Adversarial Email Generation against Spam Detection Models through Feature Perturbation

Qi Cheng*, Anyi Xu†, Xiangyang Li‡, Leah Ding§
*‡Information Security Institute, Johns Hopkins University, Baltimore, MD
†§Department of Computer Science, American University, Washington, D.C.
Email: *qcheng7@jhu.edu, †ax9843a@student.american.edu, ‡xyli@jhu.edu, §ding@american.edu

***Abstract*—Machine learning-based spam detection models learn from a set of labeled training data and detect spam emails after the training phase. We study a class of vulnerabilities of such detection models, where the attack can manipulate a trained model to misclassify maliciously crafted spam emails at the detection phase. However, very often feature extraction methods make it very difficult to translate the change in the feature space to that in the textual email space. This paper proposes a new attack method of making guided changes to text data by taking advantage of findings of generated adversarial examples that purposely modify the features representing an email. We study different feature extraction methods using various Natural Language Processing (NLP) techniques. We develop effective methods to translate adversarial perturbations in the feature space back to a set of "magic words", or malicious words, in the text space, which can cause desirable misclassifications from the attacker's perspective. We show that our attacks are effective across different datasets and various machine learning methods in white-box, gray-box, and black-box attack settings. Finally, we discuss preliminary exploration to counter such attacks. We hope our findings and analysis will allow future work to perform additional studies of defensive solutions against this new class of attacks.**

I. Introduction

Recent studies have shown the vulnerability of machine learning models to adversarial attacks where small input perturbations lead to misclassification. For example, recent works in computer vision use gradient-based methods [1], [2] to generate adversarial attacks against image classification models. Concerning text classification models, machine learning classification models have been employed for modern spam email filters, and such spam filters are susceptible to adversarial attacks [3]. In a successful adversarial attack, perturbed versions of the original spam email text can get misclassified by the spam filter, thus bypass the detection.

Adversarial attack generation against spam filters is more challenging than that in image classification models because of the discrete nature of the text space. Image data is continuous, and perturbations can be introduced directly to maximize the loss based on the gradients in the numerical feature space. In contrast, text data is discrete and needs to be converted to numerical features for a classification model. Consequently, a major challenge in applying gradient-based methods to generate adversarial attacks against spam filters arises from the interpretation of the perturbations generated in the continuous feature space to information in the discrete text space. A small perturbation in the continuous feature space may not be transformed to an equivalent change in effect when dealing with a character or word in the discrete text space.

In this paper, we present an effective method for crafting adversarial spam emails against machine learning-based spam filters. We propose an attack method for text data by leveraging gradient-based methods studied in the computer vision domain and employing NLP techniques for crafting adversarial examples in text space and modifying spam email content to bypass the spam filters. We investigate and compare different natural language feature extraction approaches such as TF-IDF (term frequency-inverse document frequency) [4], Word2vec [5], and Doc2vec [6] in white-box attacks. We propose a new method that enables identifying the changes needed in an email for successful perturbations in the feature space by modifying the Word2vec and Doc2vec techniques in order to keep the mapping between words and features.

We study the effectiveness of our methods in gray-box and black-box attacks on spam filters built with different classification models including Support Vector Machine (SVM) models [7] [8], Bayesian classification models [9] [10], Decision Tree models [11], and Multilayer Perceptron (MLP) models [12]. Additionally, we study the attack's effectiveness on ensemble learning where the classification aggregates results from three learning models. Prior works have proposed ensemble learning method as a defense strategy against adversarial attacks [13]. We empirically show that our proposed attack is effective against ensemble learning models. Through experimental evaluations on different datasets, we demonstrate that the proposed approach can generate adversarial examples in the text space and effectively reduce the spam filter's detection accuracy.

In summary, we made the following contributions:

- For popular Word2vec and Doc2vec techniques, we propose using a single feature for every word that enables the identification of "magic words" for attack email generation based on adversarial perturbations obtained in the feature space.

- We study and gain insights into the dynamics of how the attack in the feature space relates to the attack in

the email space. We show a useful model of this threat by its cost and gain from the perspective of attackers

- To have a comprehensive understanding of the problem, we conduct experimentation with different feature extraction methods, classifiers, and spam datasets in a range of white, gray, and black-box attack scenarios.

II. Related Work

Successes of adversarial machine learning are evident in domains such as image classification [1] [2] [14], voice processing systems [15] [16] [17], or movement sensory data [18]. However, two significant challenges exist for their applications to evasion attacks to bypass computer attack detection systems, including the usually difficult interpretation of feature perturbations back to the changes to an original attack in the problem space and the need to preserve the functionality and maliciousness of an attack during this process.

In one effort to handle these challenges, Han et al. [19] proposed a bi-level optimization model that separately produces successful adversarial instances in the feature space and mutates original malicious network traffic to have feature values close enough to these instances based on given constraints. The allowed mutations are changing the packet interval-time, which is continuous, and are duplicating partial traffic, in which the changes are discrete. Other relevant efforts to evade malware detection modify certain components of a malware program while preserving the functionality of the malicious code. For example, Hu and Tan [20] represented a malware piece by binary features, each of which indicates whether a certain API function is used. Only allowing adding API functions, a generative adversarial network (GAN) model in black-box settings can generate adversarial malware examples by examining perturbed feature vectors that are successful.

Adversarial attacks on spam email detection are a special case that handles a large number of words, each of which is considered in the feature space. However, most existing approaches are basic, with some linguistic constraints, searching among the variants of an email by substituting or deleting words for NLP adversarial attack examples that can cause the desired change in classification [21]. Wittel and Wu [22] categorized such attacks into three types, *tokenization attacks* where spammers intend to disturb tokenization of the email content by splitting or modifying features, such as inserting extra spaces in the middle of the words; *obfuscation attacks* where the email content is obscured from the detector using encoding or misdirection; and *statistical attacks* where spammers attempt to skew the message's statistics to distract the detector. A popular statistical attack is proposed in [23], where a set of "good words", after being added to spam emails, can cause misclassifications by two types of statistical spam detectors: maximum entropy and naive Bayes filters. Other examples of statistical attack include [24] [25].

