Platform-Oblivious Anti-Spam Gateway

Yihe Zhang
yihe.zhang1@louisiana.edu
University of Louisiana at Lafayette
Lafayette, LA, USA

Xu Yuan*
xu.yuan@louisiana.edu
University of Louisiana at Lafayette
Lafayette, LA, USA

Nian-Feng Tzeng
tzeng@louisiana.edu
University of Louisiana at Lafayette
Lafayette, LA, USA

ABSTRACT

This paper addresses a novel anti-spam gateway targeting multiple linguistic-based social platforms to expose the outlier property of their spam messages uniformly for effective detection. Instead of labeling ground truth datasets and extracting key features, which are labor-intensive and time-consuming, we start with coarsely mining seed corpora of spams and hams from the target data (aiming for spam classification), before reconstructing them as the reference. To catch each word's rich information in the semantic and syntactic perspectives, we then leverage the natural language processing (NLP) model to embed each word into the high-dimensional vector space and use a neural network to train a spam word model. After that, each message is encoded by using the predicted spam scores from this model for all included stem words. The encoded messages are processed by the prominent outlier techniques to produce their respective scores, allowing us to rank them for making the outlier visible. Our solution is unsupervised, without relying on specifics of any platform or dataset, to be platform-oblivious. Through extensive experiments, our solution is demonstrated to expose spammers' outlier characteristics effectively, outperform all examined unsupervised methods in almost all metrics, and may even better supervised counterparts.

CCS CONCEPTS

• **Security and privacy → Web application security**; *Network security*; *Intrusion/anomaly detection and malware mitigation.*

KEYWORDS

Anti-Spam; Unsupervised; Outlier Detection

ACM Reference Format:
Yihe Zhang, Xu Yuan, and Nian-Feng Tzeng. 2021. Platform-Oblivious Anti-Spam Gateway. In *Annual Computer Security Applications Conference (ACSAC '21), December 6–10, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3485832.3488024

1 INTRODUCTION

Nowadays, the ever-growing use of social platforms (e.g., social networks, emails, and others) brings great convenience to our daily life, leading to our high reliance on them for communications, conversations, or discussion. At the same time, it also pervasively attracts spammers' interests to spread spam messages or information, that pollute the social platforms purposefully. Existing anti-spam mechanisms have filtered the majority of spam messages, leaving out only a small portion. Nonetheless, spam messages that escape from anti-spam mechanisms are still plentiful, and they continuously cause gross detriments to the normal users. It remains challenging to detect and remove them for mitigating the cyberspace risks and sanitizing social environments. Given the fact that spammers inevitably exhibit behavioral patterns which differ considerably from those of normal users, such a disparity never disappears once spammy behaviors are undertaken. Furthermore, skillful spammers may keep evolving to imitate normal users by concealing their behaviors for abnormality reduction. As a result, it is imperative to design an intelligent spam detection system for mining the latent patterns and use them for classifying spammers automatically.

To date, various supervised learning methods have been proposed for spam detection. By extracting effective features and relying on labeled ground-truth datasets, the machine learning classifiers learn the latent disparity inherent to spam and ham (i.e., non-spam) messages. Extensive work has undertaken on extracting various features, including user profiles [13], behaviors [23], message contents [32, 54], user relationships [9, 49, 62], among others. However, extracting key features has been widely recognized as a challenging problem. In addition, all those features are tailored only to a specific dataset or platform, with considerable effort involved in deriving new customized features for every individual platform. Furthermore, reliable large-sized ground-truth datasets are necessary for supervised learning, but they represent another challenge. So far, there is no effective method for labeling a large-sized dataset reliably. Moreover, the supervised methods may perform poorly when applied to the real social networks data, where the spam and ham messages are highly uneven, as revealed by prominent research [29, 44, 52, 55]. Meanwhile, semi-supervised methods [11, 18, 24] have been proposed to mitigate their reliance on the ground-truth datasets. But, suitably sized ground-truth datasets are still required, and one proposed solution generally performs unsatisfactorily when applied to other platforms. Although diverse unsupervised learning methods were pursued [28, 35, 50, 51] for freeness from labeling datasets, they usually exhibit mediocre performance.

