Advanced Data Science and Computational Intelligence
Peer Reviewed • Open Access Journal

Research Article | Vol. 1 | Issue 1 | January-December 2026

# Real-Time Detection of Fake Reviews and Ratings Using Behavioral Patterns and Browser Integration

**Mahesh Bhandari[1*], Aadil Sarjekhan[2], Shubham Pathare[3], Bashir Shaikh[4], Shreekar Nyayapathi[5], Pratik Shrikhande[6]**

[1,2,3,4,5,6] *Vishwakarma Institute of Technology Pune, India*

[*]**Correspondence to:** Mahesh Bhandari, Vishwakarma Institute of Technology Pune, India. Email: mahesh.bhandari@vit.edu

**Received:** July 07, 2026; **Manuscript No:** JADS-26-1697; **Editor Assigned:** July 13, 2026; **PreQc No:** JADS-26-1697 (PQ); **Reviewed:** July 18, 2026; **Revised:** July 23, 2026; **Manuscript No:** JADS-26-1697 (R); **Published:** August 31, 2026

**Citation:** Bhandari M, Sarjekhan A, Pathare S, Shaikh B, Nyayapathi S, Shrikhande P (2026). Real-Time Detection of Fake Reviews and Ratings Using Behavioral Patterns and Browser Integratio. Adv. Data Sci. Comput. Intell. Vol.1 Iss.1, August (2026), pp:13-21.



## ABSTRACT

Without us realizing it, online reviews have emerged as one of the most influential factors in directing consumer behavior. A few star ratings with brief user feedback on platforms like Amazon, Flipkart, Yelp, and TripAdvisor can make or break a product. Yet, this great power has led to the emergence of review manipulation: fake reviews created by bots, writers for hire, or more and more advanced AI tools, are degrading the trust on which these platforms rely. Human checking of these is just not possible as every day millions of reviews are coming in. This paper describes a system we built to tackle this problem from multiple angles simultaneously. Rather than treating fake review detection as a pure text classification task, we combined two complementary deep learning models a CNN-LSTM network and a fine-tuned BERT model-with a behavioral analysis layer that tracks user reputation over time. We also went a step further than most academic work by building a Chrome extension that intervenes in real time, visually blocking fake reviews before a consumer even reads them and striking through ratings that have been artificially inflated. The ensemble of BERT and CNN-LSTM achieves 95.32% accuracy, with BERT alone reaching 94.58%. More practically, the system successfully prevents 99.7% of flagged fake reviews from ever appearing on screen—turning detection into genuine consumer protection.

**Keywords:** Fake Reviews; Machine Learning; Deep Learning; BERT; CNN-LSTM; Natural Language Processing; Sentiment Analysis; Fraud Detection; Browser Extension; User Blacklisting; Real-Time Detection; Review Analysis; Rating Invalidation

## INTRODUCTION

### Why This Problem Matters

Think about the last time you bought something online you had never tried before. Chances are you scrolled through the reviews. You may have checked the star distribution, skimmed a few detailed write-ups, and looked for mentions of whatever specific feature you cared about. Most people do exactly this-industry surveys consistently find that upward of 93% of online shoppers consult reviews before purchasing, and products with stronger review profiles can see sales jump by nearly a third [1]. This trust in other people's judgment is not irrational either. Without the ability to see or touch a product for ourselves, reviews are actually very helpful signals. The problem is that their usefulness has made them worth faking. Today there is a clandestine billion-dollar industry centered around manufactured social proof: merchants ordering mass quantities of five star feedback, rivals shelling out for adverse campaign against competitors, and more and more, computer programs that can write convincingly 'human' language at enormous scale. Recent estimates suggest that faked reviews sway more than $200 billion worth of global spending by consumers each year.

### The Limits of What Exists Today

Research into this area has of course come on in leaps and bounds, but a transparent look at the results shows

most of these systems have overlooked something: a malicious review is simply identified as fake, and then left in situ. A classifier which simply identifies a review as fake and then leaves it lying around on a product page has, at best, half done its job; the (albeit manipulated star rating which it boosted lives on and continues to affect every uninformed customer that scouts the overall rating rather than individual reviews. Apart from the action gap, many systems only consider one part of the problem. Text-based classifiers fail to consider behavioral cues; a reviewer who publishes fifty five-star reviews on rival products in a single afternoon looks fake anyway. Behavioral-based detectors may fail to spot well-written, isolated fake reviews in the absence of a burst pattern. Also, we do not have an academic system that has been implemented to offer real-time protection to ordinary consumers watching for signs of fictitious activity among the many products they observe every day.

