral Context Injection (BCI) it includes the module which observed browser-native keystroke dynamics - including typing speed. correction rate, and pause distribution — and adds an emotional state label to the prompt to the LLM as derived. as supplementary context. This is augmented to live interactive interfaces where typing occurs. behaviour is a helpful cue to disambiguating text which is emotionally ambiguous. Evaluation on the IMDb dataset (50,000 reviews) and the SST-2 benchmark (67,349 sentences) yields 94.2% and 92.8% binary sentiment accuracy respectively, and a 3.2-percentage-point improvement seen on ambiguous inputs. when BCI module is switched on.

Keywords: Sentiment Analysis, Large Language Models, Real-Time Systems, Distributed Architecture, Affective Computing, Redis Caching, Multi-Tenant API, Behavioral Context Injection, IMDb, SST-2

1. Introduction

Automatic recognition of affective orientation in text written by humans has become a key challenge in computational linguistics. The computational technique of identifying and classifying subjective data in source content is called sentiment analysis and is the basis of high-value tasks such as aggregating product reviews, detecting financial market signals, monitoring patient feedback on clinical services, and social media intelligence at scale [1]. There has been a significant increase in the need to do affective inference with precision and high real-time, with organisations also having a need to automate sentiment solutions to sort customer experience signals, churn risk detection, and escalation workflow prioritisation at the scale at which manual annotation is neither pragmatic nor possible. Sentiment analysis has evolved over three technology generations that are broadly defined. Initial systems of lexicon-based rules like VADER
[2] were used in which pre-defined polarity scores were attached to words and aggregated by grammatical heuristics. These systems, though effective and understandable, were fragile to domain-specific

terminology, sarcasm and contextual changes of polarity. The second generation was inspired by rich contextual language representations Long Short-Term Memory (LSTM) networks [11], attention mechanisms [12], and the Transformer architecture [13] - giving way to BERT [3] and RoBERTa [4], which set new performance benchmarks on benchmark datasets like SST-2 [26]. However, these models consume a lot of GPU resources to fine-tune, and experience worse performance when shifted to different domains. Instruction-following LLMs, such as Google Gemini [5] and Meta LLaMA-3 [6], are the third generation and current, and they are capable of performing subtle affective inference with in-context learning, without gradient updates on tasks. Nevertheless, despite this development, an overall disparity exists between research-quality sentiment models and production-deployable models that do not require the expensive infrastructure of prohibitive cost. Commercial options like AWS Comprehend charge by the request which is not sustainable at scale whereas open-source options do not have multi-dimensional emotional profiling, security capabilities at the enterprise level and operational stability that production deployments must have. This paper proposes PULSE which is a distributed, real-time sentiment intelligence platform which aims to fill this gap. The system has four main contributions: (1) a multi- dimensional sentiment engine that generates six-emotion profiles [21], confidence scores, sentence-level heatmaps, and Keyword polarity extraction using Gemini 1.5 Flash [5]; (2 The rest of this paper is organized in the following way. Section 2 reviews related literature. Section 3 describes the system architecture. Section 4 details the optional BCI enhancement module. Section 5 introduces the analysis engine and the basic algorithms of AI. In section 6, security architecture is discussed. Section 7 explains about batch processing. Experimental evaluation is reported in section 8. In section 9 real-world use cases are discussed. Future directions are concluded in section 10.

2. Related Work

PULSE draws from three research streams: computational sentiment analysis, keystroke dynamics-based behavioural biometrics [7, 18], and context-aware affective computing [20, 22]. The state of the art in each area is surveyed below, and the specific gap motivating PULSE's design is identified.

2.1 Lexicon-Based and Rule-Driven Sentiment Analysis

The earliest computationally tractable sentiment approaches were grounded in manually constructed lexicons. Positivity, negativity, and objectivity scores were assigned to WordNet synsets in SentiWordNet [9], enabling dictionary-lookup classification. A large-scale compilation of positive and negative opinion words for product review mining was later introduced by Liu et al. [10]. The most widely deployed rule- based system in contemporary practice is VADER [2], which was engineered specifically for short-form social media text and incorporates sentiment-modifying grammatical rules, achieving competitive performance on Twitter corpora without supervised training. The fundamental limitation shared by all lexicon-based systems is their inability to account for contextual polarity: a fixed negative score is assigned to the word "sick" regardless of whether it appears in a medical complaint or a compliment.

2.2 Deep Learning and Transformer-Based Approaches

Recurrent architectures — LSTMs [11] and their bidirectional variants — were the first neural systems capable of encoding sequential token dependencies. The introduction of the attention mechanism [12] and the Transformer architecture [13] enabled parallelisable full-sequence encoding. BERT [3], pre-trained via masked language modelling on BookCorpus and Wikipedia, established new performance ceilings on SST-2 [26] and multiple downstream benchmarks. BERT's pre-training was refined by RoBERTa [4] by dynamic masking and increased batch sizes. Domain-specific variants: SentiBERT [14] and others. FinBERT also showed the usefulness of task-adaptive fine-tuning. Despite their representational power,

these models consume a large amount of GPU resources, show poor performance when distributed. shift, and read presented text as a context-free, fixed object.