To address the aforementioned concerns on feature extraction, ground-truth labeling, cross-platform deployment, and unsatisfactory performance, we aim to develop an effective platform-oblivious framework for unsupervised spam detection. According to the fact that spam messages account for a small portion of the total data volume and that their patterns are fundamentally disparate in comparison to those from normal users, they can be easily detected and removed, if exposed as the outliers of whole data volume. The prominent outlier detection method proposed in [7] is effective in mining the outlier property of a given dataset in three dimensions, i.e., *Shapes, Magnitude,* and *Amplitude*. However, not all spammers' outlier characteristics are apparent, calling for an appropriate encoding method to make spam messages visible in the three dimensions.

This paper introduces a novel spam detection approach by exposing the outlier property of spams, deriving a platform-oblivious framework to work in an unsupervised manner. Our approach is

*Corresponding author

unified and readily deployable to multiple linguistic-based social platforms without involving extra effort individually. It relies on the spam and ham words automatically mined from the target dataset (aiming for spam classification), instead of the previously labeled ground-truth dataset, to serve as the seed corpora of spams and hams, respectively. A new method based on mining messages' structure is proposed for automatically identifying the sets of spam seed and ham seed corpora from the target dataset. We further refine them with a series of techniques and then reconstruct them. Here, the Gibbs sampling [36] is used to help us iteratively pre-label the spam and ham datasets while sampling each word's distribution in the respective datasets. Such sampled word distributions are used to calculate the spam scores of all words for reconstructing both new spam and ham corpora. Furthermore, we employ the NLP model to embed each word into the high-dimensional vector space so that it can cover a rich set of words with the similar meanings or structures. The neural network is then employed to learn the high-dimensional representations of words and their associated scores to train a spam word model, serving to predict other words' spam scores in the target dataset.

Meanwhile, we represent each message in the target dataset by a list of stem words, which are then inputted into the spam word model to predict the spam scores, being used to encode this message. In the end, *Magnitude outlier* approach stated in [7] is employed with the input of encoded messages to calculate their outlier scores for ranking them from the highest values to the lowest to exhibit spams' outlier property.

The contributions of our work are summarized as follows.

- We develop a novel anti-spam gateway that can clearly expose the outlier property of spam messages in the target datasets. Our system is unsupervised without relying on feature extraction or ground-truth labeling, able to significantly relieve the detection workload. Instead, our system utilizes the target dataset to acquire the spam words, potentially overcoming the spam feature drift problem to some extent.
- Our solution is unified for manifold linguistic-based social platforms since it does not rely on any prior knowledge (e.g., features and training data) to mine linguistic information and patterns. Thus, it is not tailored to any specific platform and can be deployed and integrated into multiple platforms to automatically conduct spam detection task, without additional effort individually. The experiments confirm that our system works efficiently on the datasets from short message service (SMS), Email, and Twitter, exposing spams as the outliers of the respective dataset.
- We propose a novel technique to estimate both the spam ratio in a dataset and the threshold value used for detecting spams via the visualized outlier curve. Such a technique is important, acting as a complement to the prior outlier technique, for automatically separating spam and ham regions.
- We implement our system and run it on four linguistic-based datasets from various platforms. Experimental results show that our system not only outperforms all examined unsupervised solutions in almost all performance metrics under the four datasets, but also may surpass its supervised counterparts, while avoiding ground-truth labeling costs.

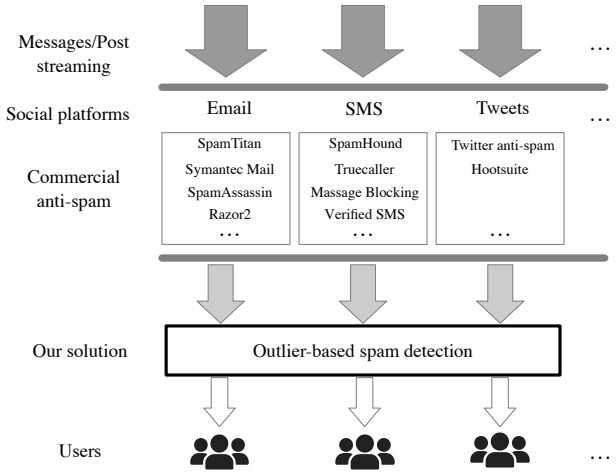


Figure 1: Illustration of our proposed solution in target problem space.

