A Novel Approach for Phishing Detection Using Natural Language Processing (NLP)

Ryan M. Adolfs Anthony P. Jotautas
Department of Computer Science, University of Alabama, USA
{apjotautas, rmadolfs}@crimson.ua.edu

***Abstract*—Phishing emails continue to pose an immediate and modern threat to global cybersecurity. This paper investigates the effectiveness of various Natural Language Processing (NLP) and Machine Learning (ML) techniques for detecting phishing emails. Three NLP techniques (N-grams, Bag-of-Words, Term Frequency-Inverse Document Frequency) are evaluated across three different ML models: Logistic Regression (LR), Random Forest (RF), and Support Vector Machine (SVM). We apply 18 unique model-vectorization combinations to a dataset of almost 82,500 emails to assess their performance on accuracy, precision, recall, and F1 score. Our results show that the combination of TF-IDF with a (1,2) N-gram range with SVM achieves the best performance metrics, achieving the highest accuracy of 99.19% and outperforming the second-best combination with a 38.17% reduction in error rate. These findings suggest that SVM combined with TF-IDF vectorization is well suited and effective for dealing with phishing detection, especially when paired with an optimized N-gram configuration.**

***Index Terms*—cybersecurity, detection, feature extraction, logistical regression, machine learning, natural language processing, phishing email, random forest, spam, support vector machines**

I. INTRODUCTION

PHISHING attempts have evolved to be one of the most prevalent and destructive forms of cyberattacks since first gaining popularity amongst hacker groups in the mid-1990s [1]. Phishing refers to attempts to steal sensitive data or install malware through solicitation where the attacker uses social engineering and mimicry to masquerade as a legitimate and trustworthy source. Phishing can be attempted through a variety of platforms and services such as email, SMS texting, phone calls or VoIP, or even video calls [4]. With the advancement of AI technology, phishing has become as complex and dangerous as it ever has with deepfake voice and video scams becoming a new face in cybercrime [5], fooling more than just the average person with realistic impressions of close family and friends.

Despite the advancement in security techniques and societal awareness, these online attacks have grown over the past few decades, phishing has become only more frequent and damaging. In 2024 alone it was reported that over US$2.7B [2] was lost from targeted business email compromises and that over US$70M [2] was lost from phishing emails sent in mass. It is also estimated that over 3B phishing emails are sent daily and are used in over 90% of cyberattacks [3].

Currently, Gmail considers the "IP address, domains/subdomains, [and] sender authentication" [6] of the incoming email for their AI-driven spam filters as well as user input such as reports of suspected spam. This paper will focus on the efficacy of using the subject, body, date, of the email as the features for ML algorithms instead to see if those are just as effective or perhaps more effective than existing approaches.

We will be processing a combination of six different datasets using three different Natural Language Processing (NLP) techniques combined with three ML methods for a total of eighteen different methods. The rest of this paper is as follows. In Section II, we discuss the six datasets used in this work. Section III presents guiding research questions for this work. In Section IV, we discuss the methodology for processing the datasets. Section V discusses the results and analysis of the three different algorithms used and how each of the six NLP techniques affected each. In Section VI, we give a comparison of our results to previous work. Section VII describes the shortcomings with our work. In Section VIII we discuss what can be done in future work. Section IX concludes our paper with key findings and possible directions for future work.

II. DATASETS

For our work we used a compiled dataset by Al-Subaiey et al. [7] which contains emails from six datasets that were combined to create a comprehensive and varied dataset for analysis. Table 1 presents the number of legitimate and phishing emails in each dataset and the ratio of legitimate emails to spam emails.

TABLE I
SUMMARY OF DATASETS USED

Dataset	CEAS-08	Assassin	Enron	NiFr	Ling	Nazario	Final
Legit	17,312	4,091	15,791	0	2,401	0	39,595
Spam	21,842	1,718	13,976	3,332	458	1,565	42,891
Legit:Scam	**44:56**	**70:30**	**53:47**	**0:100**	**84:16**	**0:100**	**48:52**
Total	39,154	5,809	29,767	3,332	2,859	1,565	82,486

A. *CEAS-08 Dataset*

The Conference on Email and Anti-Spam (CEAS) 2008 Challenge Lab Evaluation Corpus originally consisted of 137,705 messages which were collected anonymously from August 5 to August 8 in 2008 [8]. There is a broad range of different phishing/spam techniques used. However, the age of the dataset may contribute to models trained on this dataset to find difficulty detecting modern phishing techniques.