In [26], the authors proposed two new methods of crafting spam emails that take advantage of the knowledge gained in perturbing input features of an ML model. Preliminary results showed that one of these two methods can efficiently identify a set of "magic words" to add to a spam email, without changing its nature, in order to bypass a spam detector. This method is novel in that it bridges the findings of adversarial attacks in the feature space, e.g., TF-IDF vectors, back to changes needed to be made in the problem space, e.g., emails. In this present paper, we greatly expand this effort to characterize the changes in the feature and problem spaces using various feature extraction methods, classifiers, and datasets.

III. Attack Methodology

The workflow of our methodology is shown in Figure 1. We first conduct text cleaning to preprocess datasets by removing special characters from the raw datasets. We then conduct feature extraction to transform cleaned email texts into numerical feature vectors in the feature space. We utilize the Projected Gradient Descent (PGD) attack on the features of a set of spam emails to generate misclassifications. Those successful perturbations done on these spam emails are analyzed to identify a set of "magic words" that can be added to spam emails. These modified spam emails are fed to machine learning models to examine their effectiveness of bypassing the detection. In addition to white-box attacks on the SVM classifier used for the PGD attack, we study gray-box and black-box attacks where the target classifier is different, representing the scenarios in which the knowledge of targeted spam detection models is totally or partially unknown to the attacker.

A. Feature Extraction

In this step, we convert the text content of an email into a numerical feature vector, representing information of that email used for classification. There are many vectorization methods to convert text data into numerical vectors, and we study the commonly used ones to form our feature space: TF-IDF, Word2vec, and Doc2vec. However, necessary modifications are needed in order to support the interpretation of changes in the feature space to the text space.

1) TF-IDF: TF-IDF calculation includes calculating the term frequency (TF) and inverse document frequency (IDF) for each word in an email. TF is how often a word appears in one email, and IDF is how often a word appears in the corpus of an email dataset. The IDF term is smoothed to better handle common words appearing in every document. The higher frequency of a word in a particular file and the lower file frequency of the word in the entire file collection results in a higher TF-IDF value, which reflects the significance of the word or feature related to an email. For each email, a

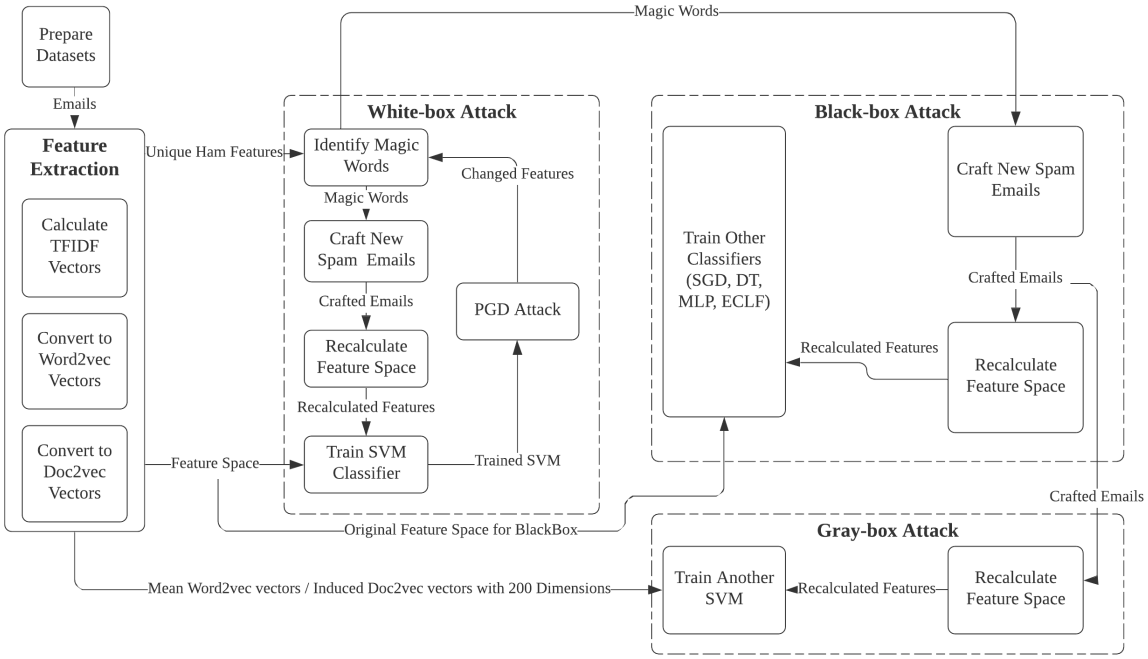

Figure 1. Workflow of generating adversarial perturbations, crafting adversarial emails, and conducting attacks

