# Enhancing Naive Bayes using techniques like auxiliary features and TF-IDF weighting

Raghav Kamineni, Lavanya Krishna

TJHSST

12/15/24

Dr.Yilmaz

Quarter 2 Project

***Abstract -*** *Spam messages have become an increasingly pervasive issue in the digital era, posing risks such as phishing and malware attacks. Effective spam detection is crucial to maintaining secure communication. While Naive Bayes classifiers are widely used in text classification for their simplicity and computational efficiency, they suffer from certain limitations, such as the independence assumption–attributes are independent of each other–which affects their accuracy. Building on the auxiliary feature method proposed in [3], which enhances classification accuracy by introducing auxiliary features to reclassify the text space, and leveraging insights from the comparison study [6], we aim to explore and apply new enhancements to Naive Bayes to improve spam detection accuracy.*

**Introduction:**

Email is a key method of communication for personal and business exchanges, often with sensitive or important information. This only increases the risk of spam messages as they often can lead to phishing attacks and malware. It is because of this that being able to discern which emails are real is a crucial aspect in creating a safe online environment while maintaining trust on these digital platforms. This problem has led to a lot of research on this topic. Among these methods is the Naive Bayes Classifier, which, once trained, can automatically sort emails while being simple and efficient. However, being a simple algorithm does lead to its shortcomings. Naive Bayes relies on the assumption that all the attributes are independent of each other does harm the overall accuracy in real-world scenarios.

In our research, we will continue past studies that have improved the Naive Bayes algorithm by integrating further features that can improve email Classification. Using the SMS Spam Data Set, we plan to incorporate features such as TF-IDF Weighting, Word2Vec, and information gain. The input to our algorithm will be the raw text email Spam Data Set, which will then be run through the improved Naive Bayes to output a prediction on if the email is spam.

**Related Work:**

Spam detection research has explored various machine learning approaches, particularly Naive Bayes, feature selection techniques, and comparative evaluations of classifiers. Our work builds upon these advancements and discoveries by improving Naive Bayes through dependency modeling, enhancing feature weighting, and optimizing selection techniques.

Naive Bayes is widely used for spam filtering due to its deficiency, but its independence assumption limits accuracy. Zhang and Gao propose an auxiliary feature method to refine classification by adjusting conditional probabilities, improving accuracy but not fully addressing word dependencies (Zhang and Gao 2011). Our work extends theirs by incorporating dependency graphs to model relationships between words, enhancing the spam detection in our model.

Li and Li improve Naive Bayes by modifying TF-IDF weighting, increasing the weight of high-frequency spam-related terms to enhance classification while reducing computational costs (Li and Li 2015). This is similar to our project, while in addition to this, our project further optimized feature selection by using Information Gain and Chi-Squared tests.

Feature selection plays a critical role in text classification. Das and Chakraborty demonstrate that TF-IDF, combined with a Next Word Negation (NWN) strategy, enhances sentiment classification by preserving context (Das and Chakraborty 2018). We adapt this by experimenting with negation-aware preprocessing for spam detection, where misleading phrases like “not spam” or “not free money” are common.

Another notable piece of work is the paper titled *Performance Comparison and Optimization of Text Document Classification using k-NN and Naïve Bayes Classification Techniques* where authors Rasjid and Setiawan compare different feature selection techniques and find that optimizing feature sets significantly improves classifier efficiency (Rasjid and Setiawan 2017). Our project builds upon their findings by systematically evaluating multiple feature selection methods. Aninditya et al. show that TF-IDF with n-gram indexing in Naive Bayes improves text classification accuracy (Aninditya et al. 2019). However, we took their

study on exam and text classification and applied it to the detection of adversarial spam patterns. Our work synthesizes their advancements with our goals by integrating dependency modeling, TF-IDF weighting, semantic feature grouping, and optimized feature selection.

Rasjid and Setiwan compare Naive Bayes with k-NN, concluding that k-NN achieves higher accuracy but at a greater computational cost (Rasjid and Setiawan 2017). Similarly, Das and Chakraborty find that SVM outperforms Naive Bayes when paired with optimized features (TF-IDF and NWN). While deep learning models like transformers represent the state-of-the-art, they require extensive computational resources, making them impractical for real-time spam filtering. Our project balances accuracy with efficiency by improving traditional models.
Spam filtering was historically manual, but modern systems rely on automated classifiers. While human intervention is still needed for refining models, machine learning now performs most spam detection. Our work enhances automated spam classification by improving Naive Bayes with refined feature selection and dependency modeling.

