Spam review detection: A survey of conventional machine learning and text extraction techniques

Nguyen Kim Chi, 11211095
Tran Thi Ngan Ha, 11211959
Nguyen Thi Minh Ngoc, 11219280
Bui Phuong Thao, 11215341

National Economic University, Hanoi, Vietnam
Instructor: Nguyen Thi Quynh Giang, PhD

May, 2024

Abstract

The increasing number of spam reviews poses significant challenges to the reliability of products on online shopping platforms. This study aims to enhance the detection of spam reviews by employing various text extraction techniques, including Bag of Words, Word2Vec, TF-IDF, and Hashing Vectorizer, coupled with multiple machine learning algorithms, such as Logistic Regression, K-Nearest Neighbors (KNN), and some Ensemble Learning models. The experiment was conducted in two phases. The first phase focuses on Logistic Regression as the baseline model, exploring different n-grams and text extraction methodologies. Findings reveal that within the baseline model, the combination of unigrams and bigrams emerges as the most stable n-grams, while tf-idf consistently provides the most reliable performance among the text extraction methods. In the second phase, advanced models, including KNN, LightGBM, XGBoost, AdaBoost, and CatBoost are evaluated alongside varying text extraction techniques, while fixing the n-gram variable. The results indicate that both LightGBM and CatBoost achieved good and stable performance, outperforming other models.

1 Introduction

The advent of the Internet has been more than a blessing for mankind thanks to the limitless conveniences it might bring. One of which is the ability to conduct a wide range of activities online, ranging from the basic ones like buying and selling to accessing various services. Since customers cannot make direct inquiries about a product, they must rely on the feedback given by preceding buyers for reference. Those user generated data in the form of social reviews is one of the defining features of Web 2.0 (Shehnepoor et al 2021 [8]). It is noticeable that people's attitudes

and opinions are greatly influenced by others, a phenomenon known as the word-of-mouth effect, which significantly shapes decision-making processes. The advent of the Internet and web-based technologies has opened up numerous opportunities for online word-of-mouth channels, which play a crucial role in shaping consumer purchasing decisions in e-commerce. Positive reviews are likely to indicate good reputations for a product or service, while negative ones are true for the opposite. It is noteworthy that, according to a survey carried out by Weber Shandwick in 2012 [9], online reviews have 83% of impacts on the consumers' opinion and 80% of the consumers have changed their purchase decision in the light of negative reviews.

Given the user-generated nature of online reviews and their growing influence on purchase decisions, the quality and credibility of these reviews become paramount concerns. Moreover, the profit incentives motivate the insertion of "Opinion Spamming" into the review system. Spammers sponsor fraudulent reviews such as fake false positives to promote products or maliciously negative to devalue services of a business. A huge number of spam reviews inserted manually or automatically have been noticed on different well-known online platforms, which leads to gradual loss in credibility as well as a rise in confusion among consumers. Poor and misleading assessment may not only cause frustration for buyers but might also eventually result in the corruption of the review system.

Consequently, investigating fake reviews has been proposed as a key focus area in digital and social media marketing research (Dwivedi et al., 2020 [10]). The first step to secure the review system involves how well the review spam is detected. Within the scope of this paper, the focus is placed on differentiating computer-generated reviews with actual ones by combining existing text extraction techniques with varying classifying models, defining key research issues, and proposing future research challenges for spam review detection.

2 Literature Review

2.1 Spam Review and Opinion Spamming

As previously stated, reviews are UGC (user generated contents) that are provided by users after experiencing a product or service, to express their personal opinions about them. A typical review includes a narrative comment and a rating of the object. Thus, a fake review is a review written or generated without any actual experience of the mentioned products. Traditionally, human-generated fake reviews have been bought and sold like commodities in a "market of fakes" (He et al, 2021 [7]). In this market, one can order a specific quantity of reviews online, which human writers then produce. However, the advances in natural language generation have facilitated the large-scale production of fake reviews, making it increasingly difficult for human readers to distinguish between a human-authored text and the one that is not (Floridi and Chiriatti, 2020 [11]).

Opinion Spamming is a similar concept to fake reviews, especially when false opinions are written to influence other online users, Ott et al(2011) [12] describes them as fictitious opinions that have

been deliberately written to sound authentic. The similarity between fake reviews and opinion spamming is that they are written partially or completely based on financial incentives rather than genuine endorsement towards the product and service.

It is important to note that fake reviews can aim to either enhance or sabotage an organization's reputation, and they can originate from various sources, including the organization itself or third parties like competitors or disgruntled customers. Recognizing the diversity in motives is crucial for understanding the full scope of the phenomenon. Fake reviews can include both praises for a particular product and attacks against rival brands. Positive reviews of a target brand, company, or product are intended to attract more customers and boost sales, while negative reviews aim to harm the target's reputation and reduce sales (Shivagangadhar et al, 2015 [13]). The display order of reviews often introduces a presentation bias. Studies carried by Karahalios (2010 [14]) have shown that reviewers without strong opinions are often influenced by previous reviews, merely "echoing" them without providing new input. Additionally, reviewers may develop an expectation for a product or service based on prior reviews, leading to biased judgments guided by these expectations. Therefore, biased or low-quality reviews should be distinguished from fraudulent reviews, which are typically added to the review system with a clear intention or goal.

2.2 Existing Spam Review Detection Approaches

Many researchers have shown their efforts in spam detection mechanisms. A basic approach, which is based on the premise that humans can detect when others behave in a fraudulent way (DePaulo et al, 1996 [15]), is to analyze reviews manually. A set of rules for differentiating incentivized and non-incentivized reviews based on the length, sentiment and the level of helpfulness has been developed by Costa et al (2019 [16]). One obvious challenge with the use of those heuristic rules is that once those rules become common knowledge, spammers are totally capable of adapting to them and changing their behavior, which invalidates the rules. Ott et al (2011) recruited humans to judge whether a review was fake or not, finding that the highest accuracy for a human was 65%, significantly lower than the 86% accuracy achieved by a machine learning model. In another study by Sun et al. (2013 [17]), human accuracy was only 52%, indicating that it is very challenging for humans to distinguish fake reviews from real ones. This suggests that the heuristics people use are not effective against the various deceptive tactics employed in fake reviews.