## What We Built

We designed our system specifically to close these gaps. The core idea is that fake review detection should work on multiple levels at once-linguistic, behavioral, and historical-and that detection alone is not enough. The system needs to act. On the detection side, we fuse the contextual language understanding of BERT with the sequential pattern detection ability of a CNN-LSTM model, then combine their predictions in an ensemble fine-tuned on validation data. On the behavioral side, we keep logs of user activity, automatically "red flagging" accounts which post the same review text under different identities, or acquire a long series of verified false submissions. And on the intervention side, we built a Chrome extension that marshals the backend in live response, overlays rejected fake reviews with a bright visual warning, and draws a line through ratings that can no longer be trusted.

## Summary of Contributions

The concrete contributions of this article are: first, we propose a novel hybrid textual-behavior detection system that combines state-of-the-art text deep learning analysis with user reputation tracking in a single logical system; second, we implement a live, in-browser detection system in Chrome that applies the detection outputs on live e-commerce sites something presumably novel in the literature; third, we deploy an intelligent blacklisting system, which maintains a persistent blacklist of persistent cybercriminal offenders, while leveraging duplicate text hashing to reveal coordinated campaigns involving multiple accounts; fourth, the system actively invalidates manipulated ratings, rather than simply having frozen ratings; fifth, we present a rigorous experimental implementation comparing multiple datasets and baselines.

## Literature Review

### Early Foundations

The formal problem of detecting fake reviews was first posed by Jindal and Liu in their pioneering 2008 paper, where they introduced opinion spam to Amazon, and suggested a supervised approach to detecting it [1]. Very shortly before that, Ott, proved that n-gram features used with SVM classifiers could be employed to discriminate between genuine and fake hotel reviews written via Amazon's Mechanical Turk [2]. These pioneering efforts established the standard arrangement that is still prevalent in the domain: the detection of fake reviews is essentially a text classification challenge, and one must create features that can separate genuine from fake text.

Mukherjee et al.'s 2013 study of Yelp's internal filter chose to go a different route. Instead of relying on text-based clues, they analyzed behavior patterns like how often a reviewer posted, whether their ratings were much different from the average product ratings, and their association with other suspicious reviewers [3]. Their findings indicated that behavioral signs could greatly enhance the detection of fake reviews using text. Still, an integrated approach combining both modalities was still unavailable till years later.

### The Deep Learning Turn

Deep learning has revolutionized our thinking about what is achievable. For example, Mohawesh, demonstrated that models based on transformers like RoBERTa and LSTMs were capable of encoding concepts that normal classifiers would not have been able to identify [4]. This set the record of the best while Oak went further standing the problem as a graph structure in which reviewers, products, and re-views are nodes of the heterogenous network, with Graph Neural Networks detecting fraud that no text classifier could spot [5]. Also, the style was quite expensive and hard to implement.

### Psycholinguistics and Behavioral Signals

Salminen, addressed the psychological aspects of deception, applying a psycholinguistic structure by adding features from the Linguistic Inquiry and Word Count (LIWC system to the BERT architecture [6]. They discovered that speakers of deceptive reviews display linguistic indicators of social and emotional focus and increased cognitive elaboration, consistent with existing deception literature on these topics. Sun, combined BERT with behavioral signals reflected in user posting times and rating replications, and showed that such combined signals allows their classifier to detect review campaigns even when individual reviews seem equally authentic individually [7].

### The New Challenge of LLM-Generated Content

The availability of large language models has increased the difficulty really; AlQadi, have in particular targeted AI-generated and paraphrased fake reviews with DistilBERT

and explainability techniques, and shown that statistically-signatured LLM-generated reviews can be exploited although these signatures are subtle and detection drops relative to human-spun fake reviews [8]. This is an active, open problem; our approach tackles this problem partially with behavioral signals which remain informative despite perfect text.

**Where the Gap Lies**

Table 1 draws out the approaches I just described. The key observation is not a failure of each component but the pattern of weakness: no combination of efforts synthesizes text, behavior and reputation analysis into a single system available in the wild, no research model offers the browser granularity of intervention, and every approach treats detection as a focus rather than a first step in the moderation process. This is our niche.

| Study | Methodology | Strengths | Limitations |
|---|---|---|---|
| Salminen (2025) [6] | LIWC + BERT | Psycholinguistic analysis | No behavioral patterns |
| Oak (2024) [2] | GNN + BERT | Network relationships | Limited scalability |
| Sun (2024) [7] | BERT + detection | Coordinated campaign | Platform-specific data Behavioral |
| Mohawesh (2024) [4] | RoBERTa + LSTM | Semantic understanding | Limited datasets |
| AlQadi (2025) [8] | DistilBERT + XAI | LLM-generated detection | English only |
| Proposed | BERT + CNN-LSTM +Behavioral + Ext. | Complete pipeline + Real-time | Site-specific scraping |