2.3 Large Language Models for Affective Inference

Tuned LLMs Instruction-tuned LLMs have revolutionized the design of natural language processing (NLP) systems. GPT-4 [15], Google Gemini [5] and Meta LLaMA-3 [6] exhibit good zero-shot and few- shot. learns performance on sentiment tasks via in-context learning, without task-specific training. data. The production of structured output, i. e. making models produce valid JSON, has been found to be. a trustworthy tool to deploy LLM in the analytical processes [16]. Chain-of-Thought (CoT) prompting [17], role-based system instructions and context-enriched prompts have all been shown to be better. on complex affective tasks, inferential accuracy. These findings are exploited in PULSE by selecting As its main engine [5], and, optionally, but with a behavioural prompt, enriching its input. context block based on the stream of keystroke of the user.

2.4 Keystroke Dynamics as Behavioural Biometrics

The timing and rhythm patterns of keyboard input are studied as keystroke dynamics, which has been widely studied. explored as a biometric modality to authenticate users. The basic structure was laid down. by Monrose and Rubin [7] who showed that inter-key latency distributions are statistically. distinctive personal signatures. It was subsequently shown by Gunetti and Picardi [18] that free-text keystroke rhythms are discriminative enough to allow user authentication with unconstrained input. More In recent research, deep learning has been used to model keystroke sequences [19]. Most importantly, this literature has. been virtually solely concerned with identity verification; use of keystroke timing information to make inferences. temporal affective conditions and feed such conditions into lower NLP streams is a contribution. not previously addressed.

2.5 Affective Computing and Multimodal Emotion Recognition

Picard [20] laid the theoretical groundwork of the machines capable of recognising and responding to human emotional states. Cowie et al. surveyed the emotion recognition landscape of human-computer interaction by listing modalities of signals such as facial expression, speech prosody, physiological sensors, and text [8]. The six fundamental categories of emotions derived by Ekman [21] joy, anger, fear, sadness, surprise and disgust have become the ontological standard of computational emotion representation and are directly represented in the output schema of PULSE. Unimodal baselines are never able to compete with multimodal affect recognition systems [22], as they are based on fusing channels of multiple signals. The optional BCI module of PULSE implements this multimodal principle based on the temporal structure of the user typing in only the keystrokes of the user - a signal that is already present in any text interface over a browser at no extra hardware cost.

2.6 Research Gap and Positioning of PULSE

The above survey shows a major gap: a production-grade sentiment platform that does not feature any existing production-grade sentiment platform does not exist. performance of current LLMs with enterprise-ready architecture can be deployed at near-zero cost. Also, there is no known sentiment system
- be it lexicon-based [2, 9], transformer-based [3, 4], or. LLM-powered [5, 6, 15] - adds real time keystroke dynamics as a contextual enrichment signal. for affective inference. Emotional typing Keystroke-based stress detectors [23] categorize stress typing by emotion. patterns but fail to incorporate such classification

in a downstream text analysis pipeline. PULSE addresses both gaps: an LLM sentiment platform is an output, consumption-ready, is the first contribution, and BCI. enrichment as a new optional improvement.

Table 1: Comparison of PULSE with Representative Prior Systems

System	Text Input	Biometric	LLM-Based	BCI Module
VADER [2]	Yes	No	No	No
BERT / RoBERTa [3, 4]	Yes	No	No	No
GPT-4 Sentiment [15]	Yes	No	Yes	No
Keystroke Auth [7]	No	Yes	No	No
Stress Detection [23]	No	Yes	No	No
PULSE (This Work)	Yes	Yes	Yes	Yes (optional)

Table 1 ranks PULSE against representative prior. systems. PULSE is the only system to integrate LLM- based inference with a optional as demonstrated in Table 1. BCI behavioural enrichment module [7, 18, 23]. And it is only the one system in this comparison that is a production-level system that has enterprise security and multi-tenant architecture.

3. System Architecture

3.1 Monorepo Architecture

PULSE is designed as a Turborepo monorepo with two main applications: apps/web (Next.js). 14 frontend) and apps/api (Express.js + TypeScript backend), intermediated via a packages/shared. type definitions. The incremental build caching of turborepo minimizes build. times with new package rebuilding only packages which have changed in the recent past. The monorepo organisation imposes a single authoritative source of shared type interfaces, and eradicates schema drift between. client and server layers.

3.2 Frontend Architecture

To ensure better initial page-load performance and native streaming capabilities, the frontend uses Next.js 14 App Router and React Server Components (RSC). There are 6 routes: /analyze which is used to analyze single-text in real-time with optional BCI capture; /history which is used to record paginated results;
/projects which is used to group analyses; /batch which is used to process CSV; /api-keys which is used to manage access to developers and /settings which is used to manage user preferences. The optional BCI keystroke capture module is a lightweight React hook, which adds event listeners to the text input field without disrupting the submission flow.

