# Spam and Fake Review Detection Using Deep Learning

Yusheng Yang, *yy115,* Shelton Zhou, *sz105,* and Chuyang Wang, *cw162*

***Abstract*—Fake online reviews undermine trust, mislead consumers, and create unfair competition. This proposal focuses on developing a machine learning system to detect and prevent fraudulent reviews, helping platforms ensure authenticity and fairness. Using advanced algorithms, the model will adapt to evolving tactics, providing reliable, long-term results. By restoring trust in online feedback, this project supports ethical business practices, empowers consumers to make better decisions, and fosters a transparent digital marketplace.**

***Index Terms*—Deep Learning, LSTM, GloVE 100D, CNN-LSTM, Doc2Vec, Logistic Regression, TF-IDF.**

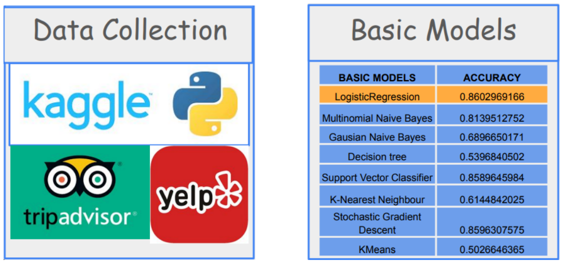
## I. Background/Motivation

Fake reviews harm consumers and brands, creating unfair competition and eroding trust. Traditional methods fail to adapt to evolving fraud tactics or address semantic complexity. Deep learning models like CNNs and LSTMs offer promising solutions but are underused in this context.

This project combines CNNs, LSTMs, and logistic regression to detect fake reviews using linguistic, behavioral, and temporal features, aiming to restore trust and fairness in online marketplaces while advancing fraud detection methods.

We will be using the 20 Chicago hotels dataset[1][2] and the Fake Reviews Dataset[3]. The Trip Advisor and Yelp data for ”Chicago hotels reviews” characterizes whether the review is fake or real, and ”Fake Review Dataset” is from Kaggle which has 40k data of computer-generated reviews and real reviews.

While some studies had similar interests, the problem is not defined exactly the same. Traditional methods like rule-based systems and classical machine learning (e.g., Naive Bayes, SVM) struggle to capture semantic and contextual nuances.


Data Collection
kaggle
tripadvisor
yelp
Basic Models

| BASIC MODELS | ACCURACY |
|---|---|
| LogisticRegression | 0.8602969166 |
| Multinomial Naive Bayes | 0.8139512752 |
| Gausian Naive Bayes | 0.6896650171 |
| Decision tree | 0.5396840502 |
| Support Vector Classifier | 0.8589645984 |
| K-Nearest Neighbour | 0.6144842025 |
| Stochastic Gradient Descent | 0.8596307575 |
| KMeans | 0.5026646365 |

(a) Data Collection

(b) Basic Models

Fig. 1

## II. Experimentation/Model

### A. Data Prepossessing

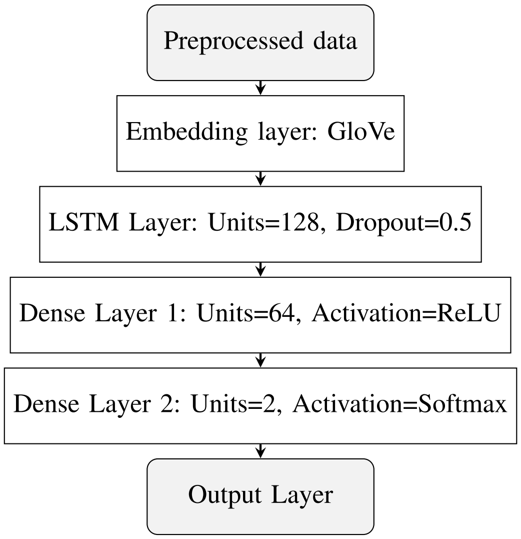
Our team employs a comprehensive approach to preprocessing, data augmentation, and data modification to prepare textual data. Preprocessing involves tokenizing text at both word and sentence levels using NLTK tools, removing stopwords to focus on meaningful terms, and lemmatizing words with the WordNet Lemmatizer to standardize variations in word forms. Additionally, regular expressions are used to clean the data by eliminating punctuation and non-alphabetic characters, ensuring a consistent and clean input format. Categorical labels are converted into numerical representations using the LabelEncoder from scikit-learn, making them suitable for machine learning models. For evaluation purposes, the dataset is split into training and testing subsets using the `train_test_split` function.