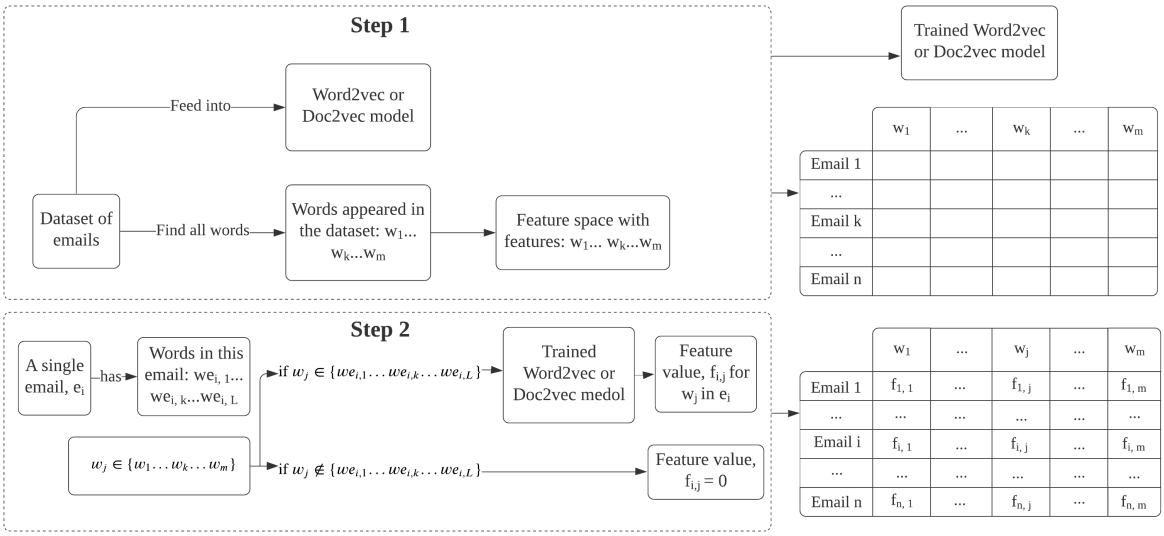

Figure 2. Constructing feature vectors for emails using Word2vec or Doc2vec technique with a single dimension for each word

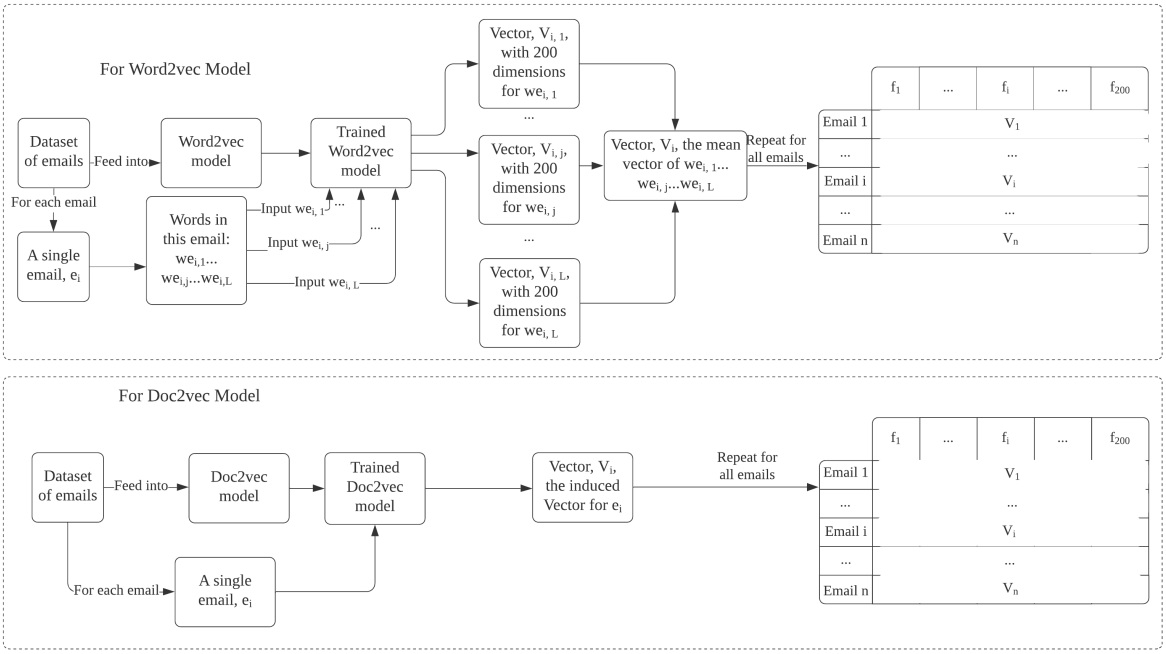

Figure 3. Constructing feature vectors for emails using Word2vec or Doc2vec technique with a 200 dimensions

vector of these TF-IDF features for all the words appearing in the dataset contains information useful to tell its nature, being normal or abnormal.

2) Word2vec: Existing Word2vec models can be trained to transform emails to feature vectors. As in many common applications, each word is converted into a vector with 200 dimensions. The steps for such a conversion are shown in the upper part of Figure 3. As a result, we can obtain a list of vectors where each vector represents one word in a given email. To represent one email with only one vector, we calculate the average values for each dimension in this list to form a new vector with 200 dimensions. After repeating this process with every email, we finally obtained a feature space with 200 dimensions.

Unfortunately, since features generated from this conventional approach do not map to individual words, we cannot use information gained during adversarial perturbations to the features to identify "magic words" for crafting spam emails. Therefore, we propose the following method to generate features using Word2vec, as shown in Figure 2 and in Algorithm 1. In our approach, we retrain the Word2vec model to map each word to a feature, i.e., a feature vector of a single dimension. Then we traverse every word in the training set and use the retrained model to calculate a numerical value for each word. The feature vector of one email is the union of the Word2vec values of all the words in the vocabulary, according to whether a word appears in this email or not. The corresponding feature has a value of zero if this word

Algorithm 1 Generating feature vectors for the emails with a single dimension to represent each word using a trained Word2vec or Doc2vec model

Input: $Dataset$, a list of emails; $w2v$, a trained Word2vec model, or a trained Doc2vec model
Output: $featureSpace$, a list of feature vectors for emails
procedure GENFEATUREVECTORS($w2v, Dataset$)
  $features$, Array
  for each $word \in Dataset$ **do**
    $features.append(word)$
  end for each
  $featureSpace$, Array
  for each $email \in Dataset$ **do**
    $featureVector$, Vector
    for each $word \in features$ **do**
      $featureValue$, Double
      if $word \in email$ **then**
        $featureValue = w2v.convert(word)$
      else
        $featureValue = 0$
      end if
      $featureVector.append(featureValue)$
    end for each
    $featureSpace.append(featureVector)$
  end for each
  return $featureSpace$
end procedure

is not contained in the email. In this way, we can obtain a feature space with each feature representing a different word, and each email generates a feature vector representing all the words appearing in it.