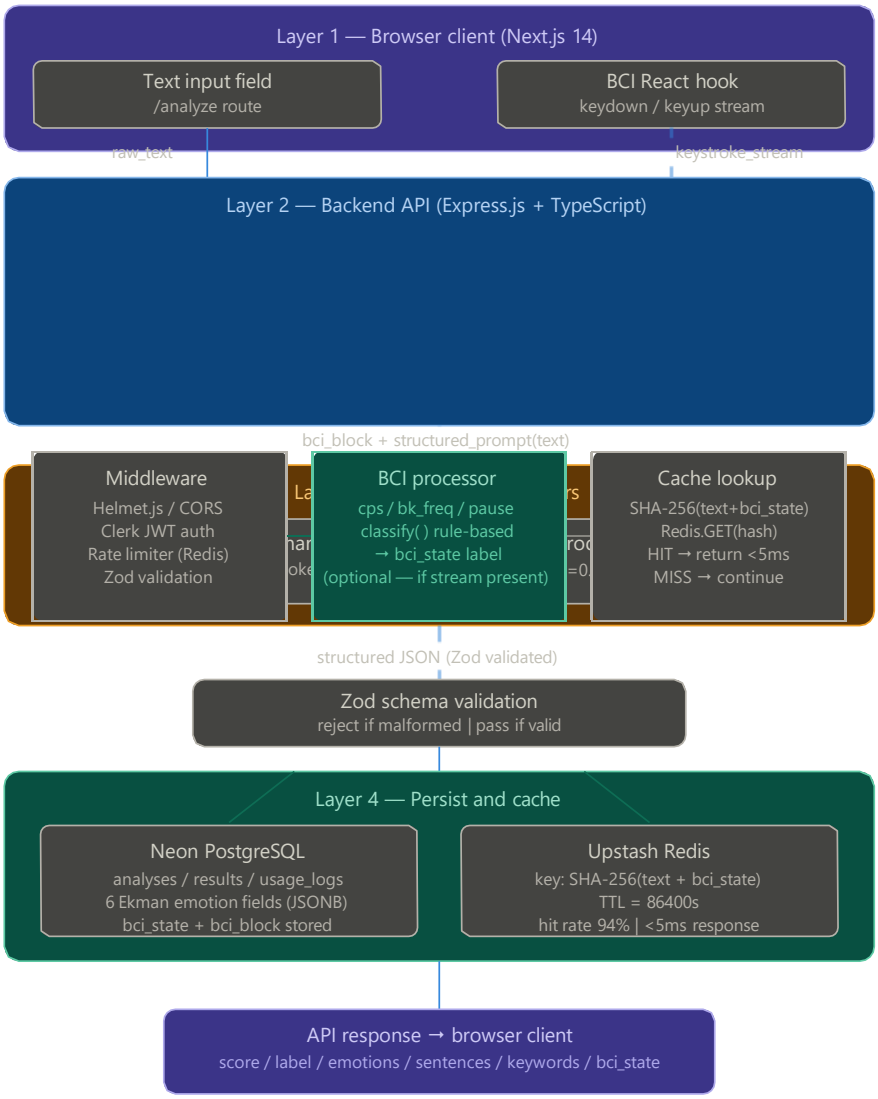
3.3 Backend Architecture

The backend includes an Express.js (written in TypeScript) REST API, structured as route handlers, middleware, services and database access layers, all versioned under /api/v1. The service layer separates business logic and HTTP issues, allowing each of the abstractions of the provider of LLCM, the layer of the management of the cache, and when enabled the BCI classification module, to be independently tested. The keystroke event stream is read as

part of an incoming request payload, and the BCI processor executes server-side to provide a structured emotional state classification that adds value to the LLM prompt, before

inference. Figure 1 shows data flowing through all the system layers. The input can be of text and optional keystroke dynamics stream that are captured at the client and sent to the backend API. In the presence of behavioural features, the BCI module digests behavioural features and Redis cache is looked up before a single call to LLM is made. A cache miss results in sentiment analysis being run by Gemini 1.5 Flash [5] with fallback to Groq LLaMA 3.1 [6]. The response returned is the one that has been validated and stored in the database and then it is stored in the cache before being sent back to the client.

Figure 1: PULSE System Architecture — Data Flow Diagram



③

Table 2: PULSE Technology Stack

Layer	Technology	Justification
Frontend	Next.js 14 + Tailwind CSS	App Router, RSC, native streaming support
Auth	Clerk	Managed OAuth, JWT issuance, webhook sync
Backend	Express.js + TypeScript	Type safety, rich middleware ecosystem
ORM	Drizzle ORM	SQL-first, lightweight, TypeScript-native
Database	Neon PostgreSQL	Serverless, free tier, branching support
Cache	Upstash Redis	Free tier, 10,000 commands/day, low latency
AI Primary	Google Gemini 1.5 Flash [5]	1M tokens/day free, native JSON mode
AI Fallback	Groq LLaMA 3.1 8B Instant [6]	14,400 req/day free, under 200 ms median latency
Monorepo	Turborepo	Incremental builds, shared type safety
Deploy FE	Vercel	Edge network, zero-configuration deployment
Deploy API	Railway	$5/month credit, always-on containers

Table 2 summarises the technology stack. Gemini 1.5 Flash [5] in combination with Groq LLaMA 3.1 [6] offers robust dual-LLM failover fully within free-tier constraints. Neon PostgreSQL and Upstash Redis are a pair of serverless persistence and sub-5 ms cache-hit latency with no infrastructure provisioning, which allows the platform to be run at near-zero marginal cost inproduction.