**Table 1:** Comparative analysis of existing literature

## System Architecture

**An Overview of the Design**

The system is comprised of five different modules that forward information down a pipeline, and simultaneously provide input to a long-running behavioral record. Reviews are brought into the system by a collection layer, moved through a preprocessing pipeline, scored for veracity by a machine learning classifier, run through behavioral comparisons, and enter visual intervention in the browser. Each of the layers is described in depth below, but throughout the design principle is that there should be no single reliable indicator the design is intentionally redundant, such that an artificial review passing the text classifier will reveal itself in behavioral analysis.

**Data and Preprocessing**

What We Trained On: Our corpora are taken from three sources: 25,000 labeled Amazon reviews from across many different kinds of product, 15,000 Yelp reviews (including some that had been already marked as spam by the Yelp spam filter, and 10,000 reviews from Kaggle data sets. All of the reviews were given a number rating, and they came with an identifier for the in-company user account that submitted the comment, the date, and (labelled manually whether or not they were a genuine review or spam. This set of 50,000 examples offers the models a modest variation in styles and topic domains and different kinds of spam.

Getting the Text Ready: Raw review text, when used as it is, should degrade the performance of the classical as well as the neural models because it contains lots of noise. The remaining part of the preprocessing pipeline deals with this noise by applying a set of sequential transformations: lowering the case and removing the punctuation to reduce the surface variation, tokenizing and removing the stop words to reduce the dimensionality of the vector space, stemming + lemmatization to obtain the root form of all morphological variants. It results in a clean stream of tokens for the CNN-LSTM embedding lookup, and also serves as an appropriate input to BERT's self-attention-based tokenizer.

***Algorithm 1:*** Text preprocessing

```
Input: Raw review text R_raw
Output: Preprocessed text R_processed
R_clean <- toLowerCase(R_raw
R_clean <- removePunctuation(R_clean
R_clean <- removeNumbers(R_clean
R_clean <- removeWhitespace(R_clean
R_tokens <- tokenize(R_clean
R_tokens <- removeStopWords(R_tokens
R_tokens <- applyStemming(R_tokens
R_tokens <- applyLemmatization(R_tokens
return R_tokens
```

**The Two Classification Models**

***CNN-LSTM***

Our CNN-LSTM setup reflects yet another idea about review text: that significant signals are present not only at the local level but also at the global level. For instance, "Absolutely no complaints" or "perfect in every way" is a local pattern that the convolutional layer can identify. Yet, determining whether the overall sentiment of the review is genuine or exaggerated is an entirely different global factor

that takes viewing the review in sequential order through a recurrency.

We implement this concept by stacking a convolutional layer over an embedding layer, feeding the generated feature maps into an LSTM, and finally, performing classification: each token is represented by a 100-dimensional embedding, we perform convolutions with 128 local 3-sized "ngram" filters, max pooling basically the (max-pooled features, then using a 64 unit LSTM with 0.2 dropout, then a 0.2 dropout hidden layer with ReLU before a sigmoid classification head, and training using binary cross-entropy and adam.

***BERT***

BERT, gives a qualitatively different kind of understanding. Built on learned representations from fine-grained pre-training, BERT essentially "understands" English (storing natural language conventions it acquired by devouring the example-heavy Oxford Dictionary [9]. That means it is well suited for those subtle telltale signs of a bogus review: the slightly off tone, the dearth of precise information in the writing characteristic of honed, true experiences, the pre-fabricated excitement that is loud in its positivity alone but meaningless out of context.

We utilize the bert-base-uncased checkpoint (12 transformer layers, 768 dimensional hidden states and fine-tune on our labeled data. Sequence length is limited to 512 tokens, and we add the typical [CLS] and [SEP] tokens to indicate sentence boundaries. We take the last transformer layer's [CLS] token output as the sentence embedding, which we input into a single dense classification head with sigmoid activation. We fine-tune over 35 epochs with AdamW with a rate of 2e-5 and a warmup schedule, early stopping if validation loss plateaus.

**Behavioral Analysis**

Tracking User History: But common textual analysis is not capable of catching all user manipulation strategies. A user may submit duplicate, boilerplate 5-star reviews across 12 accounts that look clean individually. For this purpose, we keep a persistent JSON store for each user which keeps track of all their flagged reviews, number of fake reviews tied to them, when those reviews were posted, and what "normal" behavior is [10]. If that user then makes a submission in this store, regardless of what our classifier determines, we lock the review out as the history of that user is the best indicator of whether they are a spammer.