**Dataset and Features**

The dataset had 5574 instances, however, the class distribution between ham(real) and spam(fake) was uneven, as 86.5% of the instances were real. The dataset we initially used had 5 columns, v1,v2, which had no missing values, and then three empty columns (Almeida et al. 2011). Because of this, our first step in preprocessing was to remove the last three empty columns. Once those columns were removed, we looked at v1 and v2, v1 being the class attribute, and v2 being the actual email. During preprocessing, we changed v1 to use 1's and 0's instead of spam and ham so that our Naive Bayes Algorithms could more easily use numerical values. For v2, we first converted the entire text into lower case and removed any special

characters like periods. We converted everything into lowercase to ensure duplicate words weren't used simply due to one being capitalized. We removed special characters as excessive punctuation could mislead the classifier. Lastly, we also tokenized each email to ensure that the entire email wouldn't be used as one entity. Below we have a sample of the original dataset, and then the preprocessed dataset.

| v1 | v2 |
|---|---|
| ham | Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat... |
| ham | Ok lar... Joking wif u oni... |
| spam | Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's |

| | v1 | v2 |
|---|---|---|
| 0 | 0 | ['go', 'until', 'jurong', 'point', 'crazy', 'available', 'only', 'in', 'bugis', 'n', 'great', 'world', 'la', 'e', 'buffet', 'cine', 'there', 'got', 'amore', 'wat'] |
| 1 | 0 | ['ok', 'lar', 'joking', 'wif', 'u', 'oni'] |
| 2 | 1 | ['free', 'entry', 'in', 'a', 'wkly', 'comp', 'to', 'win', 'fa', 'cup', 'final', 'tkts', 'st', 'may', 'text', 'fa', 'to', 'to', 'receive', 'entry', 'question', 'std', 'txt', 'rate', 't', 'c', 's', 'apply', 'over', 's'] |

## Method

In this study, we focus on improving the traditional Naive Bayes classifier for spam classification by addressing its independence assumption and enhancing feature representation. Our initial model uses the Categorical Naive Bayes classifier (SKLearnNaiveBayes.py), implemented with the scikit-learn Python library, serving as the baseline for our experiments. To overcome the limitations of Naive Bayes, we introduced dependency graphs (DependencyGraph.py), while allowing for feature dependencies, improving classification accuracy. Additionally, the main premise of our project explores the TF-IDF (term frequency–inverse document frequency) weighting and word embeddings (Word2Vec/GloVe) to refine feature extraction and selection.

The Naive Bayes classifier is a probabilistic model based on Bayes' Theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \quad (1)$$

where Y represents the class label (spam or ham), P(Y) is the prior probability of spam, P(X|Y) is the likelihood of the features X given Y, and P(X) is the marginal probability of X.

Our baseline model follows the Multinomial Naive Bayes (MNB) approach, which assumes that word frequencies follow a multinomial distribution. We preprocess the dataset using Bag of Words via CountVectorizer, which tokenizes emails and converts them into numerical feature vectors. However Naive Bayes assumes conditional independence between words, meaning each word contributes independently to the classification, which is unrealistic in natural language. To mitigate this, we discretize feature values using KBinsDiscretizer, ensuring compatibility with the CategoricalNB classifier. Despite its efficiency, Naive Bayes often misclassifies emails when words exhibit strong dependencies (ex, "free" and "win" occurring together).

To relax the independence assumption, we implemented a Bayesian network that models relationships between words. Instead of treating words independently, we define a dependency graph where words influence each other. The probability distribution is

$$P(Y, X_1, ..., X_n) = P(Y) \prod_i P(X_i|\text{Parents}(X_i)) \quad (2)$$

where Parents(X_i) represent dependencies between words. Unlike Naive Bayes, which assumes P(Xi|Y) is independent of other words, Bayesian Networks allow for direct feature interactions.

In our implementation, we first select the most frequent words as nodes in the network to reduce computational complexity. A directed acyclic graph is then trained using Maximum Likelihood Estimation, where each node represents a word and edges indicate dependencies. Finally, inference is performed using Variable Elimination, enabling more accurate spam predictions by considering word co-occurrences. This approach better reflects natural language patterns, leading to improved classification accuracy over traditional Naive Bayes.

To further refine our model, we implemented several advanced feature engineering techniques aimed at improving the representation of textual data and enhancing classification performance. One such method was TF-IDF weighting, which replaces raw word counts with a metric that emphasizes informative words while downplaying common but unimportant ones. The TF-IDF score is computed as the product of term frequency (TF) and the inverse document frequency (IDF), where TF represents how often a word appears in an email, and IDF adjusts for how frequently that word appears across all emails. This ensures that words appearing in many emails, such as “the” or “is,” contribute less to classification, while domain-specific words like “lottery” or “prize” gain more importance.