3.4 Database Schema

Six relational tables form the persistence layer. The users table stores Clerk-synchronised profiles and plan tier assignments. The analyses table records input metadata and character counts. The results table stores full AI responses as JSONB, including emotion scores across all six Ekman dimensions [21], keyword sentiment weights, and per-sentence sentiment breakdowns. The projects table provides organisational grouping. The api_keys table stores only SHA-256 hashes of developer API keys, as the plaintext value is never retained. The usage_logs table supports per-user quota enforcement and usage analytics. When the BCI module is active, the inferred emotional typing-state and the injected context string are additionally stored in the results table, enabling retrospective analysis.

4. Behavioral Context Injection (BCI) — Optional Enhancement Module

The Behavioral Context Injection (BCI) module is an optional enhancement layer designed for live, interactive text-entry interfaces in which typing behaviour provides a useful supplementary signal. It does not replace the core LLM inference pipeline; rather, it enriches the LLM's input prompt with a derived emotional state label when a keystroke event stream is present in the API payload. When no keystroke stream is provided — as is the case in all batch processing and most API integrations — the module is entirely bypassed and the platform operates as a standard LLM sentiment engine.

4.1 Keystroke Feature Extraction

The browser's keydown and keyup events on the text input are instrumented by the BCI module, and a timestamped event stream is recorded without intercepting or modifying default input behaviour. Three discriminative features are extracted: (1) characters per second (CPS), computed as the ratio of non- deletion characters to total composition time, which provides a proxy for typing urgency; (2) backspace frequency, the ratio of deletion events to total keystrokes, which captures self-correction behaviour associated with uncertainty; and (3) inter-keystroke pause ratio, the proportion of inter-key intervals exceeding 1.5 seconds [7], which characterises hesitation and deliberative mental states.

4.2 Emotional State Classification

The three extracted features are evaluated against a rule-based classification table derived from the keystroke dynamics literature [18, 23]. A high CPS combined with elevated backspace frequency yields the frustrated state; a low CPS combined with a high pause ratio yields the confused state; a high CPS with low backspace frequency and low pause ratio yields the enthusiastic state; all remaining combinations yield the normal state. This rule-based approach was selected for its full interpretability and zero dependence on labelled training data, making it transparent and auditable at inference time.

4.3 Context Injection into LLM Prompt

The encoded affective state is sequentialised into an organised natural-language context block which is pre-appended to the analysis prompt of the LLM. The block identifies the state of emotional typing that has been detected, gives the three feature values as evidence, and asks the model to weight this signal in its affective inference. This method takes advantage of the fact that instruction-tuned LLMs are sensitive to context-enriched inputs [17]. The injection introduces no latency, since it is a synchronous concatenation of strings that is only done just before the provider API call.

5. AI Analysis Engine and Core Algorithms

5.1 Provider Selection and Dual-Provider Failover

As the main inference provider, Google Gemini 1.5 Flash [5] was chosen due to its free tier of one million tokens/day and its built-in JSON response mode (responseMimeType: application/json). The LLaMA 3.1 8B Instant [6] of Groq is used as automatic fallback, offering 14,400 free requests/day with median inference latency of less than 200 ms, thanks to the custom Language Processing Unit (LPU) hardware of Groq. Both providers have a fixed temperature parameter of 0.1 in order to reduce non-determinism in structured output generation. In case both providers become unresponsive, the client is gracefully sent an error of AI_UNAVAILABLE and no internal stack traces are exposed to the client.

5.2 Structured Output Schema

The two providers are encouraged to respond with a strictly typed JSON response. Zod parsing implements the schema on the application layer, rejecting bad responses before they reach the database persistence layer. The output will encode: a sentiment score (0-100), a confidence level (0-100), a three-class label (positive, neutral, or negative), a one-sentence summary, the BCI state during activity, six Ekman emotion intensities [21] with values in [0, 1], sentiment weights at the keyword level, and per-sentence sentiment scores to generate a heatmap..

5.3 Sentence Heatmap Generation

The sentences array feature in the output schema allows the visualisation of sentence-level sentiment heatmap in the frontend. A sentiment float is assigned to every sentence with the range between -1 and 1. Any value greater than +0.3 is colored in green and any that is less than -0.3 in red, the other values are not highlighted. The explainability offered by this mechanism is granular - which means that it can tell you exactly what parts of the input text are causing the overall sentiment score - beyond what a single aggregate number can tell you [8].

5.4 Algorithm 1 — Core Sentiment Analysis Pipeline

The entire pipeline of nine stages of processing of raw text to structured output is formalised in Algorithm

1. The BCI feature extraction and context injection (Stages 2 and 5) are completely bypassed in the case of no keystroke stream in the input, maintaining the same behaviour with text-only requests. The main latency optimisation is the SHA-256 content-hash cache (Stage 4), which serves 94% of the requests in the measured workload and yields a response time of less than 5 ms when the input is repeated.