3) Doc2vec: While Word2vec computes a feature vector for a given word in the email dataset, Doc2vec computes the feature vector for a given email. It is a generalizing of the Word2vec method. As shown in Figure 3, when using Doc2vec as a feature extraction method, we first trained a Doc2vec model by feeding all the emails in the training dataset into the model. Secondly, we looped through the training set to find all the unique words in it. The set of unique words are the features in the feature space. Then for each email, all the words that appeared in the email and in the feature space are inputted to the trained Doc2vec model, and the model would output a vector with a single dimension for each of these words, which is essentially numeric values. All of these numeric values are combined in the feature vector of an email. In this way, the result from this approach with Doc2vec is essentially the same as the result of the second approach with Word2vec in terms of dimensionality; the result from Doc2vec also has a feature for each word and a vector for each email.

The other approach of using Doc2vec as a feature extraction method is more traditional but cannot be used to convert the changes in the feature space to the problem space as each feature in the feature space is not a word that appeared in the email. As shown in the lower part of Figure 3, for this approach, we fed each email in the training set to a Doc2vec model and then inputted selected emails into the trained model to get outputted feature vector for each email with selected dimensions. We tested this approach in the gray-box attack.

B. Generating Adversarial Perturbations in Feature Space

Our approach is based on successful adversarial perturbations made to model input features. We employ the Projected Gradient Descent (PGD) method as an attack method to modify the feature values for desirable adversarial examples in the feature domain. PGD method is considered as one of the most powerful first-order adversaries [27]. PGD algorithm iterative finds the disturbance with a constraint, *dmax* that is the Euclidean distance to the original input indicating the level of perturbations, to achieve the maximum loss in classification [2]. In our approach, we run PGD over a set of spam emails with repetition and generated adversarial examples in the feature space. Then we test them to see whether they could successfully evade email classifiers by bypassing the detection. In our experiment, we used a SVM classifier for generating adversarial examples in the feature space. Therefore, the SVM classifier is considered as a white-box case in our experimental evaluation.

C. Identifying Magic Words & Crafting Adversarial Emails

Adversarial emails are crafted by adding "magic words" to the original spam emails. The "magic words" are identified by intersecting the unique ham words with the "top words". Specifically, the unique ham words are the word that only appeared in ham emails but never in spam emails. After the PGD attack, observations are made to discover which features are modified to the largest extent. We then select the "top words" whose features have been changed the most by the PGD attack. The changes are measured using a variance. In our experiments, we used the top 100 words, which is relatively efficient because the set is relatively small and demonstrates a high success rate of bypassing classifiers.

D. White-Box, Gray-Box, Black-Box Attacks

We consider three scenarios, white-box, gray-box, and black-box. White-box attacks assume that attackers have full knowledge about the spam detection machine learning model, such as its architecture, parameters, and hyperparameters. Black-box attacks assume that attackers almost know nothing about the spam detection machine learning model. At the same time, gray-box attacks assume that attackers have partial knowledge about the target spam detection model.

In the experiment, we use a local SVM classifier as an example for the white-box attack. As shown in Figure 1, a **white-box attack** has the following steps:

1) Train a local SVM classifier in the given feature space;
2) Conduct PGD attacks on the trained SVM classifier;
3) Identify "magic words" in the given dataset by observing the changes in feature space made by successful PGD attacks and unique ham words;
4) Craft adversarial emails by adding identified "magic words" into all spam emails and then recalculate their feature vector;
5) Test the trained SVM classifier with the feature vector extracted from the adversarial emails.

In addition to white-box attacks, we also evaluated the proposed attack methodology in **gray-box** and **black-box** scenarios. As shown in Figure 1, in a **black-box attack**, we evaluate the effectiveness of our attack on different classifiers trained with feature extraction methods unknown to the PGD attacks. Note that the types of classifiers are unknown to the adversarial email crafting process. However, in a **gray-box attack**, the model architecture may be known to the attacker, for example, the same SVM model in this case, but this victim SVM is trained using features generated by a different feature extraction method. Specifically, its feature space consists of Word2vec or Doc2vec vectors that have 200 dimensions to represent individual emails. We describe the implementation and evaluation details for each scenario in the following section.

IV. Evaluation

A. Datasets

Several popular spam email datasets were utilized in this study to evaluate effectiveness of and gain insights into the different attacks:

1) Ling-Spam: We used 481 spam emails and 2,412 legitimate emails from the Ling-Spam dataset (https://metatext.io/datasets/ling-spam-dataset) We extracted 46,852 features by using the TF-IDF method and 46,878 features using Word2vec and Doc2vec methods.

2) Tutorial-Spam: We used 1,897 spam emails and 4,150 ham emails from the Tutorial-Spam dataset (https://spamassassin.apache.org/old/publiccorpus/) After data extraction, the dataset contained 1,045 spam emails and 4,031 ham emails. We extracted 53,546 features using the TF-IDF method and 53,784 features using Word2vec and Doc2vec methods.

3) Enron-Spam: We used 1,500 spam emails and 3,672 ham emails from the first folder of the Enron-Spam dataset (https://www.cs.cmu.edu/∼enron/). We got 32,569 features from the TF-IDF method and 32,594 features from Word2vec and Doc2vec methods.

For the emails we used, we removed all the HTML tags, numbers, punctuation marks, and English stop words to minimize the complexity of processing. We also converted all the words to their lowercase forms and each paragraph into a single line instead of multiple lines. In the last step of data preprocessing, we conducted stemming on all the words.

Due to the complexity of the Enron-Spam dataset, we took several additional steps; we removed all the special symbols and non-Latin character sets; we also replaced all the URL links in emails by the word 'URL'.