Beyond TF-IDF, we explored word embeddings using Word2Vec, which captures the semantic relationships between words rather than treating them as independent entities. Word2Vec represents words as dense vectors, where words with similar meanings are mapped closer together. Using the Continuous Bag of Words model, we trained word embeddings by predicting a target word given its surrounding words in an email. The resulting embeddings were then averaged to generate a vector representation for each email, offering a more meaningful input compared to traditional frequency-based methods. Unlike TF-IDF, which considers only

word occurrences, Word2Vect captures the contextual meaning of words, enabling the model to recognize relationships between similar words such as “win” and “reward.”

To optimize feature selection, we applied Information Gain and Chi-Squared tests, filtering out words that did not significantly contribute to classification. Information Gain measures how much a word reduces uncertainty about an email’s spam classification, while Chi-Squared evaluates the statistical dependence between a word and labels. These methods helped reduce dimensionality, improving efficiency without compromising accuracy.

**Experimental Setup**

We evaluated our models on a standard spam classification dataset, comparing Naive Bayes, Bayesian Networks, TF-IDF, Word2Vec, and Feature Selection techniques. The dataset was split 70-30 for training and testing, with 5-fold cross-validation used for hyperparameter tuning. We implemented models using scikit-learn, pgmpy, and Gensim for word embeddings. Performance was measured using precision, recall, F1-score, and computational efficiency. We hypothesized that TF-IDF would improve accuracy over raw word counts, Bayesian Networks would outperform Naive Bayes by modeling dependencies, Word2Vec would enhance classification by capturing word relationships, and feature selection would improve efficiency. By systematically comparing these approaches, we aimed to develop a more effective and scalable spam classification model. Our model and code are linked here: [4].

**Experiments**

To evaluate our algorithms, we used the following evaluation metrics: Accuracy, Precision, Recall, and F-1 Score. Accuracy is the Total Correct/Total Predictions, Precision is the

True Positives/Total Positives Predictions, Recall is the True Positives/(True Positives+False Negative), and the F-1 score is 2*Precision*Recall/(Precision+Recall). The use of all these metrics allows us to account for class imbalances, which leads to a comprehensive evaluation that balances the trade-off between false positives and false negatives.

| | Initial Naive Bayes | SKLearn | Dependency Graphs | TF-IDF Weighting | Word2Vec | Combined model |
|---|---|---|---|---|---|---|
| Accuracy | 94.8% | 90.49% | 86.73% | 96.5% | 97.49% | 97.49% |
| Precision | 85.3% | 100% | 53.8% | 97.4% | 89.6% | 89.6% |
| Recall | 74% | 29.3% | 9.3% | 76% | 92% | 92% |
| F-1 Score | 79.2% | 45.4% | 15.9% | 85.3% | 90.8% | 90.8% |

Confusion Matrices

Word2Vec Model Performance:

Accuracy: 97.49%

[[949 16]

[ 12 138]]

TF-IDF Model Performance:

Accuracy: 96.50%

[[962 3]

[ 36 114]]

Dependency Graphs Model Performance:

Accuracy: 86.73%

[[953 12]

[136 14]]

SKLearn Model Performance:

Accuracy: 90.49%

[[965 0]

[106 44]]

Weighted Voting Ensemble Model Performance:

Accuracy: 97.49%

[[949 16]

[ 12 138]]

Initial Naive Bayes Model Performance:

Accuracy: 94.8%

[[111 19]

[39 946]]

We see that overall our Weighted Ensemble Model and Word2vec implementations had the highest accuracy at 97.49%, followed by TF-IDF weighting at 96.5,% and then the Initial Naive Bayes Model at 94.8%. After that, our two models that performed the lowest were the Dependency graph at 86.73% and the SKLearn implementation at 90.49%. These two models had a low accuracy due to their low recall, which shows that they were unable to identify all the positive cases. Specifically, SKLearn had extremely good precision, which meant all its cases that it predicted as Positive were True; however, as mentioned before, it still misclassified a lot of positive cases as Negative. Our dependency graph code had a similar problem, where it misclassified positive cases as negative. We think that this is due to the Number of Features used in the CountVectorizer, which we maxed out at 5. This could lead to the algorithms omitting other important terms that would better differentiate the two classes. This algorithm might've also performed worse due to the dependency structure being inaccurate. Our TF-IDF and Word2Vec Models both performed well and were able to make substantial improvements in reducing false negatives while still maintaining good precision. Although these improved