Hence, it is more reasonable for researchers to resort to automatic methods as a potential solution to tackle the problem of detecting fake reviews. Two of the most prevalent approaches are machine learning based and deep learning based methods. Machine learning approach is about extracting features from the data using given rules of algorithms to solve the problem of regression or classification. Meanwhile, deep learning mimics the human brain to find the solutions to the given task without the need of human intervention by employing a neural network with 'multi-layers'. Review spam detection, similar to opinion mining, falls under content mining but also utilizes features not directly related to the content. Constructing features to describe the text of the review involves

text mining and Natural Language Processing (NLP). Additionally, there may be features linked to the review's author, its post date or time, and how the review deviates from other reviews for the same product or service.

It is important to note that while most existing machine learning techniques are not sufficiently effective for review spam detection, they are more reliable than manual detection. The primary issue, as identified by Abbasi et al [18], is the lack of distinguishing words (features) that can definitively classify reviews as real or fake. N. Jindal and B. Liu (2008 [19]) proposed a method for detecting spam based on review duplication. The data was sourced from Amazon.com and included 5.18 million reviews and 2.14 million reviewers. They used a shingle method for detecting near-duplicate reviews, calculating a similarity score and labeling those with a score over 90% as duplicates. Naive Bayes and SVM yielded poor results, but logistic regression achieved an AUC (Area Under ROC Curve) score of 0.63 using only text features and 0.78 using all features, indicating that a good classification model requires more than just the content of the reviews. Researchers observed that many spammers copy existing reviews entirely or change only a few words. Therefore, many researchers in this area have focused on methods for duplicate detection.

Detecting spam using similarity between reviews can indeed be a useful technique, but it's important to note that spammers often copy genuine reviews, making it challenging to rely solely on this approach. Besides duplicate detection, various other techniques utilize review content for spam detection. A comprehensive breakdown of these categories is provided by M. Crawford et al (2015) [20]: Bag of words approach: this method considers words or sequences of words used in reviews as features. Sequences of words, known as n-grams (where n denotes the number of words in a sequence), with values of n=1, 2, or 3 being most common. Term frequency: this approach includes n-grams as well as the number of their occurrences. Incorporating this additional information can enhance the bag of words approach. POS tags (Part-of-Speech tags): POS tags are labels assigned to words based on their context, such as adverbs, verbs, etc. This information, along with additional features, is fed into a machine learning algorithm. Janez Martin(2020 [21]) made the combined model of TF-IDF and SVM showed 95.39% F1- score and the fastest spam classification achieved with the help of the TF-IDF and NB approach after meticulous steps of processing English only text input.Alberto (2021 [22]) explained deception detection using various machine learning algorithms with the help of neural networks, random forests. The experiment discussed in this paper examines the use of Bag of Words and Part of Speech tag features to detect deception on the varied types of communication using Neural Networks, Support Vector Machine, Naive Bayesian, Random Forest, Logistic Regression, and Decision Tree, and paved a path for new research direction.

Interestingly, a hypothesis of whether a machine classifier is better or worse than people at detecting fake reviews and if the generator model can actually generate reviews that are good enough to fool human reviewers is tested by Salminen et al, 2021 [23].They create a dataset by generating sample reviews using two language models then train classifier algorithms to detect artificially generated reviews from the real ones. After that crowd workers are recruited to annotate a sample of the

original and fake reviews before the researchers compare the accuracy of crowd workers and the classification algorithms via statistical testing. Two types of baseline models that have been used are SVM algorithm, the OpenAI fake detection model before fine-tuning. Their results show much lower agreement among humans than among the ML models, which implies, one the one hand, that people differ by their ability to detect fake reviews and, on the other hand, that machine classifiers perform, regardless of the classifier, more consistently than humans for this detection task.

2.3 Summary and Research Gap

Fake reviews present a widespread and harmful issue, making it crucial yet challenging for consumers and businesses to distinguish between genuine and fake reviews (Crawford et al, 2015 [20]). Detecting fake reviews often involves a combination of manual efforts, supervised machine learning, and heuristic methods (Fontanarava et al, 2017 [24]). Despite the progress made in detection studies, significant challenges remain. Classification performance needs improvement to keep pace with evolving text-generation algorithms. Additionally, datasets may not be appropriately constructed, contain mislabeled instances, or lack public availability. The key takeaway from previous studies is that automatic fake review detection has only achieved partial success. While no single study can address all gaps, our study dive deep into the comparison of how well different text extraction techniques work in combination with multiple machine learning and boosting algorithms. By making our experiments available for future development, we aim to contribute to ongoing efforts in combating fake reviews.

3 Dataset

The dataset utilized in this study pertains to the investigation of fake online product reviews, as conducted by Joni Salminen, Chandrashekhar Kandpal, Ahmed Mohamed Kamel, Soon-gyo Jung, and Bernard J. Jansen in 2022 [1]. This dataset comprises a total of 40,000 reviews, evenly divided between 20,000 artificially generated by GPT-2 based on k-core subsets of the publicly available Amazon Review Data (2018) dataset, and 20,000 authentic reviews written by humans sourced from the same Amazon dataset.

To generate the synthetic reviews, the researchers focused on the top ten Amazon categories with the highest volume of product reviews. They randomly selected an equal proportion of reviews from each category. For each product category, 2,000 reviews were generated using a fine-tuned GPT-2 language model. Specifically, GPT-2 utilized the first five words from each sampled review to generate the remainder of the review autonomously. The distribution of review lengths informed the creation of discrete buckets with one-word intervals ranging from 10 to 350 words, which were adopted as the target sentence lengths for the generated reviews.