As an unsupervised spam detection solution, our proposed approach is readily applicable to datasets collected from a given social platform at different time points without any labeling nor model parameter rectification effort at all, for time-invariant portability with equally high accuracy. For example, our solution was applied both to the set of 2, 094, 889 tweets collected over two days in November 2020, and to Twitter Normal Dataset [3] (with 5, 823, 230 tweets collected in 2019, as detailed in Section 4.1). It reported 161,935 tweets as spams and 1, 932, 954 as ham ones for the 2020 tweet dataset, giving rise to the Precision and Recall being 86.3% and 82.4%, respectively. The results are close to those obtained by our solution on the 2019 Twitter Normal Dataset (which yielded the Precision of 85.1% and the Recall of 78.4% as shown in Table 4). Hence, the proposed solution confirms the clear advantage of an unsupervised approach, able to be platform-oblivious and time-invariant. For the rest of this paper, we focus on the platform-oblivious perspective of our solution, knowing that its time-invariant feature holds equally.

2 PROBLEM STATEMENT

This paper studies the spam detection problem in social platforms where users post and/or interact via linguistic information (i.e., messages) for discussion or conversation. By realizing that the spammy behaviors of a spammer will inevitably expose some natures that are disparate when compared to normal users' patterns, we aim to capture such an inherent nature and develop a novel platform-oblivious solution for its detection. Our proposed solution is to act as a complementary component to the commercial anti-spam mechanisms (as shown in Figure 1), to further filter out the residual spam messages that escape from existing commercial anti-spam mechanisms.

2.1 Motivation

Nowadays, commercial anti-spam mechanisms have been widely deployed in social platforms, which can block the majority of spam messages [41, 45]. However, there remains a small portion and yet

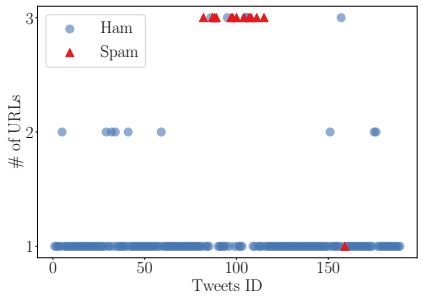


Figure 2: Spammers tend to employ more URLs than normal users, causing them to stand out.

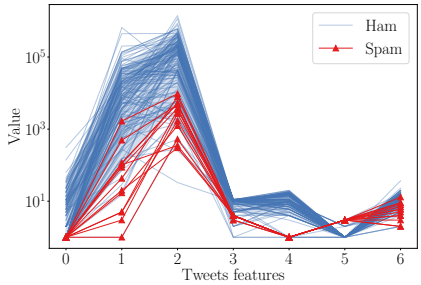


Figure 3: Spam outliers, where an outlier feature value refers to a substantially larger or smaller value than the mean, and where 0-6 on the X-axis are 7 features extracted from each tweet.

unignorable amounts that can escape from these mechanisms, pervasively polluting the social environments and causing cyberspace risks. We observe that although spammers imitate the normal users to conceal their spammy activities, their special behaviors inevitably reveal inherent disparities with respect to the majority of normal dataset, which can exhibit outlier characteristics detectable through deep analyses.

Our empirical feat has unveiled that the outlier nature indeed exists when analyzing some labeled ground truth datasets. For example, [3] outlines a set of labeled tweets collected from Twitter networks. We randomly select 189 tweets and analyze the number of URLs included in each tweet. Figure 2 shows the numbers of URLs involved in the spam and ham tweets, with red and blue points denoting URL counts of the former and the latter, respectively. Apparently, spammers tend to use more URLs than normal users since an URL realizes redirection toward a certain malicious website. This makes the spam tweets stand out, thus exhibiting the outlier property. It should be noted that a disguise spammer may try to hide its tweets' outlier property by lowering the URL counts contained therein, if the spammer knows the specific feature being targeted for outlier detection. However, our solution employs a wide range of features from both semantic and syntactic perspectives for outlier detection and those features are not static, able to evolve over time, as stated next.