### B. Models

*1) Baseline: Logistic Regression:* Using Logistic Regression as the baseline is a good metric to determine whether our subsequent deep learning implementation is successful or not. It characterizes the output into 2 categories which in our standard- they are real comments and fake/computer-generated comments. **It has a 86% testing accuracy.**

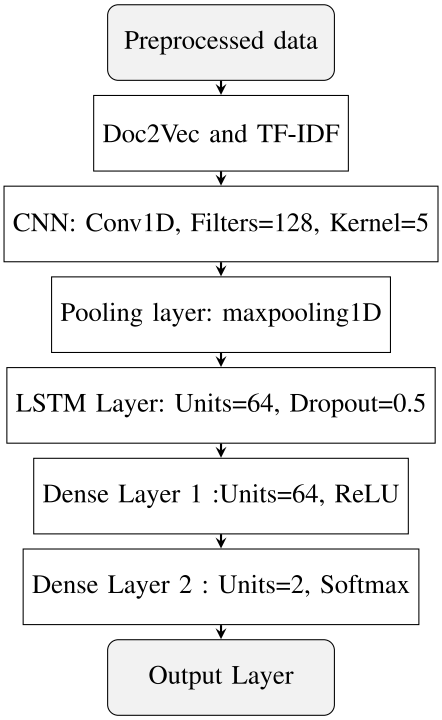
*2) LSTM+GloVe:* In this model, we introduced long-short-term memory (LSTM) and global vectors for word representation (GloVe), where GloVe builds a co-occurrence matrix using word pairs and then optimizes the word vectors to minimize the difference between the pointwise mutual information of the corresponding words and the dot product of vectors. That gives the correlation and relationship between words that allows for nuanced comparisons, then LSTM excels at handling sequential data, sentences, by effectively capturing long range dependencies between words, allowing it to remember important information from earlier parts of a sequence.

With both methods combined, we reached a relatively successful result. The model performed better than our baseline, and further details are provided in section III-B.


Preprocessed data
Embedding layer: GloVe
LSTM Layer: Units=128, Dropout=0.5
Dense Layer 1: Units=64, Activation=ReLU
Dense Layer 2: Units=2, Activation=Softmax
Output Layer

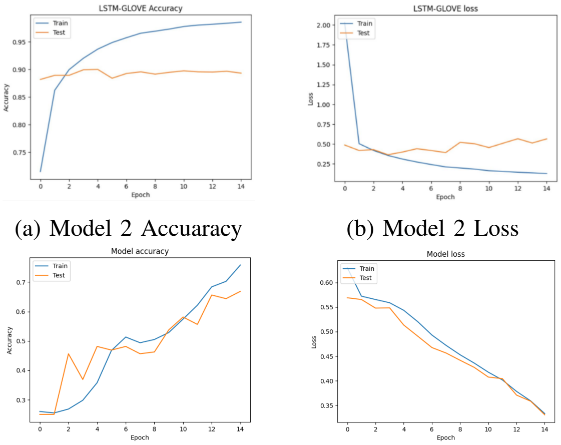
*3) CNN-LSTM+Doc2Vec+TF-IDF:* In this model, we introduced two key differences from the LSTM+GLoVe model. Instead of relying solely on GloVe embeddings, we incorporated Doc2Vec and TF-IDF to enhance feature representation by combining semantic and statistical insights. Doc2Vec is an extension of Word2Vec that generates dense vector representations of entire documents or sentences, preserving semantic relationships while capturing contextual meaning across longer text spans. This allows the model to represent textual data at the document level rather than just the word level, enabling a richer understanding of input sequences. TF-IDF, on the other hand, complements this by emphasizing the importance of words that are unique to a document relative to a corpus, thus highlighting statistically significant terms. By combining both techniques, the model leverages the semantic context of Doc2Vec with the discriminative power of TF-IDF. Additionally, we integrated a CNN layer before the LSTM to extract spatial features, using a 1D convolution with 128 filters and a kernel size of 5, followed by max pooling. This architecture is designed to capture both local patterns and long-term dependencies in the data.