Due to the active generation process, the dataset exhibits a high level of quality, with well-balanced classes across the entire dataset, within each category, and even for each rating (figure 1). The text

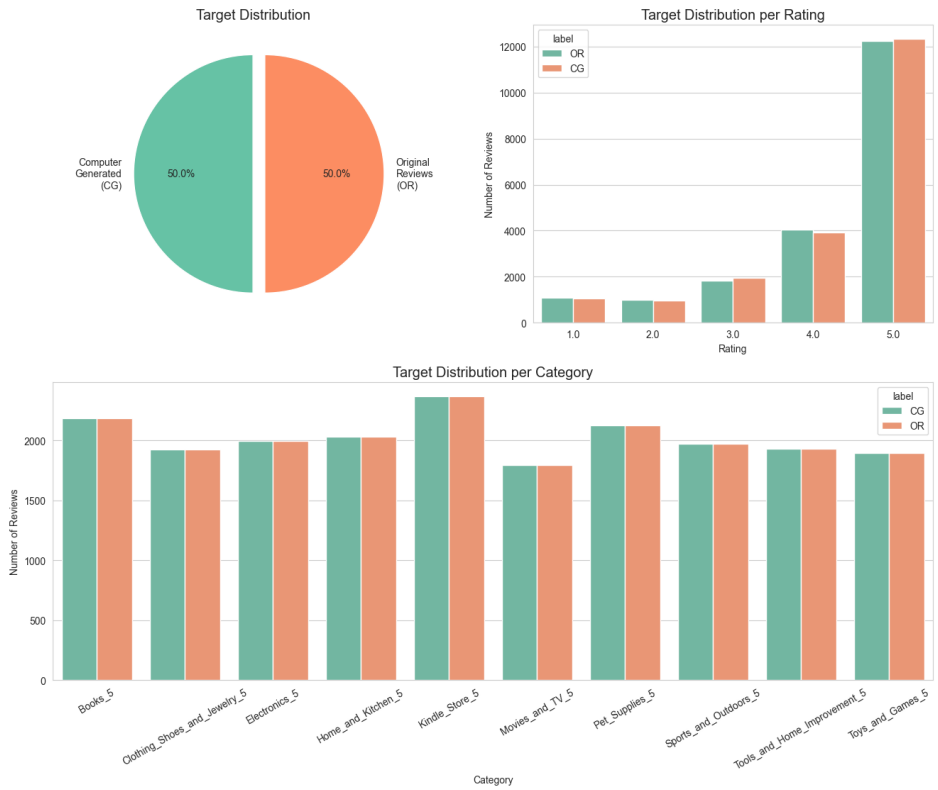

Figure 1: Target Distribution

length patterns (figure 2) of the two classes are not markedly different, suggesting that text length may not be a significant feature in the classification phase.

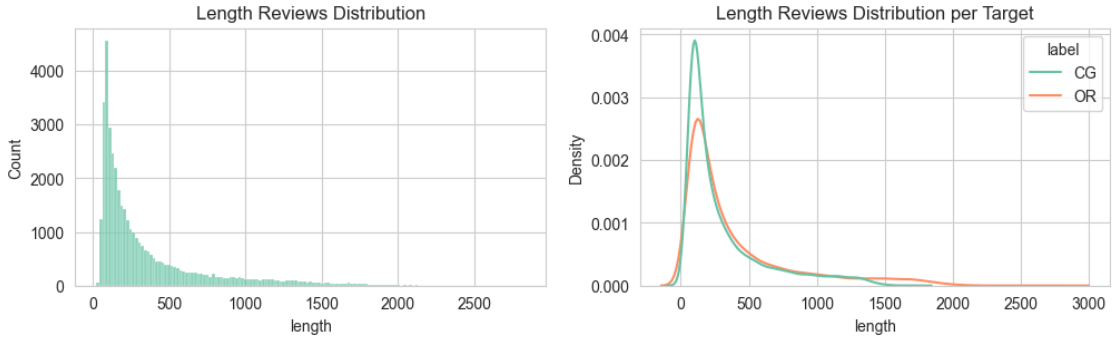

Figure 2: Length Text Distribution

4 Methodology

4.1 Text Extraction

4.1.1 Bag of Words (BoW)

The growth of using Bag of Words (BoW) method in text classification is visibly observed.In text classification, the Bags of Words (BoW) method records the number of occurrences of each bag that is created for each instance type or word disregarding the order of the words or the grammar but considering their frequency. These lexical units, known as n-grams, are constructed by selecting n consecutive words from the text and serve as features for analysis.

Taking some examples in our dataset to explain the simplest case - the unigram BoW:

- Review 1: "I love this stand mixer. The quality is just what I wanted and the mixer is just what"
- Review 2: "Love it! Would like more colors and quality.Very good quality.Nice, sturdy, and functional"

After some basic cleaning steps, each sentence is segmented into word buckets through tokenization and a "dictionary" will be constructed by combining the unique buckets found in both text documents.

{"love", "stand", "mixer", "quality", "want", "would", "like", "color", "good", "nice", "sturdy", "function"}

Subsequently, each review is represented using the BoW method. This involves constructing a vector for each review where the frequency of each word is recorded based on its position in the dictionary.

- Review 1: [1,1,2,1,1,0,0,0,0,0,0,0]
- Review 2: [1,0,0,2,0,1,1,1,1,1,1,1]

Binary count can also be used when the presence of a word in a document is considered more significant than its frequency. In this approach, all non-zero counts will be set to 1.

- Review 1: [1,1,1,1,1,0,0,0,0,0,0,0]
- Review 2: [1,0,0,1,0,1,1,1,1,1,1,1]

4.1.2 Term Frequency - Inverse Document Frequency (TF - IDF)

TF-IDF is one of the widely used feature extraction methods when working with natural language processing. In particular, TF-IDF is a statistical measure that extracts keywords from documents and determines the importance of a term within a document relative to a collection of documents.