The outlier property is evidenced by other features as well. Figure 3 shows different values of various features for spam and ham tweets, with a total of 7 features (along the x-axis) considered, i.e., number of lists, number of favorites, number of statuses, number of hashtags, username length, number of URLs, and user screen name length. In this figure, red lines and blue lines indicate respectively the spam and the ham tweets. It is observed that most spammers exhibit themselves as outliers to some extent, with respect to the values of these 7 features.

Figures 2 and 3 roughly exhibit a certain level of outlier property for spams' patterns when taking the proper features into account. These results demonstrate that spam messages indeed have the outlier property, thus motivating our development of a new spam detection framework by exploring such property. Meanwhile, we also observe some spam tweets still hide within the majority of ham tweets. The reason is possibly due to the missing of some key features or some spammers use sophisticated techniques to disguise themselves, making their latent patterns inapparent. This signifies that just selecting features to encode messages are inadequate. It is necessary to develop a new encoding method to facilitate dataset processing so that spam messages in the entire dataset can be exposed as outliers. In this work, we aim to have a deep investigation into mining the disparate patterns between spam and ham messages, for exposing the outlier characteristics of spam messages in order to make them detectable. Our solution is expected to be platform-oblivious, applicable to multiple linguistic-based social platforms without rectification.

3 PLATFORM-OBLIVIOUS FRAMEWORK

In this section, we develop a platform-oblivious unsupervised system that can be generally applied to different social platforms for effectively singling out spam messages. Our system consists of the following core components (shown in Figure 4): i) *Rough Seed collection (Section 3.1)*, ii) *Reconstructing words corpora (Section 3.2)*, iii) *Training spam word model (Section 3.3)*, and iv) *Outlier detection (Section 3.4)*. Given any target dataset aiming for spam detection, Section 3.1 proposes two measures to capture spam and ham's unique structures, respectively, in the target dataset, for helping to roughly collect the spam and ham seed corpora. Section 3.2 narrates the procedure for reconstructing spam and ham corpora by leveraging the Gibbs sampling method [36] with our customized design. Section 3.3 first details that each word in the reconstructed corpora is embedded into the high dimensional vector space so as to reflect its richer syntactic and semantic patterns, thereby able to help cover a set of words that have the similar structure or meaning. The neural network is then employed to train a spam word model with the input of word vectors and the output of associated scores. In Section 3.4, we use the trained spam model to predict the spam scores of all stem words included in each message from the target dataset and use them to encode this message. Then, the *Magnitude outlier* approach [7] is employed by taking the encoded messages as input to expose the outlier property of spam messages.

3.1 Seed Collection

Without the labeled ground truth for training, some patterns showing the disparity between spams and hams have to be identified first as the reference for distinguishing them in the design of our unsupervised solution. Since spammers typically use certain specific

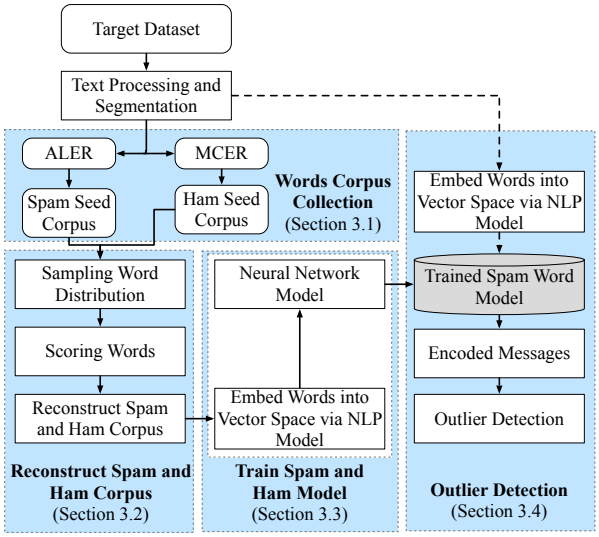


Figure 4: Flowchart of our platform-oblivious spam detection system.

syntaxes and semantics for the spammy purpose, their messages are constructed to have significant differences from the ham ones. This inspires us to identify from the target dataset, two sets of words that are highly likely to be used by spam and ham messages, respectively, referring them as the spam seed and ham seed corpora. However, deriving seed corpora is challenging, as for a given word, it is difficult to surely claim its polarity since it may have different semantic meanings in various sentence structures. Inspired by the classical directed acyclic word graphs language model often adopted to detect unique structures of strings and substrings[30], we propose two new methods, named as Average Longest Equivalent Relation (ALER) and Mass Class Equivalent Relation (MCER), for calculating the average length and the number, of different unique patterns in a message, respectively. Our spam and ham corpora are constructed according to ALER and MCER measures of all messages.