B. Experiment Settings

1) Building SVM Classifiers: We randomly divided the emails from the dataset into a training set and a testing set with a ratio of 4:1. We trained SVM classifiers by using the SecML library [28], and used its "estimate parameters" method to find the parameters for the best classification performance. The best parameter was searched for the penalty factor with other parameters at their default settings. The SVM classifier trained using TF-IDF showed a classification accuracy of 98.92% with a 3.77% false-negative rate on average across the three datasets. The SVM classifier using Word2Vec showed a classification accuracy of 98.3% with a 4.81% false-negative rate on average across the three datasets. The SVM classifier using Doc2vec showed a classification accuracy of 98.04% with a 5.49% false-negative rate on average across the three datasets.

Since the success of crafted spam emails to bypass detection is in effect similar to false negatives being generated, we could observe the effectiveness of the attacks by comparing their success rates to the false negative rates of the trained classifiers.

2) PGD Attack for Adversarial Perturbations: For adversarial perturbation, we used the projected gradient descent (PGD) algorithm from the SecML library. With access to a trained SVM classifier, we applied the PGD method to modify 100 randomly selected spam emails from the test dataset for our framework,. The *dmax* parameter in PGD determines the success rate of flipping the classification by the SVM classifier. We examined *dmax* in a large range from 0 to 0.35 for the TF-IDF feature method and 0 to 13 for Word2vec and Doc2vec feature method, which could reach a 100% success rate over the 100 spam emails being processed, to understand its impact in experimentation. We used l2 distance for the PGD algorithm in the experiments, and eta was set to 0.01, max_iter to 20, eps to $1e-6$.

3) White-Box, Gray-Box, and Black-Box Attacks: For white-box attacks, we recalculated the feature vectors of the crafted adversarial emails after adding the magic words to attack the SVM classifier that was used to identify these words. We examined the PGD attack success rate against the SVM, the number of the "magic words", and the "magic words" attack success rate against the classifier for the attack performance.

For gray-box attacks, we kept the SVM classifier as our target model. However, we trained it using Word2vec and Doc2vec features of 200 dimensions, a common choice of using these NLP techniques. For Word2vec scenarios, we first converted each word in the selected emails to a vector of 200 dimensions using trained Word2vec models. Then, we calculated the mean vector over all the words for an email. For Doc2vec features, rather than extracting vectors for each word in an email first, we converted each email into a vector of 200 dimensions by using trained Doc2vec models to induce a vector for each email to train the SVM classifiers. Again, we crafted spam emails by adding the "magic words" found by PGD attacks, with *dmax* at 0.06 (same as in the black-box attacks), which achieved one of the best success rates in white-box attacks. As the performance measure, we examined the success rate of the crafted spam emails bypassing these new SVM classifiers.

In black-box attacks, the type of spam detection model is different from the SVM classifier in a sense that information are unknown to the attacker. For each dataset, we extracted three different sets of "magic words" through the PGD attacks trained respectively by the three types of TF-IDF, Word2vec, and Doc2vec features. We used these "magic words" to craft adversarial spam emails to attack four different classifiers (Decision Tree, Logistic Regression, MLP, and ensemble classifier (ECLF)). These classifiers were trained using TF-IDF, Word2vec with one feature for each word, Doc2vec with one feature for each word, Word2vec of 200 features for each email, Doc2vec of 200 features for each email. To train those classification models, similarly, we randomly selected 80% of the emails in each dataset as the

training set and examined the accuracy of those classifiers on the testing set. To optimize these models, we used the GridSearchCV method provided by scikit-learn [29] with the default parameters. And last, we evaluated the success rate of the crafted adversarial emails from PGD attacks bypassing these different classifiers. $verbose$ is set to 1, and n_jobs is set to -1 for GridSearchCV in our experiments.

C. Performance Metrics

We used the following several metrics in evaluation:

1) False Negative Rate: For the performance of a classifier, this is the number of false negatives, i.e., spam emails undetected, divided by the total of spam emails in the testing dataset. This measures the chance of an original spam email bypassing the classifier without being detected.

2) PGD Attack Success Rate: In white-box attacks, we randomly selected 100 spam emails from the testing set and then implemented the PGD algorithm to perturb their features. The percentage of the resulting adversarial perturbations of these 100 emails that can successfully bypass the SVM classifier is PGD Attack Success Rate.

3) Number of Magic Words: This is the size of the "magic words" set identified through the PGD attack on a SVM classifier using one of the three feature extraction methods.

4) Magic Words Attack Success Rate (white-box attacks): We crafted adversarial emails by adding the identified "magic words" into all spam emails. Then we examined the percentage of these emails that can bypass the SVM classifier being attacked by the PGD algorithm.

5) Success Rate of Adding Magic Words from PGD Attack (gray-box and black-box attacks): We added the "magic words" to all the spam emails in the testing set to evaluate their success rate of bypassing a different classifier in gray-box and black-box attacks.

These metrics are reported for white-box, gray-box, and black-box attacks respectively. PGD Attack Success Rates, Number of Magic Words, and Magic Words Attack Success Rates (white-box attacks) are reported in Figures 4, 5, and 6. False Negative Rates of the SVM classifiers used for white-box attacks are already reported in Section IV.B. False Negative Rates and Success Rates of Adding Magic Words from PGD Attack (gray-box and black-box attacks) are reported in Tables I and II.

D. Results

1) White-Box Attacks: We focused on the following metrics to characterize the susceptibility of a model: the success rate of PGD perturbation to model features, the set of "bad words" being identified, the size of the "magic words" set, and the success rate of causing the SVM classifier to misclassify a spam email by adding these "magic words".

As in Figure 4 for TF-IDF features, we varied *dmax* used in the PGD perturbation, while the number of emails used to find "magic words" was kept constant at 100. As *dmax* increases, the success rate of perturbation increases too, as expected. Significant changes made to the input features certainly can move the instance into the territory of legitimate email class. However, too much modification to features does not guarantee that more useful "magic words" are found. So the success rate of "magic words" to fool the SVM detector does not always go up. This shows a more complicated relationship between these two measures.