The intuition behind this method is that when text vectorization is executed solely based on calculating the frequency of occurrence of a specific word, words that are more likely to occur would have higher rating. Regarding English, there are plenty of stop words - defined as commonly used words in a language - whose appearance only serves the grammatical purpose but has little, to none useful implication compared to less frequent ones. Thus, a more reasonable approach is to compare the uniqueness of words in a particular document with their frequency in others. If a word is commonly found in one document but rarely so in the rest, then this word is of high cruciality for that one specific document, thus receiving more weight.

TF-IDF can be calculated by multiplying two metrics, which are the two-dimensional matrix TF and the one-dimensional matrix IDF. TF stands for term frequency - measuring the occurrence of specific words in documents. DF stands for Document frequency, implying how many times a word appears in the corpus, therefore, words with high DF value do not show a significant role. Accordingly, IDF is the inverse of DF, high IDF indicates rare words, rising importance.

Suppose we have a given corpus with N documents $\{d_1, d_2, d_3, \ldots\}$. Let $d_j = \{t_1, t_2, \ldots, t_q\}$ be a document containing q terms. t_i is an arbitrary term that appears in d_j. $n(i,j)$ denotes the number of occurrences of t_i in document d_j, then the term frequency of t_i regarding d_j is computed as:

$$\mathrm{TF}(i,j) = \frac{n(i,j)}{\sum_{k=1}^{q} n(k,j)}$$

Let m_i denote the total number of documents where term t_i appears at least once, then the inverse document frequency of t_i concerning the whole corpus is

$$\mathrm{IDF}(i,D) = \log\left(\frac{N}{m_i}\right)$$

The final TF-IDF value results from the equation:

$$\text{TF-IDF}(i,j) = \mathrm{TF}(i,j) \times \mathrm{IDF}(i,D)$$

4.1.3 Hashing Vectorize

The Hashing Vectorizer uses the hashing trick to convert features into indices in a feature vector. This means that it applies a hash function to the features and uses the resulting hash value as the index for the feature in the feature vector.

A common approach is to construct, at learning time or prior to that, a dictionary representation of the vocabulary of the training set, and use that to map words to indices. Hash tables and tries are common candidates for dictionary implementation. E.g: Three documents:

- John likes to watch movies.
- Mary likes movies too.
- John also likes football.

Term Index:

Term	John	Likes	to	watch	movies	Mary	too	also	football
Index	1	2	3	4	5	6	7	8	9

Matrix is converted:

	John	likes	to	watch	movies	mary	too	also	football
Doc 1	1	1	1	1	1	0	0	0	0
Doc 2	0	1	0	0	1	1	1	0	0
Doc 3	1	1	0	0	0	0	0	1	1

Hash Function: A hash function $h : U \rightarrow 0, ..., m-1$ maps the universe U of keys to indices or slots within the table, that is, $h(x) \in 0, ..., m-1 \; x \in U$. The conventional implementations of hash functions are based on the integer universe assumption that all elements of the table stem from the universe $U = 0, ..., u-1$, where the bit length of u is confined within the word size of a computer architecture.

Hash Table: An associative array stores a set of (key, value) pairs and allows insertion, deletion, and lookup (search), with the constraint of unique keys. In the hash table implementation of associative arrays, an array A of length m is partially filled with n elements, where $m \geq n$. A value x gets stored at an index location $A[h(x)]$, where h is a hash function, and $h(x) < m$. Under reasonable assumptions, hash tables have better time complexity bounds on search, delete, and insert operations in comparison to self-balancing binary search trees. Hash tables are also commonly used to implement sets, by omitting the stored value for each key and merely tracking whether the key is present.

4.1.4 Average Word2Vec

Word embeddings are one way to represent words and whole sentences numerically by converting them into real-number vectors thereby enabling computational systems to interpret and manipulate textual data. Word2Vec is one of the most popular techniques for learning word embeddings using shallow neural networks. One of the main principles of Word2Vec is trying to make vectors of similar meaning words closer in space.

To create a vector for each word, we are going to train a simple neural network with a single hidden layer to perform a certain task with certain inputs and their corresponding output. However,

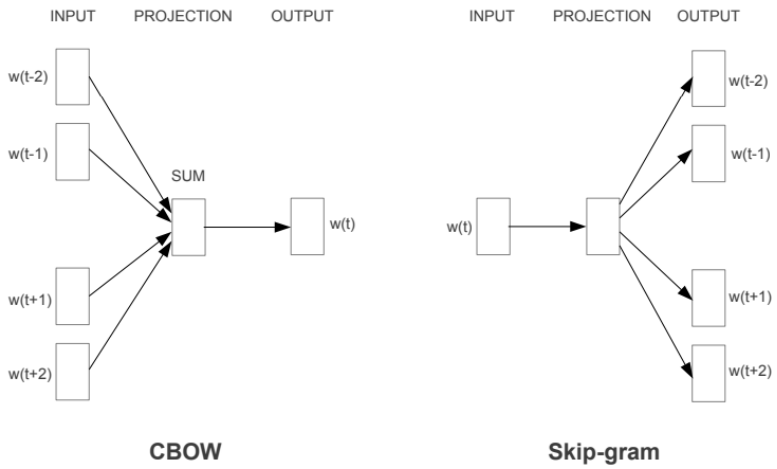

Figure 3: CBOW vs Skip-gram Architectures [2]

instead of using the output of that neural network, our actual goal is to learn the weights of the hidden layer, which are the word vectors that we are trying to find.

The inputs and their corresponding output are determined by the type of Word2Vec model architectures. There are two types of Word2Vec model architectures: Continuous Bag of Words (CBOW) and Skip-gram.

Continuous Bag-of-Words Model

The CBOW model architecture tries to predict the center word based on the context words, or surrounding words (words to the left and words to the right within the predefined context window size of the center word) [2]. For example, given the sentence "the quick brown fox jumps over the lazy dog" and a context window of size one, we will use one word to the left and one word to the right to predict the center word, such as using "brown" and "jumps" to predict the word "fox".