B. *Apache SpamAssassin (Assassin) Repository*

The SpamAssassin Public Corpus is a free and available collection maintained by Apache Software Foundation which contains legitimate and spam emails from 2002 to 2006. It is a common benchmark for comparing ML techniques as it is pre-labelled. However, it is relatively small compared to other datasets and its age may affect a model's performance for modern phishing emails.

C. *Enron Phishing Email Dataset*

The Enron Dataset (2006) was made public while the Enron corporation was under global investigation for internal fraud. Originally containing 619,446 messages, 29,767 were selected for this work [10]. Benefits of this dataset are its size and availability, however there may be biases for its lack of diversity as it solely comprises the emails of Enron employees.

D. *Nigerian Fraud (NiFr) Dataset*

A collection of "Nigerian" fraud letters dating from 1998 to 2007 [11]. Contains a specific type of phishing attempt to convince the recipient to pay a fee in advance for a promised reward. This dataset is limited in size and only contains one specific type of email.

E. *Ling-Spam Corpus*

The Ling-Spam (2000) dataset is focused on topics of interest to linguists. These were collected by the Linguist List, "a moderated mailing list about the science and profession of linguistics" [9]. They contain job postings, software availability announcements, and "even flame-like responses". This dataset serves as a common benchmark for ML algorithms however its limited size may hinder models in generalized applications.

F. *Nazario Phishing Corpus*

The Nazario dataset (2015-2022) is a collection of phishing emails from a personal inbox which is meant to be representative but not exhaustive [12]. While varied and focused on phishing emails, the size is limited which may cause the model to struggle against generalized applications.

G. *Preprocessing of Text*

After the careful selection of these datasets to build a well-rounded and diverse range of data, Al-Subaiey et al. [7] edited the dataset with several key preprocessing techniques which prepared it to be read for training the models. The first step was cleaning the text by removing irrelevant characters such as HTML tags, punctuation, excess whitespace, and stop words. The text was also made all lowercase to improve uniformity. By removing noise in the data it allows the model to focus on more relevant and important patterns. Emails with duplicate bodies were culled from the dataset along with emails that contained no message body at all.

The Enron and Ling datasets only have 'Subject' and 'Body' features while the CEAS-08, SpamAssassin, Nigerian Fraud, and Nazario datasets included the features 'Sender', 'Receiver', 'Date', 'Subject', 'Body', and 'URLs'. Al-Subaiey et al. [7] took the 'Subject' and 'Body' features from the Enron and Ling datasets and combined them into a single feature called 'text_combined'. For the CEAS-08, SpamAssassin, Nigerian Fraud, and Nazario datasets they took the 'Sender', 'Date', 'Subject', and 'Body' features which were also combined into 'text_combined". Then, all six of these datasets were merged into one final dataset 'phishing_email.csv'.

III. RESEARCH QUESTIONS

To address the broader objective of identifying the optimal combination of NLP and ML methods for phishing detection, this study presents 4 research questions (RQs). These questions are designed to systematically evaluate the effectiveness of various configurations of NLP and RL methods.

RQ1: *What combination of NLP and ML methods yields the best overall performance for phishing email detection?*

— Synthesizing the results of the below 3 questions, we identify the optimal configuration of NLP preprocessing, ML algorithm and N-gram range. This integrative evaluation provides guidance on methodological best practices for future phishing email detection tasks and applications.

RQ2: *How do varying levels of local textual content (N-gram ranges) influence the performance of different ML models in phishing classification?*

— We explicitly vary the scale of lexical context and document the performance implications. This exploration tests the importance of longer contiguous phrases in phishing emails (e.g. “verify your account”) provide greater discriminative power than isolated singular token analysis.

RQ3*: How do different NLP preprocessing methods affect the classification performance?*

— In addressing this problem, we investigate whether basic preprocessing methods like tokenization and TF-IDF weighting have any meaningful impact on predictive accuracy and generalization. Specifically, we explore how transitioning from a simpler approach like BOW to a more complex approach in TF-IDF affects key metrics such as accuracy, precision, recall, F1-score, and false positive/negative rates.

RQ4: *How do different machine learning algorithms impact phishing classification performance?*

— In scenarios where the local context level and NLP features are the same, we assess the performance of the 3 different ML models: logistic regression, random forest, and support vector machines. This comparison will clarify whether certain models inherently leverage text-based features better for phishing detection.

