Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Online Fake Review Detection Using Deep Learning

M.G. Varalakshmi[1], Dilip S[2], Deepith N[3], Darshan L[4], Hemanth Naik S[5]

[1]*Assistant Professor, Dept Of CSE, ACS College Of Engineering, Bengaluru, India*

[2] *Dept of CSE Student, ACS College Of Engineering, Bengaluru, India*

[3]*Dept of CSE Student, ACS College Of Engineering, Bengaluru, India*

[4] *Dept of CSE Student, ACS College of Engineering, Bengaluru, India*

[5] *Dept of CSE Student, ACS College of Engineering, Bengaluru, India*

Abstract- Fake online reviews have become a serious issue in the digital economy, significantly influencing consumer decisions and business credibility. As e-commerce platforms continue to grow, deceptive reviews generated by humans or AI systems attempt to manipulate product ratings and customer perception. Traditional keyword-based filtering and behavioural detection methods are insufficient against modern AI-generated spam. This paper presents a dual-approach framework combining

Classic Machine Learning techniques and Deep Learning using BERT for effective fake review detection. The system performs preprocessing, feature extraction, model training, evaluation, and deployment through a web interface. Experimental results demonstrate that the fine-tuned BERT model achieves 98% accuracy, outperforming traditional machine learning models. The proposed system ensures improved reliability, contextual understanding, and real-time classification of reviews.

***Key Words*: Fake Review Detection, Deep Learning, BERT, Natural Language Processing, Opinion Spam, Machine Learning.**

1.INTRODUCTION

In the web world today, what buyers say decides if others will trust a company. While e commerce keeps expanding, it also pulls in more misuse. Sneaky users post false reviews either to hype up products or harm competitors by slamming them unfairly. This flood of misleading data confuses real customers and hurts honest businesses trying to play fair. Years back, platforms relied on manual checks or basic keyword filters to catch fakes, but those outdated methods struggle with today's cleverer fraud waves.

To solve this issue, the project uses a tool that automatically finds fake reviews. The core relies on Natural Language Processing - this helps computers grasp text similar to people. We start by collecting actual user feedback, then tidy it up carefully so everything looks uniform. Next, rather than jumping forward, we grab essential features using techniques such as TF-IDF or clever ways to map words. This numerical data feeds into models trained to notice small clues separating genuine thoughts from paid posts.

Detecting fake reviews is tough - scammers tweak their wording constantly to appear genuine. What seems like honest input at first might actually be a sneaky forgery. Traditional methods fall apart under floods of posts by the thousands. Better machine learning systems may step in, yet rely heavily on high-quality samples to deliver results. These setups need powerful machines just to work right - otherwise they lag hard.

What's the aim? Build a tool that catches lies quickly, yet stays accurate at the same time.

1.1 Problem Statement

The core problem addressed by this project is the integrity of online feedback systems.

Volume: Popular products receive thousands of reviews daily. Manual moderation is computationally and financially unfeasible.

Sophistication: Traditional spam detection methods rely on "Keyword Filtering" (e.g., blocking words like "scam" or "buy now") or "Metadata Analysis" (e.g., tracking IP addresses). These methods are obsolete. Modern spammers use proxy servers to hide IPs and use sophisticated language models to write reviews that bypass keyword filters.

Context Blindness: Classic Machine Learning models (like Naive Bayes) treat words as independent units. They fail to understand sarcasm or complex sentence structures. For example, "I loved how the screen shattered on the first drop" contains the positive word "loved," which might trick a simple model, but a human (and our proposed Deep

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Learning model) understands the negative context.
The system must accurately classify reviews into:
Original Review (OR)
Computer Generated (CG)

2.EXISTING SYSTEM

Current industry standards for fake review detection largely rely on:

Rule-Based Systems: These systems flag reviews based on simple heuristics, such as extreme rating deviation (e.g., giving 1 star when the average is 5) or the presence of blacklisted keywords.
Behavioral Analysis: Platforms analyze user behavior, such as posting frequency or burstiness (posting many reviews in a short time).
Shallow Learning: Some systems use basic Machine Learning models like Support Vector Machines (SVM) or Naive Bayes.

3.PROPOSED SYSTEM

We propose a Deep Learning-based approach using the BERT (Bidirectional Encoder Representations from Transformers) architecture.

Contextual Understanding: Unlike previous systems that read text sequentially (left-to-right), BERT reads the entire sequence of words at once (bidirectionally). This allows it to understand the specific context of a word based on its neighbours.
Transfer Learning: We utilize a pre-trained model that has already ”read” millions of documents (Wikipedia, BooksCorpus). We then ”Fine-Tune” this model specifically on our Fake Reviews dataset. This results in a highly accurate model even with a limited amount of labelled training data.
Automated Feature Extraction: The proposed system does not require manual feature engineering (like counting adjectives). The Deep Learning model automatically learns the linguistic patterns and ”Deception Cues” hidden in the text.