Skip-gram Model

The Skip-gram model architecture usually tries to achieve the reverse of what the CBOW model does. It tries to predict the source context words (surrounding words) given a target word (the center word) [2]. Consider the same sentence from earlier "the quick brown fox jumps over the lazy dog" and a context window of size one, instead of using "brown" and "jumps" to predict the word "fox", we will use the word "fox" to predict the word "brown" and "jump".

After obtaining numerical vectors for each word using these two architectures, we compute the average values of all the word vectors in a sentence to determine the vector for that sentence.

This approach facilitates the numerical representation of sentences, enabling further computational manipulation and interpretation.

4.2 Models

4.2.1 Logistic Regression

Logistic regression is a regression model aimed at predicting the value of a discrete target variable x corresponding to an input vector y. This is equivalent to classifying input vectors x into corresponding groups y.

The output of Logistic Regression is passed through the sigmoid function to bring the output into the range $(0, 1)$ - indicating the probability of the output for each class. This function is called an activation function and its formula is as follows:

$$f(x) = \frac{1}{1 + e^{-x}}$$

If the output of the network after passing through the sigmoid function is greater than a certain threshold, then we assign the value to the class as 1; otherwise, it will be 0.

4.2.2 K-nearest Neighbors (KNN)

K-nearest neighbor is one of the simplest supervised learning algorithms in Machine Learning. During training, this algorithm does not learn anything from the training data (this is also why it is classified as lazy learning); all computations are performed when it needs to predict the outcome for new data. K-nearest neighbor can be applied to both types of supervised learning problems: Classification and Regression. KNN is also known as an Instance-based or Memory-based learning algorithm.

With KNN, in a Classification problem, the label of a new data point (or the answer to a question in a test) is directly inferred from the K nearest data points in the training set. The label of a test data point can be determined by majority voting among the nearest points, or it can be inferred by assigning different weights to each of the nearest points and then deriving the label.

4.2.3 LightGBM

Light Gradient Boosting Machine, is a highly efficient and fast gradient boosting framework developed by Microsoft. It excels in both classification and regression tasks. Boosting combines multiple weak learners to form a strong learner, enhancing prediction accuracy. Traditional Gradient Boosting Machines (GBMs) often face heavy computational demands.

LightGBM addresses this with two innovations: Gradient-based One-Side Sampling (GOSS), which prioritizes data points with large gradients for training, and Exclusive Feature Bundling (EFB),

which bundles exclusive features to optimize memory usage [3]. These techniques make LightGBM faster and more scalable than traditional GBMs while maintaining high accuracy.

4.2.4 XGBoost

XGBoost, also known as Extreme Gradient Boosting, enhances the traditional gradient boosting algorithm by incorporating a range of optimizations that significantly improve performance and efficiency. One of the key advantages of XGBoost is its ability to leverage parallel processing, which dramatically accelerates the training process compared to conventional methods. XGBoost operates by training an ensemble of models on strategically selected subsets of the training data. Instead of aiming to create a single, perfect model, XGBoost builds numerous weak learners, typically decision trees, each focusing on correcting the errors of its predecessors. These models are then combined through a process of weighted voting to generate the final prediction. This ensemble approach results in a more robust and accurate model, as it effectively reduces overfitting and captures complex patterns in the data. [4] Additionally, XGBoost incorporates several regularization techniques that help prevent overfitting, making it particularly well-suited for handling large datasets with high-dimensional features. Its ability to handle missing data, support for various objective functions, and hyperparameter tuning options further enhance its versatility and effectiveness in a wide range of machine learning tasks.

4.2.5 AdaBoost

AdaBoost, short for Adaptive Boosting, is a powerful ensemble learning technique designed to improve the performance of weak learners. The algorithm works by iteratively training a series of weak learners, typically decision stumps, on the dataset. During each iteration, AdaBoost emphasizes the importance of data points that were misclassified in previous rounds by increasing their weights. This forces subsequent learners to focus more on these challenging cases, effectively "learning from mistakes." [5]

The process continues until a predetermined number of weak learners have been trained, resulting in a collection of diverse and complementary models. The final AdaBoost model aggregates the predictions from all these weak learners, assigning higher weights to those that performed better during training. This weighted voting mechanism ensures that the final model is robust and accurate, as it leverages the strengths of multiple learners while mitigating their individual weaknesses. By adaptively focusing on the hardest-to-classify examples, AdaBoost effectively enhances the overall predictive performance.

4.2.6 CatBoost

Categorical Boosting, is an open-source boosting library developed by Yandex, tailored for regression and classification tasks with numerous independent features. Unlike traditional models,

CatBoost processes categorical features directly, bypassing the need for One-Hot or Label Encoding, which simplifies data preparation. Its use of the Symmetric Weighted Quantile Sketch (SWQS) algorithm effectively handles missing values within the boosting process, reducing overfitting and improving performance. [6]

CatBoost automatically scales features, ensuring uniform processing without manual intervention. It employs techniques including robust tree boosting, ordered boosting, and random permutations to prevent overfitting, enhancing generalization to unseen data. Designed to perform well with minimal parameter tuning, CatBoost saves users time and effort while achieving competitive results. Additionally, built-in cross-validation determines optimal hyperparameters, further boosting model reliability. For large datasets, CatBoost's GPU-accelerated version offers fast and scalable training, making it a powerful and efficient tool for handling both categorical and numerical data.

4.3 Metrics Evaluation

In this study, we employed k-fold cross-validation to obtain a robust estimate of our machine learning model's performance across various parameter settings. We evaluated the model using multiple metrics, including accuracy, F1-score, ROC AUC, and reported the average performance across all folds.

To calculate the evaluation metrics mentioned earlier, we first establish four key variables specific to binary classification problems. These are:

- True Positive (TP): The number of true positives correctly classified by the model. These are genuine instances of the positive class that the model identified correctly
- False Negative (FN): The number of false negatives incorrectly classified by the model. These are actual positive class instances that the model predicted as negative. (Also known as Type II error)
- The number of false positives incorrectly classified by the model. These are negative class instances that the model mistakenly predicted as positive. (Also known as Type I error)
- The number of true negatives correctly classified by the model. These are genuine negative class instances that the model identified correctly.