IV. Methodology

A. NLP Methods

Natural Language Processing (NLP) refers to the handling of natural language and its conversion into data. We explore 3 NLP techniques to process our dataset to feed into our ML algorithms, N-grams, Bag-of-Words (BOW), and Term Frequency - Inverse Document Frequency (TF-IDF). We used various methods of NLP to explore if certain techniques improved upon the performance of phishing detection as mentioned in our research questions.

N-grams refers to an ordered sequence of N contiguous items drawn from a longer sequence [15]. In our case, it describes how we tokenized words and phrases from the phishing emails. We decided to use N-grams because local context N-grams encode short-range syntactic and stylistic cues like “free trial” or “bank account” that a single word tokenization approach would overlook. We experimented with 3 different sets of N-grams ranges, one that included unigrams and bigrams (1,2), which would include single words and two word phrases, one that included unigrams, bigrams, and trigrams (1,3), and another that spanned unigrams through quadgrams (1,4). Our goal being to determine if shorter or larger phrases were more indicative of signaling whether an email was real or a phishing email. These N-grams would then be vectorized by the 2 NLP vectorization techniques BOW and TF-IDF.

Machine learning methods expect numerical features, so strictly giving them N-grams feature lists of contiguous local words and phrases would not be sufficient. Vectorization solves this problem. We utilize 2 vectorization techniques, BOW and TF-IDF.

BOW is a simpler technique, scanning every unique token (unigram) and contiguous token sequence (bigram, trigram, quadgrams) and creating a column index for each [16]. The constructed vector from these terms directly comes from the frequency they appear in the document with no normalization or weighting adjustments done to the frequencies. The advantages to this simpler technique include lightweight interpretability and is fast given it uses simple integer counting to determine its vectors [5b]. The drawbacks include high-dimensional sparse vectors, where each new word adds a column, and rare sequences of words in one document still get their own column, leading to huge mostly-zero matrices that can overfit and hog memory [17]. Additionally, BOW fails to note semantics or word order, leading to phrases with similar wording but opposite meaning being treated similarly. For example,“I love dogs” and “I don’t love dogs” would be treated similarly, despite having complete opposite meanings [16]. Regardless of the drawbacks, we expect this combination of N-grams and BOW to perform well.

TF-IDF utilizes two things, pure term frequency (TF) similar to BOW, and inverse document frequency (IDF), where it down-weights terms that appear in nearly every document (e.g. “The”, “and”, etc.). TF-IDF pairs naturally with N-grams as many bigrams and trigrams we care about (e.g. “free trial) are moderately common across phishing emails. IDF prevents them from eclipsing rarer yet more distinctive phrases [18]. One of the major advantages of TF-IDF includes the ability to balance common and rare terms. It amplifies the distinction of words to a specific document while down-weighting ubiquitous stop-words [19]. The drawbacks are similar to that of BOW, regarding the ignorance of word order and semantic meaning and the potential for memory intensive vector spaces [20].

B. ML Methods

We explore 3 different machine learning (ML) algorithms, logistic regression (LR), random forest (RF), and support vector machines (SVM). We aim to find which one produces the most accurate results across the varying NLP methods.

LR works via a single linear model that sums the weighted vectors of every N-gram and feeds that total through a sigmoid function [21]. This then produces a probability between 0 and 1 that the email belongs to the phishing class. If

the resulting probability is above a certain threshold, the document is flagged as phishing [21]. LR excels at handling huge, sparse matrices produced by our NLP methods given its linear nature.

RF builds hundreds of decision trees, each trained on a bootstrap sample of the training set and a random subset of N-gram features [22]. Every tree votes, the forest outputs the majority class or the average probability [22]. This behavior helps capture nonlinear interactions between phrases that linear models like logistic regression could miss. RF is memory heavier because every split replicates the feature space, projecting that it may take significantly longer to train.

SVMs seek the hyperplane that maximizes the margin between phishing and legitimate emails [23]. Given a new email vector, the signed distance to the hyperplane is computed via the equation w*x + b, a positive sign denotes phishing, the negative being a real email, with the magnitude as a confidence score. We utilized a linear kernel for the SVM, meaning no feature transformation is applied. Like RF, SVMs thrive on high-dimensional TF-IDF features, so it is predictable to say SVMs will perform well in the combinations we explore that utilize TF-IDF.