The initial step of our method is to pre-process the dataset in two steps. (1) Word Cleaning: by removing auxiliary characters and information from the dataset, such as the email address, URL links, @ or hashtag information, etc. They are the regular format or words that are commonly used in the social platforms but unhelpful to mine spammers' patterns, thus subject to removal safely. Then, we remove the stop words that have little (or no) semantic meaning by using the NLTK package in Python. (2) Stemming: by employing the stemming process to further reduce inflected (or derived) words to their stems, e.g., "take" and "took". This step aims to use a unified stem word to capture the corresponding spam or non-spam pattern.

With such two-step pre-processing, each message m_i in a target dataset $\mathbb{D}_t$ can be represented with a list of stem words, denoted as $\mathbb{S}_i = \{w_1, w_2, \cdots\}$. For each $\mathbb{S}_i$, we discover all its N-grams (with N ranging from 1 to 10, for simplicity, knowing that the range can vary for different datasets) and put them into the set of $Sub(\mathbb{S}_i)$. We express all unique grams as $Sub_T = Sub(\mathbb{S}_1) \cup Sub(\mathbb{S}_2), \cdots, Sub(\mathbb{S}_K)$, where K is the number of messages. For each gram $s \in Sub_T$, if s appears in $\mathbb{S}_k$, the corresponding index k is recorded in $Occ(s)$, i.e.,

$$Occ(s) = \{k | \text{if } s \in Sub(\mathbb{S}_k) \text{ for } 1 \le k \le K\} . \quad (1)$$

Definition 1. *For any two grams s_1 and s_2, their equivalent relation $\equiv$ is defined as: $s_1 \equiv s_2 \Leftrightarrow Occ(s_1) = Occ(s_2)$.*

Let $[s_1]_{Occ}$ and $[s_2]_{Occ}$ denote the equivalent classes of s_1 and s_2, respectively. If $s_1 \equiv s_2$, we have $s_1 \in [s_2]_{Occ}$, $s_2 \in [s_1]_{Occ}$, and $[s_1]_{Occ} = [s_2]_{Occ}$.

Definition 2. *Assume $s_1, s_2, \cdots, s_n$ are grams in Sub_T and they have the same equivalent relation to s, i.e., $s_1 \equiv s_2 \cdots \equiv s_n$. The longest equivalent relation $LER(s)$ is defined as:*

$$LER(s) = \max\{|s_1|_w, \cdots, |s_n|_w\}, \text{ for } s_1, \cdots, s_n \in [s]_{Occ},$$

where $|\cdot|_w$ denotes the length of the gram.

The rationale of counting *LER* comprises three key points. First, the *LER* measure of a gram s takes into account all other ones that have the equivalent relation, and that may share the same syntactic/semantic patterns. Second, the higher value of *LER* indicates the more unique characteristic of a gram. Third, if a message includes a commonly used syntactic/semantic pattern, it signifies more equivalent classes.

Given the fact that a spam typically uses certain uncommon syntactic and semantic patterns, we define the ALER measure, which will be used last to filter out spam seeds. ALER of $\mathbb{S}_i$ is calculated by averaging over all LER measures of grams in $Sub(\mathbb{S}_i)$, i.e.,

$$ALER(\mathbb{S}_i) = \frac{1}{|Sub(\mathbb{S}_i)|} \sum_{s \in Sub(\mathbb{S}_i)} LER(s), \quad (2)$$

where $|Sub(\mathbb{S}_i)|$ denotes the total number of grams in $Sub(\mathbb{S}_i)$. Meanwhile, the MCER measure is defined to count the number of different equivalent classes corresponding to a message, used last for filtering the ham seed, i.e.,

$$MCER(\mathbb{S}_i) = \#\{[s]_{Occ} | s \in Sub(\mathbb{S}_i)\}, \quad (3)$$

where # counts the number of unique equivalent classes. To better understand the ALER and MCER measures, we use a toy example to show their calculation procedure, as follows:

Example 3.1 (A Toy Example). *We use three messages shown in Figure 5(a) as an example. After two-step pre-processing,* i.e., *word cleaning and stemming, each message can be represented by a set of stem words shown in Figure 5(b). Then, the N-grams of each message can be found (see Figure 5(c)). We calculate the Occ according to Eqn. (1) and derive the equivalent relations shown in Figure 5(d). For each equivalent class (corresponding to each row in Figure 5(d)), we calculate the respective LER value as shown in Figure 5(e). Specifically, LER for the equivalent class $['urgent']_{Occ}$ is* 5, *since the longest gram in the same class (i.e., 'urgent grandson arrest night mexico') contains five words. Next, we can calculate the ALERs based on Eqn. (2). That is, in message 1, there are total* 15 *grams, so $ALER(\mathbb{S}_1) = 1/15 * (5 * 15) = 5$. In message 2, there are* 10 *grams, then $ALER(\mathbb{S}_2) = 1/10 * (4 * 7 + 2 * 3) = 3.4$. In message 3, there are 6 grams, then $ALER(\mathbb{S}_3) = 1/6 * (3 * 3 + 2 * 3) = 2.5$. We then calculate MCER based on Eqn. (3), with results shown in Figure 5(f). For example, in message 1, there is only one unique equivalent class $['urgent']_{Occ}$, so $MCER(\mathbb{S}_1) = 1$. For message 2, two unique equivalent classes $['see']_{Occ}$ and $['major']_{Occ}$ exist, so $MCER(\mathbb{S}_2) = 2$. For message 3,*

Urgent your grandson was arrested last night in Mexico. **(1)**	{urgent, grandson, arrest, night, mexico}
I will see the major person that can guide me. **(2)**	{see, major, person, guide}
You are the major person in my life. **(3)**	{major, person, life}

(a) Original messages (b) Two-step pre-processing results

Index	N-grams	Count
(1)	{'urgent', 'grandson', 'arrest', 'night', 'mexico', 'urgent grandson', 'grandson arrest', 'arrest night', 'hight mexico','urgent grandson arrest', 'grandson arrest night', 'arrest night mexico', 'urgent grandson arrest night', 'grandson arrest gift mexico', 'urgent grandson arrest hight mexico'}	15
(2)	{'see', 'major', 'person', 'guide', 'see major', 'major person', 'person guide', 'see major person', 'major person guide','see major person guide'}	10
(3)	{'major', 'person', 'life', 'major person', 'person life', 'major person life'}	6

(c) N-grams of each message

N-grams	Co-occurrence Index
{'urgent', 'grandson', 'arrest', 'night', 'mexico', 'urgent grandson', 'grandson arrest', 'arrest night', 'hight mexico','urgent grandson arrest', 'grandson arrest night', 'arrest night mexico', 'urgent grandson arrest night', 'grandson arrest gift mexico', 'urgent grandson arrest hight mexico'}	{1}
{'see', 'guide', 'see major', 'person guide', 'see major person', 'major person guide','see major person guide'}	{2}
{'life', 'person life', 'major person life'}	{3}
{'major', 'person', 'major person'}	{2, 3}

(d) Co-occurrence of N-grams

Equivalent Class	Longest member in the same class	LER
['urgent']$_{Occ}$	['urgent grandson arrest hight mexico']$_{Occ}$	5
['see']$_{Occ}$	['see major person guide']$_{Occ}$	4
['life']$_{Occ}$	['major person life']$_{Occ}$	3
['major']$_{Occ}$	['major person']$_{Occ}$	2

(e) Calculating LER

Message	Equivalent Class	MCER
Urgent your grandson was arrested last night in Mexico.	['urgent']$_{Occ}$	1
I will see the major person that can guide me.	['see']$_{Occ}$ ['major']$_{Occ}$	2
You are the major person in my life.	['life']$_{Occ}$ ['major']$_{Occ}$	2

(f) $MCER$ values

Figure 5: An example of calculating ALER and MCER.

there are two unique equivalent classes of ['life']$_{Occ}$ and ['major']$_{Occ}$, so $MCER(\mathbb{S}_3) = 2$.