However, this model achieved a lower accuracy compared to the baseline model. Further details are provided in section III-B.


Preprocessed data
Doc2Vec and TF-IDF
CNN: Conv1D, Filters=128, Kernel=5
Pooling layer: maxpooling1D
LSTM Layer: Units=64, Dropout=0.5
Dense Layer 1 :Units=64, ReLU
Dense Layer 2 : Units=2, Softmax
Output Layer

## III. Results

### A. Results



(a) Model 2 Accuaracy

(b) Model 2 Loss

(c) Model 3 Accuaracy

(d) Model 3 Loss

Fig. 2: Results

| | Accuracy | Comparison |
|---|---|---|
| Model 2 | 89.33% | +3.4% |
| Model 3 | 69.21% | -17% |

TABLE I: Accuracy and Comparison with baseline model

### B. Novelty of Results

Model 2 achieves 3% higher accuracy than the baseline model, demonstrating its effectiveness even on a computer-generated dataset.

Despite enhancements, Model 3 performed worse than Model 2, achieving an accuracy of 0.69, a decline of 17% compared to the baseline. This was surprising because it was demonstrated that a CNN-LSTM model applied to a hotel review dataset achieved over 90% accuracy [4]. However, it appears the model does not work well for detecting computer-generated spam reviews, or perhaps the parameters we used were not optimal.

## IV. Discussion/Conclusion

### A. Comparison with Expectations and Result Implications

We were expecting the result to be significantly better than the baseline or at least they can achieve more precision than 90% to help detect fake and computer-generated reviews. Despite the fact that model 3 did not perform as well as expected, our highest accuracy of 89.33% is a good start, and we can proceed with more modifications to boost the score in the future work.

### B. Impact of Experimental Design and Model on Outcomes

At first, we thought the outcome could be due to CNN architecture being less suitable for text processing. Then, we made more research into the problem and our experimental design, and we figured that the problem could be due to Doc2Vec as we did not run enough epochs for training. If we can successfully run multi-layer structure with Doc2Vec, then the result could be promising.

### C. Future works

Future work could explore alternative architectures, such as RNNs or Transformers, which are better suited for capturing complex textual patterns and contextual dependencies. Additionally, exploring more sophisticated feature representations, such as combining contextual embeddings with statistical features like TF-IDF, may further enhance performance.

## V. Code Availability

The Code is available here **Github Repo** or click the URL link: https://github.com/burningblazes/COMP576-Final-Project

## VI. Group Member Roles

- Yusheng Yang: Preprocessed dataset and implemented model 3.
- Shelton Zhou: Implemented model 1 and 2
- Chuyang Wang: Research and Writeup

# REFERENCES

[1] M. Ott, Y. Choi, C. Cardie, and J.T. Hancock. 2011. Finding Deceptive Opinion Spam by Any Stretch of the Imagination. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies.

[2] item M. Ott, C. Cardie, and J.T. Hancock. 2013. Negative Deceptive Opinion Spam. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

[3] Salminen, J., Kandpal, C., Kamel, A. M., Jung, S., and Jansen, B. J. (2022). Creating and detecting fake reviews of online products. Journal of Retailing and Consumer Services, 64, 102771.

[4] Salunkhe, Ashish. ”Attention-based Bidirectional LSTM for Deceptive Opinion Spam Classification.” arXiv preprint arXiv:2112.14789 (2021).

[5] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.