4.3.1 Accuracy Score

Accuracy is the proportion of correct predictions out of the total number of predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

4.3.2 F1 Score

The F1 score is described as the harmonic mean of the precision and recall of a classification model. The two metrics contribute equally to the score, ensuring that the F1 metric correctly indicates the reliability of a model.

- Precision: The precision metric determines the quality of positive predictions by measuring their correctness. It is the proportion of true positive predictions out of all positive predictions (both true and false).

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall, also called sensitivity, measures the model's ability to detect positive events correctly. It is the percentage of accurately predicted positive events out of all actual positive events.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- F1 score:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score ranges between 0 and 1, with 0 denoting the lowest possible result and 1 denoting a flawless result, meaning that the model accurately predicted each label.

A high F1 score generally indicates a well-balanced performance, demonstrating that the model can concurrently attain high precision and high recall. A low F1 score often signifies a trade-off between recall and precision, implying that the model has trouble striking that balance.

4.3.3 ROC AUC Score

AUC-ROC curve: A performance measure for binary classifiers, visualizing how well it separates positive from negative classes. AUC is the area under the curve, summarizing overall model performance. Receiver Operating Characteristics (ROC) Curve is the graphical representation of the effectiveness of the binary classification model. It plots the true positive rate (TPR) vs the false positive rate (FPR) at different classification thresholds.

- True Positive Rate:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- False Positive Rate

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

5 Experimental Results

5.1 Baseline Models

The primary concern in this phase revolves around the role of text length as a feature in classification. Specifically, we aim to determine whether including text length during training is essential or whether its influence is negligible, rendering its inclusion optional.

To answer this question, other components will be fixed so that the impact of length text feature will be obtained. The findings of our analysis (figure 4) reveal that for all text extraction techniques and ngrams, there is no notable disparity in performance when considering text length as a training feature compared to excluding it. This suggests that the presence of the text length feature is not imperative and can be considered optional.

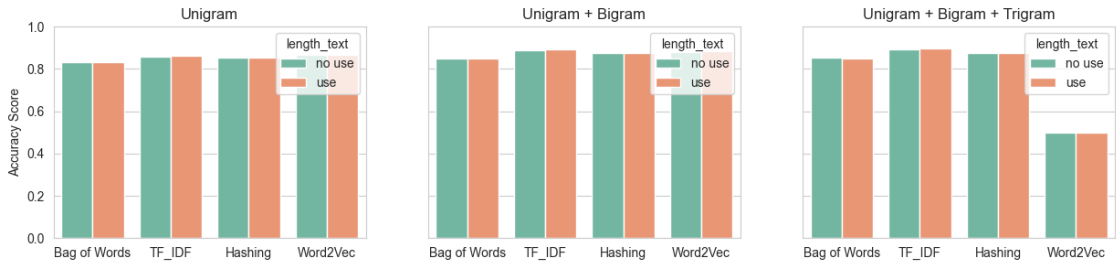

Figure 4: Impact of Length (Different Ngrams)

Further, while examining the performance across different n-gram combinations (figure 5), it was observed that the variations were not particularly substantial, except in the case of the combination of unigram, bigram, and trigram using the Word2Vec technique. However, the general variation was enough to indicate that the combination of unigram and bigram exhibits the most consistent performance.

Note that, the parameters for each text extraction technique were chosen after extensive testing, with other components fixed. For instance, in the case of Word2Vec, we tested the performance with varying vector sizes, window sizes, and the number of epochs for both CBOW and skip-gram architectures. The results indicated that for CBOW, a higher number of epochs improved performance, while for the skip-gram model, larger vector sizes yielded better results. Despite the longer training time required for skip-gram models, their performance was generally superior to CBOW. Consequently, within the scope of this study, we opted for the skip-gram model with a vector size of 600, a window size of 10, and 15 epochs. Similar testing was conducted for other techniques, leading to specific parameter selections such as binary bag of words outperforming the normal bag of words in this context, and higher feature numbers improving results for both the hashing vectorizer. Additionally, for both bags of words and tf-idf, the minimum frequency

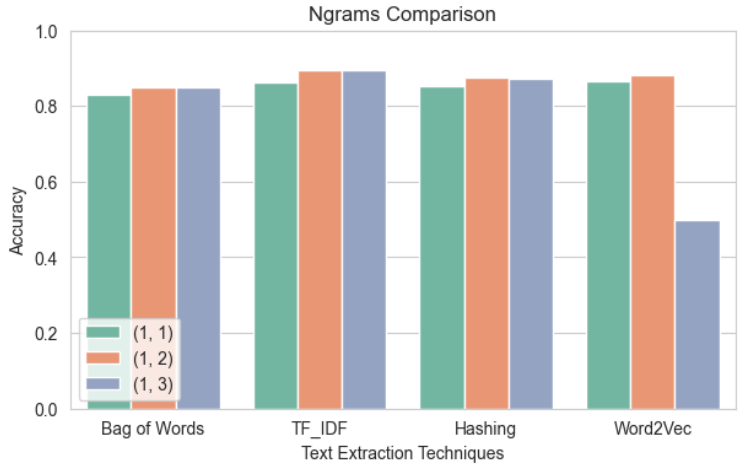

Figure 5: Ngrams Comparison

threshold for each word should be carefully defined to be low but not too low.

In terms of the performance of the baseline model, Word2Vec emerged as the best choice for unigrams (with highest accuracy scores), but the least effective for the combination of unigrams, bigrams, and trigrams. Meanwhile, the TF-IDF method demonstrated the most stability across all n-gram combinations, consistently achieving the highest accuracy scores. This suggests that TF-IDF may be the most reliable text extraction technique for ensuring robust model performance across different n-gram configurations (figure 6).