4. OBJECTIVE

The specific objectives of this project are:

- To collect and preprocess a labeled dataset of Original and Computer-Generated reviews.
- To implement TF-IDF (Term Frequency-Inverse Document Frequency) vectorization for baseline model training.
- To implement BERT Tokenization and fine-tune the bertbase-uncased model for binary classification.
- To evaluate and compare the models based on Accuracy, Precision, Recall, and F1-Score.
- To develop a user-friendly Web Dashboard using Flask that allows end-users to verify the authenticity of reviews in real-time.

4.LITERATURE SURVEY

Previous research has explored various methods for fake review detection. Ott et al. introduced linguistic deception cues. Mukherjee et al. analyzed reviewer behavior and burst patterns. Rayana and Akoglu used graph-based detection approaches. Recent works implemented hybrid deep learning and attentionbased models. Transformer-based architectures such as BERT have shown significant improvements due to contextual embeddings and bidirectional processing capabilities.

6. SOFTWARE REQUIREMENTS

- Operating System: Windows 10/11 or Linux.
- Programming Language: Python 3.8 or higher.
- Deep Learning Framework: PyTorch.
- NLP Library: Hugging Face Transformers.
- ML Library: Scikit-learn.
- Data Handling: Pandas, NumPy.
- Web Framework: Flask.
- IDE: Visual Studio Code / Jupyter Notebook.

7. HARDWARE REQUIREMENTS

- Processor: Intel Core i5 / AMD Ryzen 5 (Quad-core) or better.
- RAM: Minimum 8GB (16GB recommended).
- GPU: Development: NVIDIA Tesla T4 (via Google Colab) is required for training the BERT model efficiently. Deployment: Can run on a standard CPU, though inference will be slower.
- Storage: At least 10 GB of free disk space to store the dataset and the saved model weights (.pth file, approx 450MB).

8. METHODOLOGY

The methodology follows a complete ML pipeline:

- Data Preprocessing: Removal of HTML tags, URLs, special characters, and lowercasing text.
- Feature Extraction: TF-IDF for classic ML and BERT Tokenization for Deep Learning.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

- Model Training: Random Forest baseline and fine-tuned BERT (4 epochs, AdamW optimizer).
- Evaluation Metrics: Accuracy, Precision, Recall, F1Score, Confusion Matrix.

9. SYSTEM TOOLS

Python: Python was selected as the primary programming language because of its dominance in the Data Science field. It offers a vast ecosystem of libraries for NLP, Machine Learning, and Web Development, making it the ideal ”glue” language for this project.
Flask: Flask is a micro-web framework written in Python. Unlike Django, it is lightweight and flexible. We used Flask to create the API endpoints (/predict) that connect our HTML frontend with our PyTorch backend. It handles HTTP requests (GET/POST) and renders templates.
PyTorch and Transformers: PyTorch: An open-source machine learning library developed by Facebook’s AI Research lab. It provides the tensor computation and automatic differentiation required for Deep Learning.
Transformers: The Hugging Face library provides state-ofthe-art pre-trained models. It allowed us to download and implement the BERT architecture with just a few lines of code, saving months of development time.
Scikit-learn: Used for the Classic Machine Learning implementation. It provided the TfidfVectorizer for feature extraction and the RandomForestClassifier and LogisticRegression algorithms. It also provided the metrics module for calculating the Confusion Matrix.
NumPy: Numerical Python (NumPy) is used for handling large, multi-dimensional arrays and matrices. It is the fundamental package for scientific computing in Python and is used internally by PyTorch and Scikit learn.
Pandas: Pandas is used for data manipulation and analysis. We used it to load the dataset (CSV files) into DataFrames, allowing for easy cleaning, sorting, and splitting of the data.
Matplotlib: A plotting library used to visualize the training progress. We used it to plot graphs showing the ”Training Loss” vs. ”Validation Loss” over the 4 epochs, helping us visualize the model’s learning curve.
JSON and OS libraries: The OS library is utilized to manage file paths and directory operations, ensuring the seamless loading of the trained model and dataset across different operating systems. Additionally, the JSON library handles the formatting of server responses and configuration data, facilitating efficient data exchange between the backend and the web interface.

10. SYSTEM DESIGN

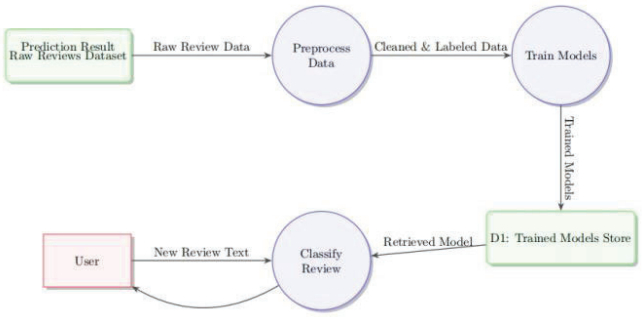

Fig: System Architecture

The system follows a Model-View-Controller architecture:

- Model: This is the Deep Learning component (BERT). It resides in the backend and is responsible for the logic of classification. It accepts numerical tensors and outputs probability scores.
- View: This is the User Interface, built using HTML and CSS. It presents the input form to the user and displays the final results (Real/Fake badges).
- Controller: This is the Flask Application (app.py). It accepts input from the View, processes it using the Model, and returns the result to the View.