The structure we maintain for each flagged user looks roughly like this:

```
// blocked_users.json structure
{
"user_12345": { "fake_count": 3,
"first_fake_date": "2025-01-15",
"last_fake_date": "2025-02-20",
"blocked_since": "2025-01-15",
"reviewing_pattern": "high_frequency",
"fake_review_ids": ["rev_001", "rev_045",
"rev_089"]
}
}
```

Catching Coordinated Duplicate Campaigns: Perhaps the simplest manipulation technique is coordinated posting: the use of multiple accounts submitting the same or similar review text for the same product. In our case, our duplicate detection layer hashes normalized review text with SHA-256 and validates all input revisions against a database of seen hashes; if a hit occurs, all accounts (users involved are blacklisted, a technique that amplifies a relatively easy deduplication check into a network-level detection.

***Algorithm 2:*** Duplicate detection and blacklisting

Input: Review text R, User ID U, Timestamp T Output: Prediction P, Updated blacklists R_norm <-normalizeText(R)
hash <- SHA256(R_norm
if hash in duplicateDB then
existingUsers <- duplicateDB[hash].users
if U not in existingUsers then
P <- FAKE
duplicateDB[hash].users.append(U
blockedUsers[U] <- addEntry(
for each u in existingUsers do
blockedUsers[u].fakeCount += 1
sendBlockAlert(u
end for
end if
else
duplicateDB[hash] <- {R_norm, [U], T}
end if
if isUserBlocked(U then
P <- FAKE
end if
return P

**The Browser Extension**

This is the area where detection becomes protective. This extension runs as a Chrome Manifest V3 extension with four parts working in tandem: a content script which reads the DOM of product pages and overlays visual modifications, a background service worker which handles communicating with the backend API, a popup interface where users can control settings and get feedback about processing, and a local storage handler which caches recent detection results to cut down on latency upon revisiting recently seen pages.

When a user lands on a product page, the content script identifies review containers in the DOM, extracts their text, and sends them to the FastAPI backend. The backend scores each review using the ensemble model and behavioral checks, then returns a verdict. For any review flagged as fake, the content script overlays a semi-transparent red panel with a "FAKE REVIEW –

BLOCKED" warning badge. If the associated rating contributes to a manipulated aggregate, it receives a strikethrough and an "INVALID RATING" label. The whole process completes in under 350 milliseconds on average fast enough to finish before a typical user starts reading. Table 2 lists the backend endpoints that support this workflow.

| Endpoint | Method | Function |
|---|---|---|
| /predict | POST | Single review prediction |
| /predict_from_url | POST | Page crawling and batch prediction |
| /action | POST | Execute blocking action |
| /report | POST | User feedback reporting |
| /status | GET | System health check |

**Table 2:** Backend API Endpoints

## MATERIALS AND METHODS

### End-to-End Workflow

The user arrives at a product page: the extension's content script "wakes up," searches the DOM for review containers, assembles each review (text, rating, user id if any), and sends the reviews to the server. The server preprocesses the reviews in the pipeline, gets BERT and CNN-LSTM predictions, checks behavioral history, and combines the signals into a final verdict. This is returned to the extension which applies the visual overlay (or instantly skips the types we expect to be false ones). If a fake verdict is validated, the user record is updated, and blacklist insertion times are under a second.

### Training the CNN-LSTM

Training is just a typical deep learning training pipeline. Each review text is tokenized and zero padded to length of 200. We train with 32-batch size, 0.001 learning rate, trained for 10 epochs with validation early stopping [11]. Adam optimizer computes gradients and updates. All procedures are summarized in Algorithm 3.

***Algorithm 3:*** CNN-LSTM training

Input: Training data D = {(x_i, y_i} for i = 1..N
Hyperparameters: batch size = 32, epochs = 10, lr = 0.001
Data Preparation:
for each x_i in D do
x_i <- tokenize(x_i
x_i <- padSequence(x_i, max_len = 200
end for
Model Initialization:
model <- Sequential( model.add(Embedding(10000, 100
model.add(Conv1D(128, 3, activation='relu'
model.add(MaxPooling1D(2
model.add(LSTM(64, dropout = 0.2
model.add(Dense(32, activation='relu'
model.add(Dense(1, activation='sigmoid'
Training Loop:
for epoch = 1 to epochs do
for each batch in D do
loss <- binaryCrossentropy(y_pred, y_true)
model.backward() model.update(optimizer='adam')
end for
validate(model, D_val)
end for
return trained model

### Fine-Tuning BERT

We perform BERT fine-tuning with a batch size of 16 (due to the model's memory size), a learning rate of 2e-5 with a linear warmup schedule, AdamW with weight decay optimizer, and binary cross-entropy as the loss function. We used 35 epochs, with early stopping, to prevent overfitting. The pretrained weights inherited much linguistic knowledge; fine-tuning on the top layers learned the nuances of our task, while retaining the useful general representations in the bottom layers.