5.2 Advanced Models

In the second phase, we mainly focus on implementing further Machine Learning models with the aim of improving the model's accuracy. Using the conclusions from first phase, we fixed the n-gram variable as the combination of unigram and bigrams.

Evaluating the results between the models (figure 7), we find that the advanced models do not outperform the baseline model in terms of accuracy. Except CatBoost achieves higher accuracy than Logistic Regression - our baseline model - when using Word2Vec as a feature extraction technique, Logistic Regression consistently yields the highest result. Looking at the distribution among the text extraction techniques, we can see that the models have relatively consistent results with each other. It is only with KNN that we observe significant variation. With the hashing method, KNN yields a significantly lower value compared to the other five models. However, with Word2Vec, the accuracy of KNN is higher than that of AdaBoost, despite KNN typically being the model with the lowest accuracy. After analysis, we can conclude that among the advanced models,

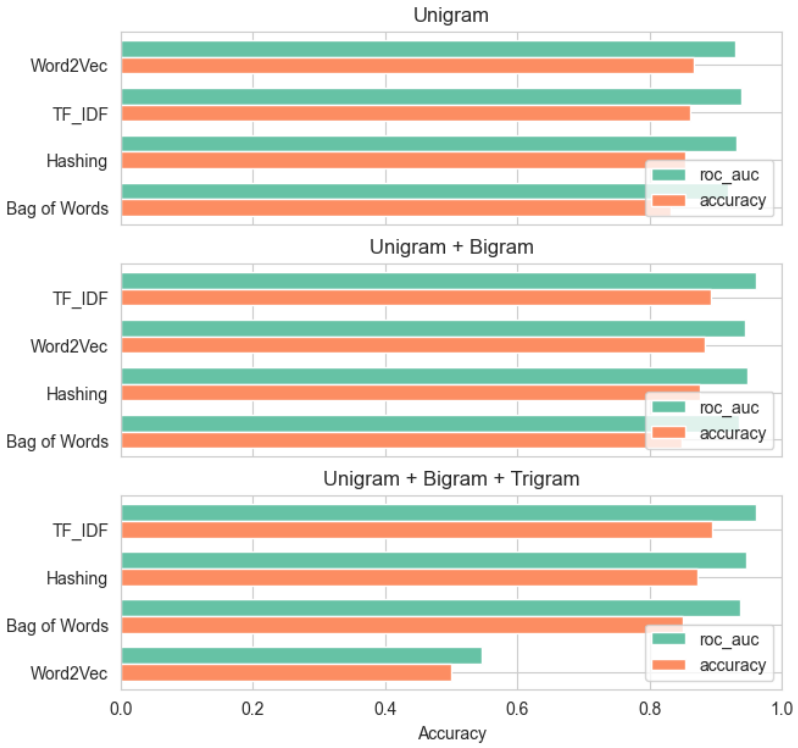

Figure 6: Performance of Different Text Extraction (Baseline)

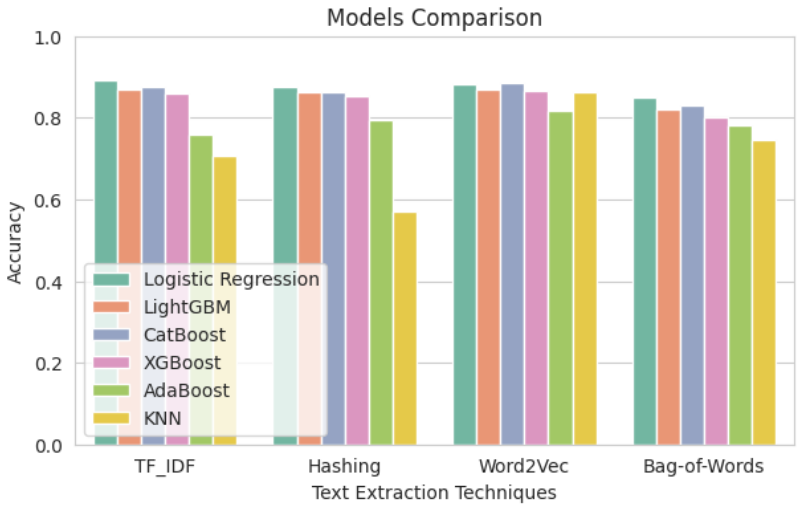

Figure 7: Performance of Baseline and Advanced Models

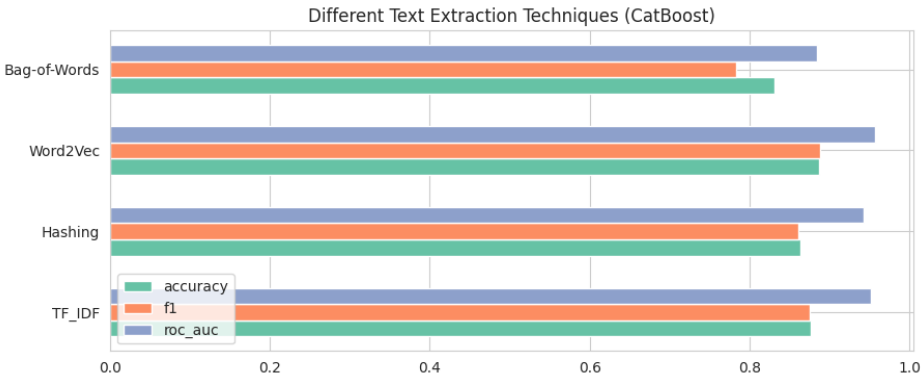

Figure 8: Performance of Different Text Extraction Techniques (CatBoost)

CatBoost consistently yields the best results across all four text extraction methods.

Comparing specifically the combination of CatBoost with text extraction techniques (figure 8), we observe no significant difference in results between the methods. This could imply that CatBoost is robust to variations in text preprocessing or feature extraction methods.