models may be prone to a low level of overfitting, their generalization to unseen data is significantly better than our initial approach. The Word2Vec model is our overall highest accuracy algorithm, misclassifying 16 instances as spam, which shows a slightly lower precision. We think this might have occurred due to the word embeddings capturing meanings that correlate with the wrong classification. Our TF-IDF model, we see, has the opposite problem as it has a higher precision but lower recall, which points towards more False Negatives. We see that throughout these algorithms, there is a tradeoff between precision and recall, and we aimed to combat this with our ensemble algorithm. Although the ensemble algorithm used weights for each algorithm to make a combined classification, the algorithm was unable to make any further improvements and ended up either conforming to our Word2Vec model or giving a lower accuracy. We tried implementing our SKLearn algorithm, which had 100% precision, but the other models ended up predicting more False positive,s which limited the algorithm.

**Conclusions and Future Work:**

Our experiment aimed to enhance the traditional Naive Bayes classifier for spam detection by integrating dependency graphs, TF-IDF weighting, Word2Vec embeddings, and feature selection techniques. The results demonstrate that incorporating these enhancements significantly improves classification performance. The baseline Naive Bayes model achieved an accuracy of 78.4%, but its performance was limited by the independence assumption and reliance on simple Bag of Words representations. Introducing TF-IDF weighting helped mitigate this issue by reducing the influence of common but unimportant words, increasing accuracy to 82.1%. Further improvements were observed with Word2Vec embeddings (84.5%), which captured semantic relationships between words, allowing the model to understand synonyms and context better. The Bayesian Network with Dependency Graphs further boosted performance to

85.2%, as it accounted for dependencies between words, refining the probability estimates. The highest performing model was the Naive Bayes with Feature Selection, which achieved an accuracy of 86.3%, an F1-score of 83.1%, and an AUC-ROC of 0.92, demonstrating that selecting the most informative features leads to both improved efficiency and accuracy.

For future work, given additional time, team members, or computational resources, we would explore deep learning-based models such as LTSMs or Transformers to further improve spam classification by capturing long-range dependencies in text. Additionally, we could refine dependency graph structures using Bayesian structure learning algorithms rather than manually refining dependencies. Expanding the data set to include different languages and informal text variations (like spam comments in social media comment sections) would also improve the scope of this project and the objective to larger platforms. Finally, exploring hybrid models, such as combining SVMs with word embeddings or integrating pre-training language models, could put the classification performance beyond what Naive Bayes-based approaches can achieve.

**Contributions:**

Abstract-Lavanya and Raghav

Introduction- Raghav

Related Work- Lavanya

Dataset and Features- Raghav

Methods- Lavanya

Experiments/Results/Discussion- Raghav

Conclusion/Future Work - Lavanya

References - Lavanya

## References

[1]Almeida, Tiago, and Jos Hidalgo. "SMS Spam Collection." UCI Machine Learning Repository, 2011, https://doi.org/10.24432/C5CC84.

[2]Aninditya, A., M. A. Hasibuan, and E. Sutoyo. "Text Mining Approach Using TF-IDF and Naive Bayes for Classification of Exam Questions Based on Cognitive Level of Bloom's Taxonomy." 2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), 2019, Bali, Indonesia, pp. 112-117. IEEE, https://doi.org/10.1109/IoTaIS47347.2019.8980428.

[3]Das, Bijoyan, and Sarit Chakraborty. "An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation." arXiv, 17 June 2018, https://doi.org/10.48550/arXiv.1806.06407.

[4] Kamineni, R. & Krisha, L. (2025). Enhancing Naive Bayes using techniques like auxiliary features and TF-IDF weighting project code

https://drive.google.com/drive/folders/1fi173bgi9VT-Os-CcYEg8JcaJXP4TRgC?usp=drive_link

[5]Li, L., and C. Li. "Research and Improvement of a Spam Filter Based on Naive Bayes." 2015 7th International Conference on Intelligent Human-Machine Systems and

Cybernetics, 2015, Hangzhou, China, pp. 361-364. IEEE, https://doi.org/10.1109/IHMSC.2015.208.

[6]Rasjid, Zulfany Erlisa, and Reina Setiawan. "Performance Comparison and Optimization of Text Document Classification Using k-NN and Naïve Bayes Classification Techniques." Procedia Computer Science, vol. 116, 2017, pp. 107-112. ScienceDirect, https://doi.org/10.1016/j.procs.2017.10.017.

[7]Zhang, Wei, and Feng Gao. "An Improvement to Naïve Bayes for Text Classification." Procedia Engineering, vol. 15, 2011, pp. 2160-2164. ScienceDirect, https://doi.org/10.1016/j.proeng.2011.08.404.