The following is a table summarizing some benefits and tradeoffs between the 3 selected ML methods:

TABLE II
COMPARISON OF ML METHODS

Criterion	Logistic Reg.	Random Forest	Linear SVM
Training speed	Fast	Medium	Fast-to-medium
Handles non-linear feature interactions	✗	✓	✗
Native probability output	✓	✓ (mean vote)	✗ (needs calibration)
Interpretability	Weight coefficients	Tree/forest feature importance	Support vectors / hyperplane
Works out-of-the-box with sparse N-gram TF-IDF	✓	✓ (larger RAM)	✓

For all of our ML methods, we utilized a 80-20 training-testing split, with 80% of our data (65,989) being used to train the models, and 20% (16,497) of it being used to evaluate the performance of the models. 80-20 is a widely used proportion across many ML applications, as it gives an adequate amount of data to train on and test on given our moderately large dataset of 82,486. It is also compatible with our chosen ML models as they benefit from additional training rows with each extra training email reducing variance in the parameter or split estimates.

C. Approach

All our experiments were performed in a Jupyter notebook utilizing the SciKit-learn python library. Our pipeline approach to performing the experiments was as follows: We first loaded the datasets and cleaned them, held out data to achieve our 80-20 stratified split of data, then with the training data, we tokenized it and generated N-gram bags of ranges (1,2), (1,3), and (1,4), which were then mapped to sparse matrices via BOW (CountVectorizer) and TF-IDF (TfidfVectorizer). Then, we fed each combination of N-gram range and vectorizer method to each of our 3 ML methods to be trained, creating 18 different combinations of N-gram range-NLP Vectorizer-ML to be evaluated. Then, once training was finished, we evaluated them on the remaining 20% of data and produced statistics including accuracy, precision, recall, f1-score, as well as confusion matrices for each combination.

The results of these tests will be shown in the following section.

V. RESULTS AND ANALYSIS

We trained and evaluated different combinations of 3 NLP representations across 2 vectorization methods, BOW, TF-IDF, 3 N-gram ranges (1,2) (1,3) (1,4), and 3 ML methods, LR, RF, and SVM, to produce 18 total combinations. We display the results with the following metrics: Accuracy, precision, recall, F1-score, and total time taken (seconds) to train and evaluate.

A. Overall Performance Tables

Tables III, IV, and V show the results of each combination of NLP and ML methods, with each table representing a different N-grams range.

B. Addressing Research Questions

RQ1: *What combination of NLP and ML methods yields the best overall performance for phishing email detection?*

— From the above tables, you can see that the single best combination of the explored methods was N-grams of range (1,2), with TF-IDF vectorization using SVMs. This combination achieved an accuracy score of 99.19%, 0.5% higher than the best non-SVM combination. While 0.5% does not sound significant, it truly is, this significance will be discussed further in section 6.7.

We theorize this combination performed the best for the following reasons: First, the N-grams range of (1,2) sufficiently captured all semantically rich cues without sparsity blow-up. Second, the TF-IDF slightly down-weighs ubiquitous tokens, letting rarer alert phrases stand out. Third, SVM successfully exploits the near-linear separability of TF-IDF vectors and ignores irrelevant dimensions via the support-vector mechanism, successfully finding an effective hyperplane.

TABLE III

RESULTS FOR N-GRAMS RANGE (1,**2**)

NG (1,2)	BOW-LR	BOW-RF	BOW-SVM	TF-LR	TF-RF	TF-SVM
Accuracy	0.9869	0.9807	0.9858	0.9838	0.9788	**0.9919**
Precision	0.99	0.98	0.99	0.98	0.98	**0.99**
Recall	0.99	0.98	0.99	0.98	0.98	**0.99**
F1-score	0.99	0.98	0.99	0.98	0.98	**0.99**
Time (s)	66.03	2970.36	2609.45	**14.61**	3019.96	5541.8

TABLE IV

RESULTS FOR N-GRAMS RANGE (1,**3**)

NG(1,3)	BOW-LR	BOW-RF	BOW-SVM	TF-LR	TF-RF	TF-SVM
Accuracy	0.9869	0.9771	0.9848	0.9831	0.9744	**0.9912**
Precision	0.99	0.98	0.98	0.98	0.97	**0.99**
Recall	0.99	0.98	0.98	0.98	0.97	**0.99**
F1-score	0.99	0.98	0.98	0.98	0.97	**0.99**
Time (s)	69.08	4488.41	2521.91	**27.08**	3933.47	7160.65