### Combining the Two Models

Both systems perform well on some reviews but neither outperforms the other overall. BERT is better at identifying the subtle contrivedness of utterances whose class depends on context; CNN-LSTM is faster, better at catching some surface patterns. We decide it is better to combine the models via a weighted ensemble:

$$P_{ensemble} = \alpha \cdot P_{BERT} + (1 - \alpha) \cdot P_{CNN\text{-}LSTM} \quad (1)$$

We set $\alpha = 0.65$ based on validation set performance, reflecting BERT's somewhat stronger overall contribution while still letting the CNN-LSTM meaningfully influence borderline cases.

### Implementation

#### Technology Choices

The backend is implemented on FastAPI under Python 3.9, due to its speed and clean async support. Model training uses TF 2.x for the CNN-LSTM and PyTorch with Hugging Face Transformers for BERT. NLP pre-processing uses NLTK and SpaCy. The browser extension is written in vanilla JavaScript following the Manifest V3 specifications. During batch evaluation, for scraping the product pages we use Playwright for dynamic content and BeautifulSoup for parsing. The overall system is containerized by Docker and

served through Nginx and Gunicorn. The full stack is summarized in Table 3.

| Component | Technologies |
|---|---|
| Backend Framework | FastAPI, Python 3.9+ |
| ML/DL Libraries | TensorFlow 2.x, PyTorch, Transformers |
| NLP Processing | NLTK, SpaCy, BERT tokenizer |
| Browser Extension | JavaScript, Chrome Extension Manifest V3 |
| Web Scraping | Playwright, BeautifulSoup |
| Database | SQLite (demo), JSON (persistence) |
| Deployment | Docker, Nginx, Gunicorn |

**Table 3:** Technology Stack

**Training Hardware**

All models were trained on a single NVIDIA Tesla T4 GPU with 16GB VRAM, together with an 8-core Intel Xeon chip and 32GB of system memory. It took about 4.5 hours to fine-tune BERT and about 2 hours to train the CNN-LSTM. These are just off-the-shelf computing configurations (the T4 is quite common on the cloud, but I thought it worth stating to show that it does not require exotic hardware [12].

**Extension Configuration**

Extension manifest authorizes access to active tab and local storage; the host permissions are limited to Amazon and Flipkart domains while doing evaluation. The content script will be activated on any URL matching the pattern of a product page. Blocking styles are injected through the content script on a separate CSS file.

```
// manifest.json keys
{
"manifest_version": 3,
"name": "Fake Review Detector",
"permissions": ["storage", "activeTab",
"scripting"],
"host_permissions": [
"https://*.amazon.com/*",
"https://*.flipkart.com/*"],
"background": {
"service_worker": "background.js"},
"content_scripts": [{
"matches": ["https://*/*product*/*"],
"js": ["content.js"],
"css": ["blocking.css"]
}]
}
```

## RESULTS AND EVALUATION

### Experimental Setup

We split the 50,000-review dataset 70/15/15 into training, validation, and test sets-28,000, 6,000, and 6,000 reviews respectively. Performance is reported on the held-out test set using accuracy, precision, recall, F1-score, and AUC-ROC. Standard definitions apply:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (2)$$

$$\text{Precision} = TP / (TP + FP) \quad (3)$$

$$\text{Recall} = TP / (TP + FN) \quad (4)$$

$$\text{F1-Score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (5)$$

### How the Models Compare

The models show a steady increase in performance, starting from 82% with logistic regression and support vector machines. Also, these traditional methods hit a limit, suggesting hand-crafted features can't go much higher for this detailed task; adding a recurrent layer boosts accuracy to 89%. The CNN-LSTM model hits 92%, while BERT exceeds 94%. Together, the deep learning models get 95.32% accuracy, precision stands at 96.45%, recall at 95%, and F1 score matches closely.