Overall, there is a positive correlation between the number of "magic words" and their capability to cause false negatives. When *dmax* is small, this approach tends to generate more "magic words" even though these words are not effective in inducing false negatives. Likely such small perturbations do not make changes to those features that matter to the classification. When *dmax* is big enough, successful perturbations can help to locate and modify those "important" features. Adding to a spam email enough words corresponding to these features has a good chance to flip its classification label to legitimate.

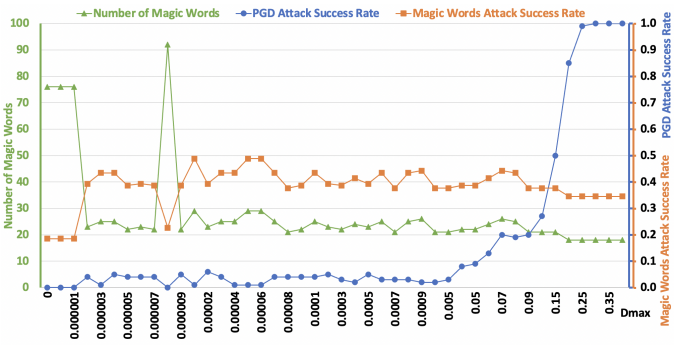

Figure 4. Success rates of the PGD attack and the "magic words" attack as well as the size of the resulting "magic words" set using TF-IDF features on the SVM classifier using the Ling-Spam dataset in white-box attacks

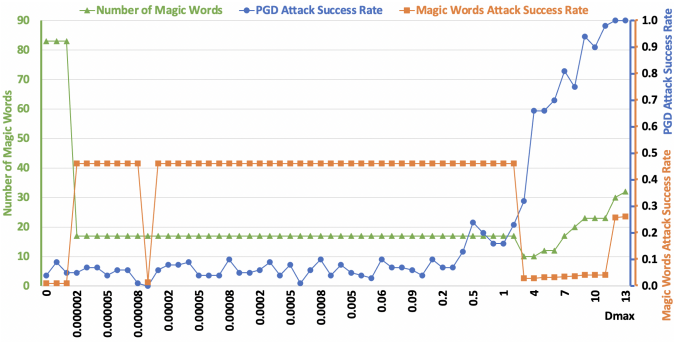

Figure 5. Success rates of the PGD attack and the "magic words" attack as well as the size of the resulting "magic words" set using Word2vec features on the SVM classifier using the Ling-Spam dataset in white-box attacks

The observations discussed in the above are also present in Figures 5 and 6 for Word2vec and Doc2vec features. They are informative of the complex dynamics of such an attack.

Table I
RESULT OF GRAY-BOX ATTACKS AGAINST SVM CLASSIFIERS USING DIFFERENT FEATURES

Dataset	Classifier Feature (Dimentionality)	Classifier's Accuracy	False Negative Rate	Success Rate of Magic Words from PGD Attack		
				TF-IDF	Word2vec	Doc2vec
Ling-Spam	Word2vec (200)	98.96%	0%	24.21%	13.68%	16.84%
	Doc2vec (200)	96.20%	1.05%	2.11%	3.16%	4.21%
Tutorial-Spam	Word2vec (200)	96.52%	9.65%	19.69%	25.48%	17.37%
	Doc2vec (200)	94.23%	13.07%	13.46%	13.85%	13.85%
Enron-Spam	Word2vec (200)	93.91%	18.95%	94.12%	95.10%	92.48%
	Doc2vec (200)	91.21%	15.38%	64.88%	75.58%	75.58%

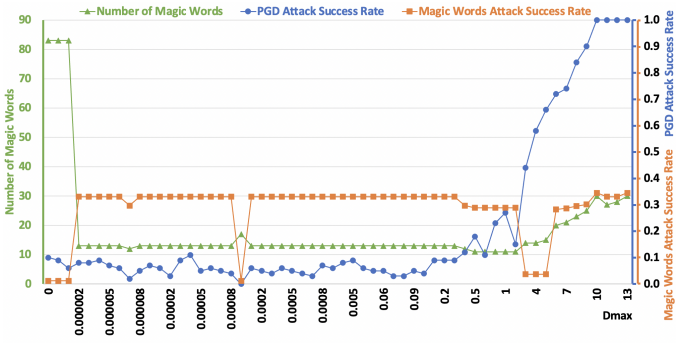

Figure 6. Success rates of the PGD attack and the "magic words" attack as well as the size of the resulting "magic words" set using Doc2vec features on the SVM classifier using the Ling-Spam dataset in white-box attacks

From an attacker's perspective, there is a sweet point of adding a set of "magic words" into the spam emails that can best balance the cost and gain in attacks. The attack certainly wants to achieve a good chance to succeed, in terms of a satisfying success rate of using a set of "magic words". At the same time, the more "magic words" are added, the more effort it requires to hide or disguise these words. An attacker would prefer using fewer "magic words" so that those crafted spam emails would not be easily identified as spam by readers after bypassing the spam filter. Attackers will need to seek a trade-off between the number of "magic words" and the success rate of bypassing by selecting an appropriate *dmax* setting.

2) Gray-Box Attacks: Table I shows the accuracy and the false-negative rate of the classifiers, as well as the success rate of adversarial emails bypassing these classifiers. In comparison, it can be observed that the attack success rates in gray-box attacks are significantly lower than those in the white-box experiment, e.g., for the Ling-Spam dataset. Moreover, the attack success rates on models trained with Word2vec are higher than those against models trained with Doc2vec vectors. This is interesting, especially since the models trained with Doc2vec have a lower accuracy. This seems to suggest that the Doc2vec method is more resistant to such attacks. Classifiers trained with Doc2vec are less sensitive but more generalizable with a "more coarse" decision boundary, and they are more robust in the presence of adversarial changes.