TABLE V

RESULTS FOR N-GRAMS RANGE (1,**4**)

NG(1,4)	BOW-LR	BOW-RF	BOW-SVM	TF-LR	TF-RF	TF-SVM
Accuracy	0.9865	0.9756	0.984	0.9821	0.9725	**0.9912**
Precision	0.99	0.98	0.98	0.98	0.97	**0.99**
Recall	0.99	0.98	0.98	0.98	0.97	**0.99**
F1-score	0.99	0.98	0.98	0.98	0.97	**0.99**
Time (s)	63.04	5209.48	4489.36	**33.59**	5076.16	9919.41

RQ2: *How do varying levels of local textual content (N-gram ranges) influence the performance of different ML models in phishing classification?*

— Below in figure 1 a heatmap of the average performances of the various N-gram ranges across all combinations of NLP and ML methods. The heatmap indicates that with larger levels of local textual content, the performance marginally worsened across all metrics, with N-gram range (1,2) performing the best, achieving the highest average metrics of 98.5% across all 4 metrics.

We attribute these results to Bigram features already achieving high enough coverage to capture phrases like “click here” and “login link” that signal phishing. The larger context windows simply seemed to add more noise and reduce accuracy and effectiveness across all combinations of methods as they exploded the feature space. Additionally, the average time to train and evaluate significantly rose, increasing from averaging 2,370s, to 3,033s, to 4,131s as the N-gram range increased respectively.

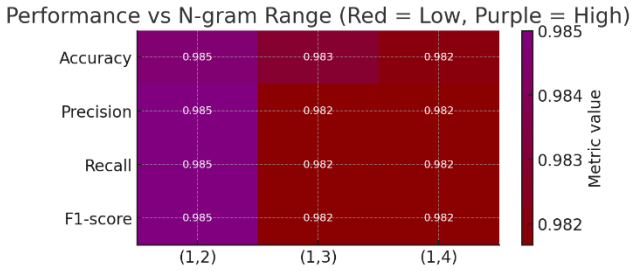

Fig. 1. Heatmap of Average N-gram range performance

RQ3*: How do different NLP preprocessing methods affect the classification performance?*

— TF-IDF achieved a minutely better accuracy (0.98322 vs 0.98314) than BOW and were evenly matched across the precision, recall, and F1-score metrics. This result was most likely due to the emails already containing highly class-specific tokens, where the raw counts alone were informative enough to achieve satisfactory levels of accuracy. Though, it is important to note that the highest achieved accuracy was with the use of TF-IDF. Thus, indicating that with models that fit well with the nature of TF-IDF vectors, a higher level of accuracy is achievable.

Below in figure 2 is a bar chart representing the average accuracy of these NLP methods across all combinations of methods.

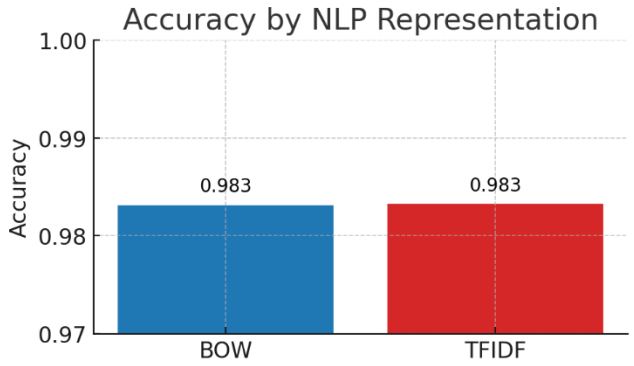

Fig. 2. Average Accuracy of NLP Vectorization Techniques

RQ4: *How do different machine learning algorithms impact phishing classification performance?*

— Across all NLP and N-gram ranges, we compared the average ML model accuracies across the combinations. It was found that SVM had the highest average accuracy by 0.3%. This occurred because SVMs exploit the geometry of the sparse high-dimensional space created by both BOW and TF-IDF vectors as that while the vectors are of high-dimensionality, they are almost linearly separable. LR shares the same linear decision surface but optimizes log-loss rather than hinge-loss, so it lands slightly short. Random Forests must evaluate thousands of token-presence splits; sparsity means most splits are empty, hurting both training time and

generalization.