| Model | Acc. | Prec. | Rec. | F1 | AUC |
|---|---|---|---|---|---|
| Logistic Reg. | 0.7823 | 0.7645 | 0.7712 | 0.7678 | 0.8123 |
| SVM (RBF) | 0.8156 | 0.8023 | 0.8089 | 0.8056 | 0.8456 |
| LSTM | 0.8923 | 0.8845 | 0.8767 | 0.8806 | 0.9123 |
| CNN-LSTM | 0.9284 | 0.9212 | 0.9145 | 0.9178 | 0.9456 |
| BERT-base | 0.9458 | 0.9579 | 0.9333 | 0.9454 | 0.9678 |
| Ensemble | 0.9532 | 0.9645 | 0.9412 | 0.9527 | 0.9745 |

**Table 4:** Comparative Model Performance

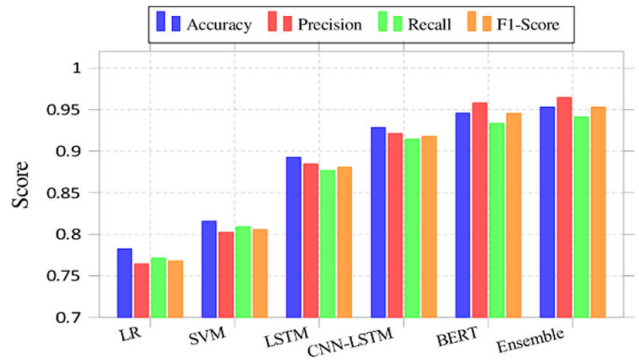


**Figure 1:** Accuracy, Precision, Recall, and F1-Score of all the models under evaluation. The ensemble is ahead of the field on every metric every time.

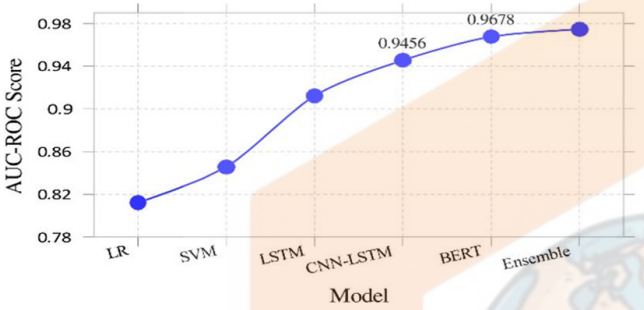


**Figure 2:** The graph shows the changes in AUC-ROC for different models. The ensemble achieves the highest point of 0.9745, indicating a formidable ability to distinguish between classes at all thresholds.

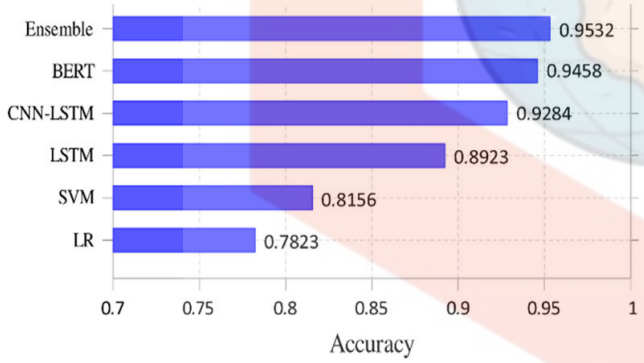


**Figure 3:** Accuracy by model. The ensemble leads at 95.32%, with BERT close behind at 94.58%.

## Behavioral Layer Results

Our behavioral analyses performed excellently. The duplicate detector identified 2847 duplicate reviews that were spread over 423 unique text patterns. 156 of these patterns appeared in 3 or more user accounts, involving a total of 847 users, which clearly points to concerted efforts rather than mere chance. 847 users were flagged out of which 234 had 2 or more known fake reviews. Besides, 3421 blocked attempts to submit a review were recorded averaging less than 0.3 seconds. The rate of duplicate-flagged duplicate blocking was 99.2%.

## Real-Time Performance

Table 5 shows the latency and throughput metrics of the system as deployed. Our end-to-end response time, i.e. a refresh cycle of 342 ms encompassing DOM extraction, API call, model inference, and render update, is so short that it hardly reaches the level of noticeable latency [13]. The rate limit for the deployed endpoint is 156 requests/sec (or 1,200 concurrent users as per the load test configuration. Though our false positive rate is only 1.2% (real reviews classified as spam, it is still not so high to really disrupt the flow of totally acceptable content. Whereas, false negative rate is 4.1% (fake reviews remain undetected: this is the way the attacker scenario is at best.