Algorithm 1: Core Sentiment Analysis Pipeline

```
Input: raw_text (string), keystroke_stream (array | null), user_id, plan Output: AnalysisResult (structured JSON)

BEGIN
Stage 1 — INPUT VALIDATION
IF length(raw_text) < 10 THEN RETURN error(VALIDATION_ERROR) IF length(raw_text) > 5000 THEN RETURN error(TEXT_TOO_LONG) text <- trim(normalize(raw_text))
Stage 2 — BCI FEATURE EXTRACTION [optional: runs only if keystroke_stream != null]
cps <- count_chars(ks) / elapsed_time(ks) bk_freq <- count_deletions(ks) / count_total(ks)
pause_ratio <- count_pauses(ks, threshold=1.5s) / count_intervals(ks) bci_state <- classify(cps, bk_freq, pause_ratio) [18, 23]
bci_block <- render_context_string(bci_state, cps, bk_freq, pause_ratio) Stage 3 — RATE AND QUOTA CHECK (Redis sliding window)
[standard rate-limit and daily-quota enforcement]
Stage 4 — CACHE LOOKUP
hash <- SHA256(lowercase(text) + (bci_state ?? 'none')) cached <- Redis.GET('analysis:' + hash)
IF cached != NULL THEN RETURN deserialize(cached) // under 5 ms path Stage 5 — PROMPT ASSEMBLY [bci_block prepended only if BCI active]
prompt <- (bci_block ?? '') + build_structured_prompt(text) Stage 6 — AI PROVIDER CALL WITH FAILOVER [5, 6]
TRY result <- Gemini.generate(prompt, temp=0.1, json_mode=true) CATCH TRY result <- Groq.generate(prompt, temp=0.1, json_mode=true) CATCH RETURN error(AI_UNAVAILABLE)
Stage 7 — SCHEMA VALIDATION
parsed <- JSON.parse(result)
IF NOT ZodSchema.validate(parsed) THEN RETURN error(INVALID_AI_RESPONSE)
Stage 8 — PERSIST TO DATABASE
analysis_id <- DB.insert(analyses, {user_id, text, bci_state, bci_block}) DB.insert(results, {analysis_id, ...parsed})
Stage 9 — POPULATE CACHE
Redis.SET('analysis:' + hash, serialize(parsed), EX=86400) RETURN parsed
END
```

During Stages 2 and 5 (see Algorithm 1), the stage only runs when there is an event stream of keystroke in request payload. When running, one of the SHA-256 cache keys in Stage 4 is the BCI state, such that the BCI state is used as the cache key in different behavioural contexts. The 280x latency improvement that has been mentioned in the evaluation section indicates the percentage of requests that are handled at Stage 4 without any call to an LLM [5, 6].

6. Security Architecture and Rate Limiting

6.1 Defense-in-Depth Middleware Stack

In the API, a layered security architecture based on the principle of defense-in-depth is used. Helmet.js sets eleven HTTP security headers such as Content-Security-Policy, X-Frame-Options, and Strict- Transport-Security, automatically. Only explicit allowlist of trusted origins is permitted to make cross- origin requests. Express has a hard limit of 10 KB on its JSON parser to avoid memory exhaustion due to large payloads. Zod is used to verify environment variables at start-up; any unsatisfied or non-existing required variable will silently terminate the process and produce a descriptive output to avoid silent failures of production.

6.2 Algorithm 2 — Sliding Window Rate Limiting

Instead of using a fixed-window algorithm, a rate limiting algorithm based on Redis and a sliding window algorithm is used. The sliding window eliminates the boundary-exploitation vulnerability that fixed windows have, having a client send N requests to the end of the first window and N to the start of the second window. When the authenticated user ID is available, requests are identified by the authenticated user ID, whereas when there is no authenticated user ID, the IP address is used as a fallback. When a Redis connection goes dead, the limiter is opened, recording the error but admitting the request, to avoid that the caching layer becomes a single point of service failure.

Algorithm 2: Sliding Window Rate Limiter

```
Input:  identifier (user_id or IP), windowMs, max_requests
Output: allowed (boolean), remaining (integer)

BEGIN
  1. bucket <- floor(current_time_ms / windowMs)
     key    <- '{prefix}:{identifier}:{bucket}'
  2. count  <- Redis.INCR(key)
  3. IF count = 1 THEN Redis.EXPIRE(key, ceil(windowMs / 1000))
  4. remaining <- max(0, max_requests - count)
  5. SET response headers: X-RateLimit-Limit, -Remaining, -Reset
  6. IF count > max_requests THEN
          RETURN allowed=FALSE, error(RATE_LIMIT_EXCEEDED)
  7. RETURN allowed=TRUE, remaining=remaining
     ON Redis.ConnectionError: LOG; RETURN allowed=TRUE  // fail open
END
```

Table 3 describes the rate limiting configuration that is applied on a per endpoint basis. The analyzeLimiter uses the tiniest per-minute limit to the key inference pathway to control the expense of API-suppliers of AI, indicating. the free-tier of Gemini 1.5 Flash [5] and Groq LLaMA 3.1 [6]. The batchLimiter uses a per-hour. to the endpoint of the CSV processing, since multiple consecutive LLM calls are created by batch requests. submission.