Below in figure 3 is a bar chart displaying the disparity of average accuracies across the various NLP and N-gram combinations

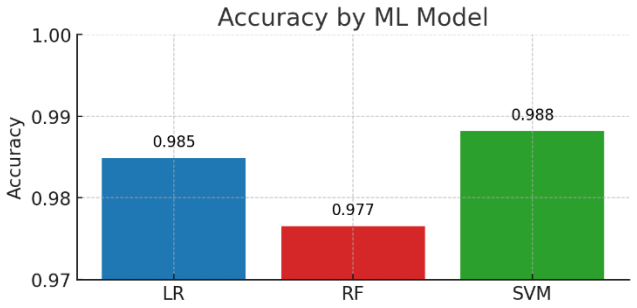

Fig. 3. Average Accuracy of ML Methods

C. Significance of Results

When comparing our two best distinct combination results, (TF-IDF+(1,2)+SVM vs BOW+(1,2)+LR), we saw an accuracy difference of 0.5% (99.19% vs 98.69%). However, this accuracy difference, while appearing minute, is not indicative of the real impact. To truly understand the magnitude of this difference, we must look at additional statistics including error rate, false positives, false negatives, and most importantly, relative error reduction.

1) Error Rate

Error rate is simply the total number of false positives and false negatives divided by the total samples. It can also be calculated by simply subtracting the accuracy from 1. The error rate allows us to quantify the number of mistakes each algorithm combination makes and compare them effectively.

2) False Positives

False positives refer to legitimate emails incorrectly being flagged as phishing. Minimizing these is significant as incorrectly filtering out a potentially important email can have significant impacts on the user in the case of email spam filters. Potentially causing a user to miss a bill payment or an important work email from a boss or colleague.

3) False Negatives

False negatives refer to phishing emails falsely being flagged as legitimate emails. False negatives carry far higher cost than false positives. While false positives are still of relative importance, minimizing false negatives is of utmost importance as a missed false negative can be catastrophic. Credential theft, ransomware, and many other previously discussed impacts have been shown to produce potentially multi-million dollar fallouts and legal liabilities. So minimizing false negatives should be the most important statistic to minimize.

4) Relative Error Reduction

Relative error reduction is a simple calculation, taking the error rate of the baseline minus the error rate of the model, and dividing it by the error rate of the baseline, you get a statistic that showcases by what percentage a model has reduced error. This effectively amplifies what may look like a minor improvement in accuracy, like our 0.5% increase in accuracy.

Below table VI showcases the difference in performance amongst these statistics across our top 2 models.

TABLE VI
SIGNIFICANT STATISTICS OF TOP 2 MODELS

	TF-SVM-(1,2)	**BOW-LR-(1,2)**
Accuracy	0.9919	0.9869
Error Rate	0.0081	0.0131
Rel. Err. Reduction	0.381679389	-0.617283951
False Pos.	58	127
False Neg.	76	89
Total Errors	134	216

As shown, the TF-SVM-(1,2) model **reduces the amount of errors by 38.17%**. This is massive, as 38.17% less errors indicates a potentially massive amount of preventative savings in phishing related costs over time.

Additionally, there was a reduction of 13 false negatives (14.6% reduction) and 69 false positives (54.33% reduction) from the 2nd best model to the first.

This showcases how much more effective our best model TF-SVM-(1,2) is in comparison to our other models.

VI. RELATED WORK

TF-IDF and SVMs have been tested on the same dataset that we used ourselves [7]. In the same scenario, our results were comparable, if not slightly better than theirs. They reported an achieved accuracy of 0.991 and precision, recall, and F1-score of 0.99 on their proposed SVM_991 model [7]. While our results reported an accuracy of 0.9919 and precision, recall, and F1-score of 0.99. This supports our conclusion that the combination of TF-IDF NLP preprocessing and SVM model is the most effective of that we tested, while also potentially showcasing that expanded N-grams ranges from single tokens (unigrams) to the inclusion of bigrams can expand the effectiveness of these techniques. Rabbi et al. [14] use TF-IDF vectorization along with LR, RF, and 4 other algorithms on the Ling-Spam dataset and experiment with excluding the ‘Subject’ feature. Their best performing

algorithm was RF, however, their accuracy did not exceed 98.6% and their F-score did not exceed 98.59% in their testing. Champa et al. [13] focuses on the preprocessing and curation of datasets to be easily accessible for use with algorithms. They also experiment with the importance of features like 'Subject', 'Body', and 'Sender' to see which is the most impactful on the efficacy of algorithms. Champa et al. [13] used SVM however it performed the worst out of the five algorithms they tested (including RF). One reason may be that the SVM struggles with larger feature sets [24].