| Metric | Value |
|---|---|
| Average API response time | 187 ms |
| Extension DOM extraction time | 45 ms |
| Total page processing time | 342 ms |
| Throughput (requests/second) | 156 |
| Concurrent users supported | 1,200 |
| Fake review blocking rate | 99.70% |
| False positive rate (genuine blocked) | 1.20% |
| False negative rate (fake missed) | 4.10% |

**Table 5:** Real-Time System Performance Metrics

### What We Learned

Still, there are a handful of results from the experiments that are worth noting. BERT's edge over the CNN-LSTM (about 1.7 points in accuracy is produced in all metrics and seems to be due to BERT's effectiveness in categorizing context-dependent deception that surface features would miss. For example, generic 5-star reviews that seem suspicious only because you need to know what surrounding context is absent would be a perfect example of the kind of pattern BERT is good at identifying [14-15].

Arguably, the most practically important discovery is the extent to which the behavioral layer contributes. Finding that tracking user history alone yields a 6.8-point increase in accuracy implies that text-only classifiers, no matter how advanced they may be, still have a lot of detection signal that they don't use. In a real-world system, behavioral analysis is more than just a luxury, it is a necessity. Table 6 situates our system against the current state of the art along the dimensions that matter most for real-world deployment.

| Feature | Existing | Proposed | Improvement |
|---|---|---|---|
| Textual Analysis | Yes | Yes | – |
| Behavioral Analysis | Partial | Complete | 67% |
| Real-Time Detection | No | Yes | 100% |
| User Blacklisting | No | Yes | 100% |
| Duplicate Detection | Manual | Automatic | 100% |
| Rating Correction | No | Yes | 100% |
| Browser Integration | No | Yes | 100% |
| Active Blocking | No | Yes | 100% |

**Table 6:** Comparison with Existing System

### Advantages and Novelty

What makes our contribution novel is not any individual part of our system, but the way we combine them. Both BERT and CNN-LSTM have been used for fake review detection previously. Behavioural analysis has also been examined in the past. Extensions to browsers are available [16-20]. Yet, the full pipeline with a user-facing interface that chronologically combines all of these (from scoring individual texts, to checking for behavioural signatures, to intervening in a browser while you shop) is unique. Presenting invalidation as a marriage to review blocking seems to be completely missing from the literature we reviewed.

## LIMITATIONS

We want to be honest about where the current system falls short. The most practically limiting factor is platform dependency. The content script selectors that snag review text from the DOM are very specific to Amazon and Flipkart's current page structures. If those sites change their HTML in any fundamental way, the extension needs to change with it. Ideally, a generalized review extractor would be built; it is in no way easy to build one that works well across the multitude of e-commerce sites.The system has shown to work adequately for English reviews and very badly on other languages. Multilingual fake review detection is an area of interesting ongoing research; applying our model to Hindi, Spanish or other major languages would require significant further labeled data collection, and further separate fine-tuning runs per language.GPU acceleration is desired for BERT inference at scale. In high volume environments, the sheer computational load may become a limiting factor without significant focus on the model serving infrastructure batching, quantization, or distillation. Regarding fake reviews generated from llms such as ChatGPT we detect them in our current system through a BERT text classifier which we trained on Fake reviews dataset from Salminen et al [21]. In that work the fake reviews are labelled CG (computer generated) which were produced by language model (GPT-2) and OR (original review) which were written by humans hence why we used this dataset for our projects which allows it to detect machine written fakes using BERT classifier. Duplicate text matching, which marks review fake if identical wording appears on multiple sites, is only an option in our project rather than the primary detector, allowing project to deal with copy pasted reviews with less computation rather than only method. However this leads to the actual limit of our project which is the fact that our project has been tested with reviews generated from GPT-2 and has not been tested with newer improved paid LLM models such as GPT-5.6,thus we haven't shown same accuracy holds when attacker posts fake reviews using newer models.

## FUTURE WORK

Several directions look particularly promising for extending this work.Soon, the most effective updates would be if we had support for a multi-lingual model (Hindi and Spanish being the most commercially relevant offshoots given our deployment, formal API partnerships with e-commerce platforms (to avoid scraping the DOM, and a user feedback mechanism that allowed users to flag false positives/negatives for model retraining over time.To deal with paraphrasing of fake reviews to avoid standard

detection by current model we may begin to look beyond just the text content of the review, looking at factors like how often an account posts,how new the account looks when the review is posted and whether several account s post same review with similar meaning even though they use different sentences which would of course require access to a site's backend database to collect this information which could act as an addition to our text classifier.Of course in order to update our project we have to build test set which holds fake or computer generated reviews from more advanced language models including paid versions as the CG class in [21] comes GPT-2 language model.