3) Black-Box Attacks: The results are shown in Table II. Based on the results, the decision tree classifier has the weakest resistance to the attacks (12 times the weakest in 15 feature spaces). This could be because the decision tree takes a step-wise approach in using the features. Compared to logistic regression, the decision is impacted more significantly by a subset of features. Therefore, adversarial attacks on a decision tree model can be "more effective" by focusing on these significant features to navigate in the feature space. It is worth noticing that the multi-layer perceptron classifiers are also very susceptible to the attack even though they have the highest accuracy and the lowest false-negative rate in all feature spaces. Another interesting observation is that the attack success rate against the ensemble classifier is always in the middle, which indicates that the ensemble classifier could potentially be used as a robust measure against the attacks to prevent the worst-case scenario.

Attack transferability is demonstrated among these models in the results. Adding a set of "magic words" which are identified from using TF-IDF, Word2vec, or Doc2vec methods in PGD attacks can increase the chance of adversarial emails bypassing a spam filter regardless of which type of feature it is using. In other words, the success rate of such attacks is higher than the false-negative rate of the classifier. It is worth noticing that there are special cases where the chance of bypassing the filter is not increased, or even decreased, which happened in the Tutorial-Spam and Doc2vec(200) scenario. The possible explanation for this decrease is Doc2vec(200) was not used as the features in the white-box attack so that using the "magic words" can make those adversarial emails stand out as abnormal cases.

There are some other trends in the black-box attack results. First, the success rates of magic word attacks seem to be dependent on the dataset. For example, all the attacks are very successful for the Enron-Spam dataset, with higher than 60% success rates; while all the success rates of attacks are lower than 20% for the Ling-Spam dataset. Second, the "magic words" extracted from TF-IDF are more effective when the

Table II
RESULT OF BLACK-BOX ATTACKS AGAINST DIFFERENT CLASSIFIERS

Dataset	Classifier Feature (Dimentionality)	Classifier Type	Classifier's Accuracy	False Negative rate	Success Rate of Magic Words from PGD Attack		
					TF-IDF	Word2vec	Doc2vec
Ling-Spam	TF-IDF	Decision Tree	95.68%	13.68%	16.84%	15.79%	14.74%
		Logistic Regression	99.31%	2.11%	37.89%	17.89%	21.05%
		MLP	99.48%	1.05%	32.63%	17.89%	12.63%
		ECLF	99.48%	1.05%	33.68%	17.89%	15.79%
	Word2vec(single)	Decision Tree	96.03%	10.53%	73.68%	73.68%	73.68%
		Logistic Regression	99.65%	2.11%	20.00%	16.84%	17.89%
		MLP	99.83%	1.05%	42.11%	49.47%	17.89%
		ECLF	99.65%	2.11%	32.63%	17.89%	16.84%
	Doc2vec(single)	Decision Tree	94.99%	13.68%	100.00%	100.00%	100.00%
		Logistic Regression	99.31%	0%	16.84%	13.68%	14.74%
		MLP	98.62%	0%	12.63%	16.84%	17.89%
		ECLF	99.31%	0%	16.84%	16.84%	17.89%
	Word2vec(200)	Decision Tree	97.58%	2.11%	37.89%	33.68%	32.63%
		Logistic Regression	98.79%	1.05%	32.63%	25.26%	25.26%
		MLP	98.79%	2.11%	28.42%	13.68%	22.11%
		ECLF	98.96%	1.05%	30.53%	15.79%	22.11%
	Doc2vec(200)	Decision Tree	95.85%	12.63%	34.74%	34.74%	36.84%
		Logistic Regression	96.72%	1.05%	4.21%	7.37%	3.16%
		MLP	99.31%	1.05%	14.74%	16.84%	15.79%
		ECLF	99.31%	1.05%	14.74%	16.84%	14.74%
Tutorial-Spam	TF-IDF	Decision Tree	93.92%	15.00%	17.31%	17.31%	17.69%
		Logistic Regression	98.26%	6.92%	8.85%	8.85%	7.69%
		MLP	98.89%	3.85%	9.23%	8.46%	6.15%
		ECLF	98.34%	2.69%	8.46%	8.46%	7.31%
	Word2vec(single)	Decision Tree	93.06%	14.34%	16.67%	16.67%	16.67%
		Logistic Regression	97.79%	9.69%	10.85%	10.85%	10.47%
		MLP	98.26%	2.33%	3.88%	2.33%	2,33%
		ECLF	98.50%	5.04%	5.81%	5.81%	5.81%
	Doc2vec(single)	Decision Tree	94.63%	12.69%	13.46%	13.46%	13.46%
		Logistic Regression	98.26%	5.77%	5.77%	5.77%	5.38%
		MLP	98.74%	2.69%	4.23%	2.69%	2.69%
		ECLF	98.42%	3.85%	4.62%	3.85%	3.85%
	Word2vec(200)	Decision Tree	93.68%	16.98%	20.85%	26.64%	23.55%
		Logistic Regression	95.49%	11.96%	15.06%	16.60%	15.83%
		MLP	97.55%	7.33%	10.42%	13.13%	12.36%
		ECLF	96.68%	6.56%	8.11%	11.20%	10.42%
	Doc2vec(200)	Decision Tree	91.00%	23.07%	18.08%	23.46%	22.31%
		Logistic Regression	93.13%	16.92%	17.31%	18.46%	18.08%
		MLP	97.31%	7.69%	6.92%	12.69%	12.69%
		ECLF	97.08%	9.23%	7.31%	14.23%	13.85%
Enron-Spam	TF-IDF	Decision Tree	94.01%	11.11%	100.00%	100.00%	100.00%
		Logistic Regression	98.84%	2.29%	83.66%	82.03%	77.78%
		MLP	98.65%	2.61%	70.92%	72.55%	73.53%
		ECLF	98.84%	2.29%	83.66%	82.03%	77.78%
	Word2vec(single)	Decision Tree	95.17%	6.35%	100.00%	100.00%	100.00%
		Logistic Regression	98.07%	1.00%	91.64%	97.32%	88.63%
		MLP	98.16%	1.00%	86.96%	95.65%	95.65%
		ECLF	98.55%	0.33%	93.31%	98.66%	95.65%
	Doc2vec(single)	Decision Tree	94.98%	5.56%	100.00%	100.00%	100.00%
		Logistic Regression	97.78%	5.88%	94.12%	96.41%	96.41%
		MLP	98.36%	2.29%	89.54%	93.46%	94.77%
		ECLF	98.74%	1.63%	94.44%	96.73%	97.39%
	Word2vec(200)	Decision Tree	91.11%	24.51%	80.72%	85.62%	81.70%
		Logistic Regression	94.30%	11.76%	79.74%	83.66%	80.39%
		MLP	94.69%	16.67%	81.37%	82.68%	77.45%
		ECLF	94.98%	14.71%	81.70%	84.64%	80.39%
	Doc2vec(200)	Decision Tree	88.41%	19.06%	42.47%	66.55%	66.22%
		Logistic Regression	89.76%	16.72%	64.21%	71.57%	76.58%
		MLP	94.69%	8.03%	69.89%	74.24%	76.25%
		ECLF	93.82%	9.70%	61.87%	73.24%	76.58%