VII. Threats To Validity

The datasets we used were entirely in English which may affect the multi-lingual application of our findings. Creating multi-lingual datasets in the future would further the progress of phishing email research and perhaps help strengthen the detection of spam emails.

Most of our data also contained emails older than a decade. The age of these emails could bias our work to struggle with modern phishing emails especially since phishing has only grown harder to detect as time has gone on. If we had datasets of modern phishing emails from within the past five years, it would strengthen confidence that our work could be used in modern applications.

We also only considered the 'Subject', 'Body', 'Sender', and 'Date' features which were combined into a single feature for our analysis. If we kept the features separate or perhaps incorporated datasets with different features, we could look at which features are most important and meaningful to strengthen the precision of our work.

Another possible shortcoming was that we did not fine tune the models. This could have affected algorithms like RF which have hyperparameters that can improve the performance [25].

VIII. Future Research

Curating or sourcing a dataset with modern and tricky phishing emails would benefit phishing detection research greatly. A continuous model which dynamically integrates new and recent data would be able to work most effectively against modern and recent phishing attacks.

Another area of research could be using deep learning models for phishing detection. LR, RF, and SVM are traditional ML methods and future studies could integrate neural networks in hopes of detecting more challenging patterns in modern phishing emails. For example, phishing emails with subtle linguistic and psychological tactics could be harder to detect with traditional models. However, deep learning has greater potential for dealing with these borderline phishing cases.

Other traditional models could be experimented with too like Naive Bayes (NB) or variants like Multinomial or Bernoulli NB.

IX. Conclusion

In conclusion, phishing continues to be a growing and dangerous threat which urges the importance of stopping the attacks before they ever reach the recipient. Adjusting modern algorithms for spam filters to stay robust with the evolution of cybercrime is a global necessity. NLP and ML algorithms pose a significant possibility for becoming the standard in defending against phishing.

Our work compliments related research [7] [13] [14] in solidifying the argument that SVM with TF-IDF tokenization is the best solution for choosing a model to detect phishing emails. We also improve upon it by giving experimenting with N-grams, determining that an N-gram range of (1,2) with TF-IDF SVM was the best combination of methods and models. Using this combination, we achieved a 99.19% accuracy rate and a 38.17% reduction in errors compared to our 2nd best distinct combination of methods which was BOW with LR and an N-gram range of (1,2). We also achieved a 14.6% reduction in false negatives which is significant in ensuring the least amount of phishing emails bypass our methods. A N-gram range of (1,2) worked best over larger ranges with not only quicker model training but better in all of our measured performance metrics. We also found that vectorization techniques (TF-IDF, BOW) and their performance metrics were more significantly affected by the ML model used rather than the vectorization technique itself. This is evident by TF-IDF having almost identical performance metrics to BOW when results were averaged across all the combinations used.

References

[1] G. Ollmann. (2007). "The Phishing Guide: Understanding and Preventing Phishing Attacks" [Online]. Available: http://www.technicalinfo.net/papers/Phishing.html

[2] Internet Crime Complaint Center, "2024 Internet Crime Report", FBI, Washington, D.C., USA, 2024. Available: https://www.ic3.gov/AnnualReport/Reports/2024_IC3Report.

[3] "Data finds only 14% of domains worldwide truly protected from spoofing with DMARC enforcement, an increase from 2020." VALIMAIL.com., San Francisco, CA., USA., Mar. 22, 2021. Accessed: Mar. 18, 2025. [Online.] Available: https://www.valimail.com/newsroom/valimail-report-reveals-3-billion-spoofed-emails-are-sent-every-day

[4] E. Baptista, E. Cao, and C. Fernandez, "'Deepfake' scam in China fans worries over AI-driven fraud." Reuters, Beijing, China, May 22, 2023. Accessed: Apr. 7, 2025. [Online.] Available: https://www.reuters.com/technology/deepfake-scam-china-fans-worries-over-ai-driven-fraud-2023-05-22/