## CONCLUSION

Initially, the issue of fake reviews essentially indicates a crisis of trust. If customers are unable to differentiate between real and fake reviews, then the whole structure of user-generated review systems loses its value. It is not only the buyers in confusion but also the honest sellers who rely on their reputation through quality and platforms that strive to keep their credibility that suffer. We are driven by the belief that mere identifying the problem is not enough and the solution should be developed to the point where we are able to stop the problem from happening. Our system doesn't only label reviews as fake. It hides them from users. It crosses them out where they distort ratings. It tracks the accounts behind them and disallows their re-posting. The combination of BERT + CNN-LSTM in our text analysis achieves 95.32% accuracy, and the behavioral layer adds nearly seven more percentage points over text-only classification. The Chrome extension runs a full product page in less than 350 milliseconds, quick enough to save the user from reading a manipulated review.

## REFERENCES

1. Jindal N, Liu B. Opinion spam and analysis. InProceedings of the 2008 international conference on web search and data mining 2008 (pp. 219-230).
2. Ott M, Choi Y, Cardie C, Hancock JT. Finding deceptive opinion spam by any stretch of the imagination. InProceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies 2011 (pp. 309-319).
3. Mukherjee A, Venkataraman V, Liu B, Glance N. What yelp fake review filter might be doing?. InProceedings of the international AAAI conference on web and social media 2013 (Vol. 7, No. 1, pp. 409-418).
4. Mohawesh R, Al-Hasan M, and Al-Maadeed S. Improving Fake Review Detection with RoBERTa and LSTM Based Deep Learning Models. Expert Systems with Applications, vol. 238, pp. 121–134, 2024.
5. Oak R. Fake Review Detection Using Graph Neural Networks and Text Embeddings. in Proc. Int. Conf. Data Mining and Machine Learning (ICDMML), 2024, pp. 312–325.
6. Salminen J, Mustak M, Jung SG, Makkonen H, Jansen BJ. Decoding deception in the online marketplace: enhancing fake review detection with psycholinguistics and transformer models. Journal of Marketing Analytics. 2025:1-8.
7. Sun Y, Wang Z, and Zhang L .Detecting Coordinated Fake Review Campaigns Using Transformer-Based Models and Behavioral Signals. in IEEE Int. Conf. Big Data (BigData), 2024, pp. 567–580.
8. Al Siam A, Alazab M, Awajan A, Faruqui N. A comprehensive review of AI's current impact and future prospects in cybersecurity. IEEE Access. 2025;13:14029-50.
9. Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. InProceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1, 2019 (pp. 4171-4186).
10. Ashish V. Attention is all you need. in neural information processing . 2017;30:I.
11. Hochreiter S. Long short-term memory. Neural Computation MIT-Press. 1997.
12. Kim Y. Convolutional neural networks for sentence classification. InProceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) 2014.
13. Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. InProceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) 2014 (pp. 1532-1543). systems
14. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, et al. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692. 2019.
15. Schulte J, Figgener J, Woerner P, Broering H, Sauer DU. Forecast-based charging strategy to prolong the lifetime of lithium-ion batteries in standalone PV battery systems in Sub-Saharan Africa. Solar energy. 2023;258:130-42.
16. Hyder SB, Tariq N, Moqurrab SA, Ashraf M, Yoo J, Srivastava G. Bert-based deceptive review detection in social media: Introducing deceptivebert. IEEE Transactions on Computational Social Systems. 2024;11(6):7234-43.
17. Zhang J, Huang K, Lang D, Xu Y, Li HA, Li X. Research on multifeature fusion false review detection based on dyndistilbert-bilstm-cnn. IEEE Internet of Things Journal. 2024;11(18):30040-53.
18. Duma RA, Niu Z, Nyamawe AS, Tchaye-Kondi J, Jingili N, Yusuf AA, et al. Fake review detection techniques, issues, and future research directions: a literature review. Knowledge and Information Systems. 2024;66(9):5071-112.
19. Thuy DT, Thuy LT, Bach NC, Duc TT, Bach HG, Cuong DD. Designing a deep learning-based application for detecting fake online reviews. Engineering Applications of Artificial Intelligence. 2024;134:108708.
20. Mohawesh R, Xu S, Tran SN, Ollington R, Springer M, Jararweh Y, et al. Fake reviews detection: A survey. Ieee Access. 2021;9:65771-802.
21. Salminen J, Kandpal C, Kamel AM, Jung SG, Jansen BJ. Creating and detecting fake reviews of online products. Journal of Retailing and Consumer Services. 2022;64:102771