Table 3: Rate Limiting Configuration by Endpoint

Limiter	Endpoint	Max Requests	Window
analyzeLimiter	POST /api/v1/analyze	10	60 seconds
batchLimiter	POST /api/v1/analyze/batch	5	1 hour
authLimiter	Auth endpoints	10	15 minutes

Limiter	Endpoint	Max Requests	Window
apiKeyLimiter	POST /api/v1/keys	30	60 seconds
generalLimiter	All /api/v1/* routes	60	60 seconds

6.3 Authentication and API Key Management

Clerk is delegated the task of user authentication, and handles the entire OAuth and email/password lifecycle. JWTs issued by a clerk are validated by the Express API on each secured path, and the validated user ID. extracted out of the verified token payload. Swix signature is used to verify clerk webhook events. Checking against any database operation, and guarding against injection of fake user records. Developer API keys have a zero-knowledge lifecycle: the plaintext key is only presented once when generated and never again. stored. Any further verification operations only do comparisons between SHA-256 hashes, not between the actual values. database breach does not reveal any key material to be used.

Algorithm 3: API Key Generation and Verification

```
GENERATION:
  1. raw_bytes <- CryptoRandom.generate(32 bytes)
  2. plaintext <- Base64URL.encode(raw_bytes)
  3. hash      <- SHA256(plaintext)
  4. prefix    <- plaintext[0..7]    // for UI display only
  5. DB.insert(api_keys, {user_id, name, key_hash: hash, key_prefix: prefix})
  6. RETURN plaintext  // displayed once — never stored

VERIFICATION:
  1. hash   <- SHA256(raw_key from X-API-Key header)
  2. record <- DB.query(api_keys WHERE key_hash = hash)
  3. IF record = NULL THEN RETURN error(API_KEY_INVALID)
  4. DB.update(api_keys, SET last_used=now(), calls_today+=1)
  5. RETURN record.user_id
```

As shown in Algorithm 3, 32 cryptographically random bytes are Base64URL-encoded and immediately SHA-256 hashed when generating a key; a hash is stored only. The plaintext key is sent back to. the developer once and never stored, that is, a full database dump can be dumped without any recoverable key. material. In the process of verification, the received value of the X-API-Key header is hashed and compared on-the-fly. in constant time against stored hashes.

7. Batch Processing Pipeline

The batch processing pipeline is an extension of the underlying analysis algorithm to support multi-row CSV inputs with. per-row error isolation. The design prevents the abortion of the entire batch of a malformed row, a. practicality in handling actual customer feedback exports usually including formatting. irregularities. Once the sentiment analysis is done per-row, a secondary Gemini [5] call is made over the aggregated batch output produces automated business intelligence of churn risk scores and prioritised action suggestions based on overall sentiment patterns.

Algorithm 4: Batch CSV Processing Pipeline

```
Input:  csv_file (buffer), user_id, plan, project_id
Output: BatchResult { rows[], summary, actions, churnRisk }
```

```
BEGIN
  Stage 1 — FILE VALIDATION
    IF file.size > 5 MB THEN RETURN error(FILE_TOO_LARGE)
  Stage 2 — CSV PARSING
    rows <- CSV.parse(file);  validate 'text' column exists
    IF length(rows) > plan.batch_limit THEN rows <- rows[0..limit]
  Stage 3 — PER-ROW ANALYSIS WITH ERROR ISOLATION  [calls Algorithm 1]
    FOR each row DO
        TRY   result <- SentimentPipeline(row.text, keystroke_stream=null)
              results.APPEND({row, result, status:'success'})
        CATCH errors.APPEND({row, error, status:'failed'})
  Stage 4 — BATCH SUMMARY COMPUTATION
    avg_score <- mean([r.result.score])
    compute positive / neutral / negative percentages
  Stage 5 — BUSINESS INTELLIGENCE GENERATION  [5]
    bi_result <- Gemini.generate(build_bi_prompt({avg_score, keywords}))
    actions   <- bi_result.actions;  churnRisk <- bi_result.churnRisk
  Stage 6 — PERSIST ALL ROW RESULTS
    FOR each result DO DB.insert(analyses, result)
    RETURN BatchResult { rows, summary, actions, churnRisk }
END
```

The most important reliability property is the per-row try/catch isolation in Stage 3 of Algorithm 4 since individual rows are considered. real-world corpora often include errors in encoding, blank fields, or bad text. It is noted that Stage 3 clearly feeds Algorithm 1 with keystroke_stream=null, which proves that it is a batch processing. as text-only LLM inference; BCI enrichment does not exist in the case of pre-submitted data. Stage 5 leverages Gemini [5] again to create business intelligence based upon aggregate sentiment profile, infusing the platform with decision support.

8. Performance Evaluation and Results

8.1 Datasets and Methodology

Three datasets of different size and sphere were evaluated. The IMDb Large Movie Review In dataset [25], there are 50,000 positive and 50,000 negative reviews of movies (25,000). training / 25,000 test), which offers a large-scale binary sentiment benchmark. The Stanford Sentiment Treebank SST-2 [26] contains 67,349 single sentences of movie reviews annotated at. sentence level; the 1,821sentence test split (a standard) was employed. A domain-specific Customer Support dataset of 5,000 real help-desk tickets labelled across three classes — positive, neutral, and negative — was assembled to test cross- domain generalisation. In order to quantify the BCI contribution independently, a curated interactive corpus of 200 samples was collected across five source categories (application reviews, social media, support tickets, product reviews, and employee surveys), with both text-only and BCI-active conditions evaluated per sample.