[5] S. Sjouwerman, "Deepfake Phishing: The Dangerous New Face Of Cybercrime." Forbes, Jan. 23, 2024. Accessed: Apr. 7, 2025. [Online.] Available: https://www.forbes.com/councils/forbestechcouncil/2024/01/23/deepfake-phishing-the-dangerous-new-face-of-cybercrime/

[6] N. Kumaran, “Understanding Gmail’s spam filters.” Google, May 27, 2022. Accessed: Apr. 7, 2025. [Online.] Available: https://workspace.google.com/blog/identity-and-security/an-overview-of-gmails-spam-filters

[7] A. Al-Subaiey, M. Al-Thani, N. Abdullah Alam, K. F. Antora, A. Khandakar, and S. A. Uz Zaman, “Novel interpretable and robust web-based AI platform for phishing email detection,” Computers and Electrical Engineering, vol. 120. Elsevier BV, p. 109625, Dec. 2024. doi: 10.1016/j.compeleceng.2024.109625.

[8] D. DeBarr and H. Wechsler, “Spam detection using Random Boost.” Pattern Recognition Letters, vol. 33, no. 10, pp. 1237–1244, 2012. doi: 10.1016/j.patrec.2012.03.012.

[9] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos, “A memory-based approach to anti-spam filtering for mailing lists,” Information retrieval, vol. 6, pp. 49–73, 2003. doi: 10.1023/A:1022948414856.

[10] B. Klimt and Y. Yang, “The Enron Corpus: A New Dataset for Email Classification Research.” ECML, 2004, pp. 217–226. doi: 10.1007/978-3-540-30115-8_22.

[11] D. Radev, “CLAIR collection of fraud email (Repository).” ACL Data and Code Repository, ADCR2008T001, https://aclweb.org/aclwiki

[12] J. Nazario, “The online phishing corpus.” https://monkey.org/~jose/phishing/

[13] A. I. Champa, M. F. Rabbi and M. F. Zibran, "Curated Datasets and Feature Analysis for Phishing Email Detection with Machine Learning," *2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*, Mt Pleasant, MI, USA, 2024, pp. 1-7, doi: 10.1109/ICMI60790.2024.10585821.

[14] M. F. Rabbi, A. I. Champa and M. F. Zibran, "Phishy? Detecting Phishing Emails Using ML and NLP," *2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA)*, Orlando, FL, USA, 2023, pp. 77-83, doi: 10.1109/SERA57763.2023.10197758.

[15] Jurafsky, D. & Martin, J. H. *Speech and Language Processing*, 3rd ed. Draft, 2023 – Ch. 3 (“N-Gram Language Models”).

[16] Neural Ninja, “Bag of Words: Unpacking Textual Data - Let’s Data Science,” *Let’s Data Science*, Jun. 28, 2023. https://letsdatascience.com/bag-of-words/

[17] IBM, “Bag of words,” *Ibm.com*, Dec. 22, 2023. https://www.ibm.com/think/topics/bag-of-words

[18] “The 40 Most Dangerous Phishing Email Examples In 2024,” *caniphish.com*. https://caniphish.com/phishing-email-examples

[19] N. Ninja, “TF-IDF: Weighing Importance in Text,” *Let’s Data Science*, Jun. 30, 2023. https://letsdatascience.com/tf-idf/

[20] A. Simha, “Understanding TF-IDF for Machine Learning,” *Capital One*, Oct. 07, 2021. https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

[21]“Text Classification using Logistic Regression,” *GeeksforGeeks*, Mar. 04, 2024. https://www.geeksforgeeks.org/text-classification-using-logistic-regression/

[22] A. A. Akinyelu and A. O. Adewumi, “Classification of Phishing Email Using Random Forest Machine Learning Technique,” *Journal of Applied Mathematics*, vol. 2014, pp. 1–6, 2014, doi: https://doi.org/10.1155/2014/425731.

[23] T. Team, “Spam Detection using SVM - TechVidvan,” *TechVidvan*, Sep. 14, 2022. https://techvidvan.com/tutorials/spam-detection-using-svm/

[24]J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, “A comprehensive survey on support vector machine classification: Applications, challenges and trends,” *Neurocomputing*, vol. 408, no. 1, pp. 189–215, Sep. 2020, doi: https://doi.org/10.1016/j.neucom.2019.10.118.

[25]P. Probst, M. N. Wright, and A. Boulesteix, “Hyperparameters and tuning strategies for random forest,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 3, Jan. 2019, doi: https://doi.org/10.1002/widm.1301.