6 Conclusion

Our research explored spam review detection through two phases. In Phase 1, we established a baseline using Logistic Regression, examining the impact of text length, different n-gram representations, and various text extraction techniques. Results showed that text length had minimal impact, and the combination of unigrams and bigrams provided the most consistent performance, with TF-IDF being the most stable and accurate extraction method.

In Phase 2, we tested advanced models including K-Nearest Neighbors, LightGBM, XGBoost, AdaBoost, and CatBoost with the optimal n-gram combination from Phase 1. While most advanced models did not outperform the baseline, CatBoost consistently delivered the best results across all text extraction methods, demonstrating its robustness and reliability.

Overall, the study highlights the effectiveness of TF-IDF for text extraction and CatBoost for model performance, contributing valuable insights for improving spam review detection on online platforms. Future work should focus on further refining feature engineering and optimizing models to enhance detection accuracy.

7 Limitations and Future Work

Our study has primarily focused on the detection of computer-generated reviews, neglecting the significant portion of spam reviews created by humans. Human-generated spam reviews often exhibit different characteristics and patterns that our current models may not effectively capture. Future research should expand to investigate these human-generated spam reviews to enhance the comprehensiveness and robustness of spam detection systems. Additionally, our current work is limited to English-language reviews, whereas reviews can be written in a multitude of languages, including non-Latin scripts. Addressing multilingual and non-Latin language reviews is crucial for developing a universally applicable spam detection system.

Besides linguistic pattern analysis, which our current research emphasizes, future work should also consider analyzing user behavior patterns, such as review posting frequency and interaction history, as these may reveal additional indicators of spam. Another limitation of our study is the reliance on traditional machine learning models, without exploring advanced neural networks and deep learning models. These advanced models have demonstrated significant potential in various natural language processing tasks and should be investigated for their applicability and performance in spam review detection. Finally, to make this research more practical, future work should not only focus on identifying the most effective model but also aim at developing a comprehensive system capable of detecting spam reviews in real-time. This would involve integrating efficient processing algorithms and scalable architectures to handle large volumes of review data swiftly and accurately.

By addressing these limitations and exploring these future directions, we hope to develop more effective and reliable spam review detection systems, thereby preserving the authenticity and trustworthiness of online reviews.

References

[1] Joni Salmien, Chandrashekhar Kandpal, Ahmed Mohamed Kamel, Soon-gyo Jung, and Bernard J. Jansen (2022) Creating and detecting fake reviews of online products. Available here.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean (2013) Efficient Estimation of Word Representations in Vector Space. Available here.

[3] Guolin Ke1, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma1, Qiwei Ye1, Tie-Yan Liu (2017) LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Available here.

[4] Tianqi Chen, Carlos Guestrin (2016) XGBoost: A Scalable Tree Boosting System. Available here.

[5] Yoav Freund, Robert E. Schapire (1996) Experiments with a New Boosting Algorithm. Available here.

[6] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, Andrey Gulin (2017) CatBoost: unbiased boosting with categorical features. Available here.

[7] He, Sherry and Hollenbeck, Brett and Proserpio, Davide, The Market for Fake Reviews (October 1, 2022). Available here.

[8] Saeedreza Shehnepoor, Roberto Togneri, Wei Liu, Mohammed Bennamoun, Social Fraud Detection Review: Methods, Challenges and Analysis (November 10, 2021). Available here.

[9] Weber Shandwick, The company behind the brand: in reputation we trust citation (November 10, 2021). Available here.

[10] Yogesh K. Dwivedi, Elvira Ismagilova, D. Laurie Hughes, Jamie Carlson, et al (2021) Setting the future of digital and social media marketing research: Perspectives and research propositions. Available here.

[11] Floridi, L., Chiriatti, M. GPT-3: Its Nature, Scope, Limits, and Consequences (2020). Available here.

[12] Myle Ott, Yejin Choi, Claire Cardie, T. Jeffrey, Hancock, Finding deceptive opinion spam by any stretch of the imagination (2011)Available here.

[13] Kolli Shivagangadhar, H. Sagar, Sohan Sathyan, C.H. Vanipriya, Fraud detection in online reviews using machine learning techniques (2015). Available here.

[14] E. Gilbert and K. Karahalios,Understanding deja reviewers (2015). Available here.

[15] Bella M. DePaulo, Deborah A. Kashy, Susan E. Kirkendol, Melissa M. Wyer, Jennifer A. Epstein,Lying in everyday life,(1996). Available here.

[16] Ana Costa, Joao Guerreiro, Sergio Moro, Roberto Henriques,Unfolding the characteristics of incentivized online reviews (2019). Available here.

[17] Huan Sun, Alex Morales, Xifeng Yan Synthetic review spamming and defense (2013). Available here.

[18] Ahmed Abbasi, Zhu Zhang, David Zimbra, Hsinchun Chen, Jay F. Nunamaker Jr, Detecting Fake Websites: The Contribution of Statistical Learning Theory (2010). Available here.

[19] Nitin Jindal, Bing Liu, Opinion spam and analysis (2008). Available here.

[20] Crawford, M., Khoshgoftaar, T.M., Prusa, J.D. et al. Survey of review spam detection using machine learning techniques (2015). Available here.

[21] Francisco Janez-Martino, Eduardo Fidalgo, Santiago GonzÃ¡lez-MartÃnez, Javier Velasco-Mat, Classification of Spam Emails through Hierarchical Clustering and Supervised Learning (2020). Available here.

[22] Ceballos Delgado, Alberto Alejandro,Glisson, William Bradley,Shashidhar et al. (2021) Detecting Deception Using Machine Learning. Available here.

[23] Joni Salminen, Chandrashekhar Kandpal, Ahmed Mohamed Kamel, Soon-gyo Jung, Bernard J. Jansen, Creating and detecting fake reviews of online products (2022). Available here.

[24] J. Fontanarava, G. Pasi and M. Viviani, Feature Analysis for Fake Review Detection through Supervised Classification (2017). Available here.

Appendix

Link to Repository