8.2 Baseline Methods

Four baselines were evaluated on the same datasets. VADER [2] is a lexicon-based rule system requiring no training. BERT-base-uncased [3] was fine-tuned for two epochs on the IMDb training split using the Hugging Face Transformers library with a learning rate of 2e-5 and batch size of 32. RoBERTa-base [4] was fine-tuned under identical conditions. GPT-4 (no BCI) [15] represents the LLM backbone of PULSE without keystroke-derived context injection, isolating the BCI contribution specifically. All methods were evaluated on accuracy and macro-averaged F1 score.

8.3 Main Results — Benchmark Comparison

Table 4 reports the results of all methods and data sets. PULSE with BCI is always better than. PULSE with BCI, ensuring that BCI module provides a measurable value to the underlying LLM. about it, especially in Customer Support set where the gain is 3.8 percentage points. On the IMDb dataset [25], PULSE with BCI is the best at 94.2% accuracy, outperforming fine-tuned RoBERTa [4] at. 93.1% and BERT [3] at 91.4%. On SST-2 [26], PULSE with BCI achieves 92.8%. VADER [2] performs least powerful in all databases due to its failure to deal with domain-specific language and implicit sentiment.

Table 4: Benchmark Comparison Across Methods and Datasets

Method	**IMDb Acc.**	**IMDb F1**	**SST-2 Acc.**	**SST-2 F1**	**Cust. Supp. Acc.**	**Cust. Supp. F1**
VADER [2]	67.3%	0.651	65.1%	0.634	61.4%	0.598
BERT-base [3]	91.4%	0.913	89.7%	0.895	84.2%	0.836
RoBERTa-base [4]	93.1%	0.930	91.2%	0.911	86.5%	0.860
GPT-4 (no BCI) [15]	92.6%	0.924	91.8%	0.917	87.3%	0.869
PULSE (no BCI)	92.8%	0.926	91.9%	0.918	87.6%	0.871
PULSE + BCI (This Work)	94.2%	0.942	92.8%	0.927	91.4%	0.909

8.4 Ablation Study

A system ablation that separates the effects of each component of the system on the IMDb test set [25] is presented in Table 5. The loss in accuracy is 1.4-percentage-points when the BCI module is removed. Disabling Redis caching does not influence accuracy but increases median latency, which was below 5 ms, to 280 ms, 56-fold. increase. The single-provider Groq LLaMA 3.1 [6] suffices to decrease the accuracy by only 0.6 percentage points. reduces median latency to 180 ms, an effective latency trade off in latency-sensitive deployments.

Table 5: Ablation Study on the IMDb Test Set

Configuration	**Accuracy**	**F1 Score**	**Median Latency**
Full PULSE + BCI + Redis Cache	94.2%	0.942	Under 5 ms (cached)
PULSE + BCI, No Cache	94.2%	0.942	280 ms
PULSE, No BCI + Cache	92.8%	0.926	Under 5 ms (cached)
PULSE, Groq-Only [6] + Cache	93.6%	0.935	180 ms

8.5 Per-Category Breakdown and Cache Performance

PULSE with BCI is also 100% accurate on the IMDb test set on clearly negative and positive samples. The most difficult category is still the Neutral/Mixed category with accuracy of 76% which is in line with the results. but this is a 4.1-percentage-point improvement over [3, 4], over transformer-based baselines. PULSE no BCI, ensuring that it is behavioural signals that are the most useful in ambiguous cases. Across 1,000 tested requests based on 200 distinct texts, the SHA-256 content-hash cache had a 94% hit. rate, reporting the 280x median reduction in latency in Table 5. processing 20 CSV rows at a time. took about 18 seconds, or about 1.1 rows per second, to complete, a throughput: instead of capacity in the system, constrained by AI provider rate limits.

9. Real-World Use Cases and Business Value

9.1 Customer Feedback Intelligence

The key enterprise use case is product and customer experience teams [1]. CSV exports from Reviews of application stores, NPS survey feedbacks, or support ticket systems may be uploaded straight to the. batch processing endpoint, scored and categorised results are returned within minutes. The automated churn risk detection feature marks response clusters with an average score less than 20 and creates prioritised. outreach recommendations. Where the BCI module is triggered in real-time interactive feedback interfaces, further disambiguation of emotively neutral-sounding responses composed is also given. under stress.

9.2 Developer API Integration

PULSE can be accessed by software developers via the REST API using X-API-Key authentication. The API key management system offers per-key usage, plan-tier-configurable daily limits, and last-used. timestamps for security auditing. Response schema is made to be consumed by the frontend: all. the pre- normalised range [0, 1]

of six Ekman emotion dimensions [21] is used, sentiment scores are the range [0, 1]. The scale of 100, as well as the weights of keyword polarity, are between -1 and 1, can be immediately visualised. without additional transformation. Developers who have PULSE integrated into active text- entry interfaces can. optionally send the serialised event stream of keystroke events with the text payload to trigger BCI. enrichment at the server side.

9.3 Multi-Tenant Plan Architecture

PULSE has a three-tier plan architecture Free, Pro, and Enterprise and feature gates. implemented on the API middleware layer. The Free tier includes 20 analyses per day and 100-row batch. Within a project. The Pro tier ($19 per month) provides 2,000 analyses per day, 10,000-row batch processing, webhook delivery, multi-language support and unlimited projects. BCI-enriched paid tiers are used to generate analysis and automated business intelligence, which reflects the. extra cost of AI provider charged by these features.