The system is modularized into specific Python scripts to ensure maintainability:
train.py: This script is responsible for the offline training process. It loads the dataset, trains the BERT model, and saves the weights to fake review detector.pth.
app.py: This is the main entry point for the web application. It loads the saved model into memory on startup and defines the URL routes.
inference.py: A helper module containing the predict sentiment function, which handles the tokenization and tensor conversion logic.

11. SYSTEM TESTING

System testing evaluates the complete, integrated software to ensure it meets all specified requirements before deployment. We verified the interaction between the Deep Learning model and the web interface to guarantee accurate predictions. The process focused on validating functionality, error handling, and the overall reliability of the fake review detection system.
Testing was conducted on a Windows 11 machine with an Intel Core i5 processor and 16GB RAM to handle model inference. We used Python 3.10 with PyTorch and Flask libraries, ensuring a stable and consistent software stack. The web interface was verified using the Google Chrome browser to ensure proper rendering and responsiveness.

Types of Testing Performed are:

- Unit Testing: We tested individual functions like text cleaning and tokenization in isolation to ensure they process data correctly. This step verified that the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

preprocessing logic removes noise and converts text to IDs without errors. It ensures the foundational blocks of the code are bug-free before being integrated into the larger system.

- Integration Testing: This phase verified the communication between the Flask web server and the PyTorch backend model. We ensured that user input from the browser is correctly passed to the inference engine and results are returned. It confirms that the frontend and backend components exchange data seamlessly without format mismatches.
- System Testing: We evaluated the entire application as a whole to validate end-to-end functionality against user requirements. This included stress testing with various input types to ensure the system handles errors gracefully without crashing. Latency checks confirmed the model delivers predictions within the acceptable two-second timeframe.

Performance evaluation results:

TABLE I

MODEL ACCURACY COMPARISON

Model	Accuracy
Logistic Regression	86%
Random Forest	85%
BERT	98%

12. RESULT

The Deep Learning BERT model significantly outperformed traditional machine learning models. The system successfully classifies reviews in real-time through a web interface. False negatives were considerably reduced compared to baseline models.

13. CONCLUSION

The ”Online Fake Review Detection” project successfully addresses the critical challenge of opinion spam in ecommerce by deploying a robust ”Dual-Approach” methodology. Through a rigorous comparison between Classic Machine Learning and Deep Learning, the study conclusively proves the superiority of the BERT Transformer model. While traditional algorithms like Logistic Regression provided a baseline accuracy of 86model achieved an exceptional 98semantic context and linguistic nuances that simpler models miss.

REFERENCES

[1] M. Ott joined forces with Y. Choi and C. Cardie - alongside J.T. Hancock - who explored new ways to uncover fake reviews at the 49th ACL gathering in 2011.

[2] Mukherjee, A. Kumar, plus B. Liu teamed up with J. Wang - and N. Glance - to present a paper called ”Finding Fake Reviewers in Customer Feedback” during the 2012 International WWW Conference, held that year as its 21st edition.

[3] S. Rayana worked alongside L. Akogbuo, hunting down false reviews by studying how users link up as well as their behavior clues - this took place during the 21st KDD meetup centered on uncovering meaning in data around 2015.

[4] B. K. Rout and R. K. Dash, “Fake Review Detection using Machine Learning and Natural Language Processing: A Survey,” Procedia Computer Science, Elsevier, 2021.

[5] N. Jindal with B. Liu looked at fake reviews in a 2007 paper presented during a web focused event - this work focused on spotting lies online through fast-alert methods rather than broad assumptions or generic solutions.

[6] Kumar along with R.S. Thakur tried out deep learning methods to spot fake product reviews on online shopping platforms - this study showed up in the International Journal of Information Management Data Insights in 2022.

[7] G. Fei teamed up with A. Mukherjee - rather than going solo - to roll out a method using sudden spikes in review numbers, along with help from B. Liu, M. Hsu, and M. Castellacos, to spot fake reviews; R. Ghosh hopped on board afterward. The work popped up at the 7th AAAI workshop on blogs, linked to social platforms, in 2013.

[8] J. Xuan, Y. Ma, along with J. Jiang teamed up with several others to test spotting fake reviews using semi-supervised learning plus attention mechanisms; their work showed up in IEEE Transactions on Computational Social Systems in 2020.

[9] Y. Li, J. Zhang, Q. Chen - along with X. Li - explored methods to spot fake news and misinformation in a 2020 paper from ACM Transactions on Management Information Systems. .

[10] H. Li, Z. Chen, B. Liu, and X. Wei - along with J. Shao - tested how graphs can catch fake reviews when combined with different clues; their work appeared in Knowledge-Based Systems through Elsevier in 2021.