false-negative rate of a classifier is relatively low, with high sensitivity; the "magic words" extracted from Word2vec and Doc2vec are more effective when the false-negative rate of a classifier is relatively high, being more robust. So overall, the SVM classifiers trained with Word2vec and Doc2vec features have the potential to generate more sophisticated attacks with the "magic words" identified through PGD attacks.

E. Additional Discussion on Defense

Some preliminary results suggested that feature selection can be deployed as a defense mechanism against these attacks. In one experiment on decision tree classifiers, the ratio of selected features to original features was about 1:10. The resulting decision tree classifiers showed a significant decrease in the success rate of using these magic words. In addition, reinforced model learning can be considered as another defense measure. When "magic words" are identified, we can use such information to retrain a classifier so it can become more resistant to adversarial emails utilizing these words.

V. Conclusion and Future work

This paper explored a new method of identifying "magic words" using various feature extraction methods and investigated the effectiveness of crafted adversarial emails injected with these "magic words" to fool spam filters on three datasets. We generated different sets of "magic words" based on the PGD attack on SVM classifiers using TF-IDF, Word2vec, and Doc2vec as feature extraction methods. We showed the effectiveness of such an attack on various classification models including SVM, decision tree, logistic regression, MLP, and ensemble classifiers.

Future research is certainly needed to make this line of study more comprehensive. Additional experimentation with other feature extraction methods and more datasets will gain more insights into the effect of using different feature extraction methods. Moreover, as briefly discussed, employing advanced defense mechanisms is essential to understanding how persistent such attacks can be.

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[2] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[3] C. Wang, D. Zhang, S. Huang, X. Li, and L. Ding, "Crafting adversarial email content against machine learning based spam email detection," in *Proceedings of the 2021 International Symposium on Advanced Security on Software and Systems*, 2021, pp. 23–28.

[4] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[6] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*. PMLR, 2014, pp. 1188–1196.

[7] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[8] S. O. Olatunji, "Extreme learning machines and support vector machines models for email spam detection," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2017, pp. 1–6.

[9] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, "An evaluation of naive bayesian anti-spam filtering," *arXiv preprint cs/0006013*, 2000.

[10] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A bayesian approach to filtering junk e-mail," in *AAAI Workshop on Learning for Text Categorization*, vol. 62. Madison, Wisconsin, 1998, pp. 98–105.

[11] X. Carreras and L. Marquez, "Boosting trees for anti-spam email filtering," *arXiv preprint cs/0109015*, 2001.

[12] T. Hastie and R. Tibshirani, "& friedman, j.(2008). the elements of statistical learning; data mining, inference and prediction," 2009.

[13] T. Strauss, M. Hanselmann, A. Junginger, and H. Ulmer, "Ensemble methods as a defense to adversarial perturbations against deep neural networks," *arXiv preprint arXiv:1709.03423*, 2017.

[14] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[15] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," pp. 1–7, 2018.

[16] H. Yakura and J. Sakuma, "Robust audio adversarial example for a physical attack," *arXiv preprint arXiv:1810.11793*, 2018.

[17] Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5231–5240.

[18] H. Guo, Z. Wang, B. Wang, X. Li, and D. M. Shila, "Fooling a deep-learning based gait behavioral biometric system," in *2020 IEEE Security and Privacy Workshops (SPW)*, 2020, pp. 221–227.

[19] D. Han, Z. Wang, Y. Zhong, W. Chen, J. Yang, S. Lu, X. Shi, and X. Yin, "Practical traffic-space adversarial attacks on learning-based nidss," *CoRR*, vol. abs/2005.07519, 2020. [Online]. Available: https://arxiv.org/abs/2005.07519

[20] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on gan," 2017.

[21] J. Y. Yoo, J. X. Morris, E. Lifland, and Y. Qi, "Searching for a search method: Benchmarking search algorithms for generating nlp adversarial examples," 2020.

[22] G. L. Wittel and S. F. Wu, "On attacking statistical spam filters." in *CEAS*. Citeseer, 2004.

[23] D. Lowd and C. Meek, "Good word attacks on statistical spam filters." in *CEAS*, vol. 2005, 2005.

[24] Z. Jorgensen, Y. Zhou, and M. Inge, "A multiple instance learning strategy for combating good word attacks on spam filters." *Journal of Machine Learning Research*, vol. 9, no. 6, 2008.

[25] B. Kuchipudi, R. T. Nannapaneni, and Q. Liao, "Adversarial machine learning for spam filters," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020, pp. 1–6.

[26] C. Wang, D. Zhang, S. Huang, X. Li, and L. Ding, "Crafting adversarial email content against machine learning based spam email detection," in *2021 International Symposium on Advanced Security on Software and Systems (ASSS '21)*. ACM, 2021.

[27] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International conference on machine learning*. PMLR, 2018, pp. 274–283.

[28] M. Melis, A. Demontis, M. Pintor, A. Sotgiu, and B. Biggio, "secml: A python library for secure and explainable machine learning," *arXiv preprint arXiv:1912.10013*, 2019.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.