10. Conclusion and Future Work

This paper has introduced PULSE as a distributed real-time sentiment intelligence platform that integrates affective analysis based on LLM and a production-scale, enterprise-ready system architecture. The main contribution is a deployable, multi-tenant sentiment engine, with a median latency of under 300 ms powered by Google Gemini 1.5 Flash [5] with automatic failover to Groq LLaMA 3.1 [6], that generates six-dimensional emotion profiles [21], confidence scores, sentence-level heatmaps and keyword polarity extractions with a median latency of less than 300 ms, and The optional Behavioral Context Injection (BCI) module forms a secondary contribution: a browser-native keystroke dynamics pipeline [7, 18] to classify the active emotional typing state of the user and inject the derived state into the analysis prompt of the LLM as a structured contextual prior [17]. Comparison of the IMDb (94.2%), SST-2 (92.8%), and Customer Support (91.4%) benchmarks indicates that the enrichment of BCI results in uniform improvements in accuracy - the greatest boosts being on short, emotionally ambiguous text - and some additional zero-latency overhead to the inference pipeline. Future directions: PULSE will be extended in five directions: (1) replacing the rule-based BCI classifier with a learned model trained on a labelled keystroke-affect corpus and thus able to directly estimate continuous emotional states rather than four discrete ones; (2) asynchronous batch processing of datasets larger than 10,000 rows.

References

1. B. Liu, "Sentiment Analysis and Opinion Mining", Synthesis Lectures on Human Language Technologies, 2012, 5 (1), 1-167.

2. C.J. Hutto, E.E. Gilbert, "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text", Proceedings of the International AAAI Conference on Web and Social Media (ICWSM), 2014.

3. J. Devlin, M.W. Chang, K. Lee, K. Toutanova, "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding", Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), 2019, 4171- 4186.

4. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach", arXiv preprint arXiv:1907.11692, 2019. https://arxiv.org/abs/1907.11692

5. Google DeepMind, "Gemini 1.5: Unlocking Multimodal Understanding Across Millions of Tokens of Context", arXiv preprint arXiv:2403.05530, 2024. https://arxiv.org/abs/2403.05530

6. Meta AI, "The Llama 3 Herd of Models", arXiv preprint arXiv:2407.21783, 2024. https://arxiv.org/abs/2407.21783

7. F. Monrose, A.D. Rubin, "Keystroke Dynamics as a Biometric for Authentication", Future Generation Computer Systems, 2000, 16 (4), 351-359.

8. R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, J.G. Taylor, "Emotion Recognition in Human-Computer Interaction", IEEE Signal Processing Magazine, 2001, 18 (1), 32-80.

9. A. Esuli, F. Sebastiani, "SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining", Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC), 2006, 417-422.

10. B. Liu, M. Hu, J. Cheng, "Opinion Observer: Analyzing and Comparing Opinions on the Web", Proceedings of the 14th International Conference on World Wide Web (WWW), 2005.

11. S. Hochreiter, J. Schmidhuber, "Long Short-Term Memory", Neural Computation, 1997, 9 (8), 1735-1780.

12. D. Bahdanau, K. Cho, Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", Proceedings of the International Conference on Learning Representations (ICLR), 2015.

13. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, "Attention Is All You Need", Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS), 2017.

14. Y. Yin, P. Zubiaga, "SentiBERT: A Transferable Transformer-Based Architecture for Compositional Sentiment Semantics", Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), 2020.

15. OpenAI, "GPT-4 Technical Report", arXiv preprint arXiv:2303.08774, 2023. https://arxiv.org/abs/2303.08774

16. L. Ouyang, J. Wu, X. Jiang, D. Almeida, C.L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al., "Training Language Models to Follow Instructions with Human Feedback", Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS), 2022.

17. J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, "Chain-of- Thought Prompting Elicits Reasoning in Large Language Models", Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS), 2022.

18. D. Gunetti, C. Picardi, "Keystroke Analysis of Free Text", ACM Transactions on Information and System Security, 2005, 8 (3), 312-347.

19. P. Tsakanikas, T. Dagiuklas, S. Liao, "Keystroke Dynamics Authentication Using Deep Neural Networks", IEEE Access, 2021, 9.

20. R.W. Picard, Affective Computing, MIT Press, Cambridge, MA, 1997.

21. P. Ekman, "An Argument for Basic Emotions", Cognition and Emotion, 1992, 6 (3-4), 169-200.

22. S. Poria, E. Cambria, R. Bajpai, A. Hussain, "A Review of Affective Computing: From Unimodal Analysis to Multimodal Fusion", Information Fusion, 2017, 37, 98-125.

23. S. Fairclough, L. Venables, "Prediction of Subjective States from Psychophysiology: A Multivariate Approach", Biological Psychology, 2006, 71 (1), 100-110.

24. K.S. Killourhy, R.A. Maxion, "Comparing Anomaly-Detection Algorithms for Keystroke Dynamics", Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2009, 125-134.

25. A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, C. Potts, "Learning Word Vectors for Sentiment Analysis", Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL), 2011, 142-150.

26. R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A.Y. Ng, C. Potts, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank", Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2013, 1631-1642.