We scan the spam and ham seeds that lie in the top 1% measures of ALER and MCER, respectively. More specifically, the stem words in the messages which belong to top 1% ALER measure but fail to appear in the messages with top 1% MCER measure, will be considered as the spam seed. Similarly, the stem words in the messages which belong to top 1% MCER measure but fail to appear in the messages with top 1% ALER measure, will be used as the ham seed. Notably, the threshold 1% is selected based on our empirical study. We have conducted experiments on 3000 randomly selected email messages from the Metsis email dataset [26] for verification. Figure 6 shows the ALER and MCER measures of all messages. It

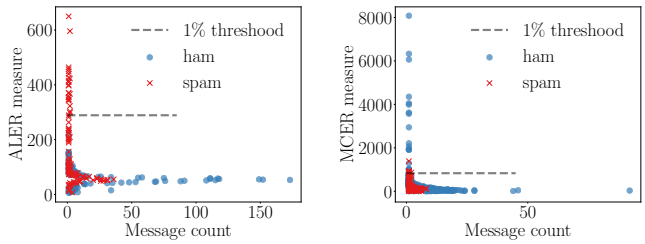


Figure 6: ALER and MCER measures of 3000 emails.

is observed that a set of spam and ham messages can be safely singled out by setting the threshold values of 1% for ALER and MCER, respectively. We further conduct extensive experiments (see Section 4.5) on various datasets and vary threshold values from 0.2% to 10%. Experimental results confirms that 1% is a confident threshold on various sized datasets in our system.

3.2 Reconstructing Words Corpora

We have roughly collected both spam seed and ham seed corpora from the target dataset. But these two seed sets are raw and too tiny if directly applied for spam detection. We next focus on mining inherent features of the roughly collected seed corpora for enhancing spam and ham words corpora through reconstruction. We aim to use the word distribution in the target dataset to reconstruct word corpora, taking advantage of such inherent features for improving our seed corpora. In the following, we give the details of sampling word distribution and reconstructing word corpora.

3.2.1 Sampling Words' Distribution. Given the spam seed corpus $\mathbb{S}_s$, ham seed corpus $\mathbb{S}_h$, and target dataset $\mathbb{D}_t$, we aim to derive a score for each word to represent the probability of a message classified as a spam if it contains the word. We use $s_i \in [0, 1]$ to denote such a score for each word i while using L to denote the label of a message, i.e., $L = 1$ for a spam message and $L = 0$ for a ham message. Thus, if a message m_j contains the word w_i, its probability of being a spam is $Pr(L(m_j) = 1)) = s_i$. Specifically, if this score is close to 1, the message m_j is more likely to be a spam; otherwise, it is more likely to be a ham. For each word in the spam seed or ham seed corpus, we set the spam scores to be the constant values c_s or c_h, respectively, with $0.5 < c_s \leq 1$ and $0 \leq c_h \leq 0.5$. For each of other words not in the seed corpora but in the target dataset, s_i represents the posterior probability which will be derived according to the distribution of a word presenting in the final spam or ham datasets. Since the target dataset is unlabeled, it is unrealistic to directly distinguish the spam and ham datasets so as to calculate the words distribution (i.e., the frequency of a word over the summation of all words' appearance frequencies) in each of these two datasets. The Gibbs sampling method [36] is employed here to iteratively generate the sampling labels and calculate the words' distribution. We define a vector $\boldsymbol{\theta}_L$ with the size of V, in which each element corresponds to one word and its entry represents this word's distribution in the dataset labeled with L. The sampling values of $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ can be generated by the following five steps.

Step 1: Label Initialization. Denote $\mathbb{S}_r$ as the union of spam and ham seed corpora, i.e., $\mathbb{S}_r = \mathbb{S}_s \cap \mathbb{S}_h$. The probability p_j of message m_j being labeled as a spam is calculated by

$$p_j = \frac{\prod_{w_i \in \mathbb{S}_r \cap \mathbb{S}_j} \bar{s}_i}{\prod_{w_i \in \mathbb{S}_r \cap \mathbb{S}_j} \bar{s}_i + \prod_{w_i \in \mathbb{S}_r \cap \mathbb{S}_j} (1 - \bar{s}_i)}, \tag{4}$$

where p_j is a posterior probability of Bayesian inference [17] and $\mathbb{S}_j$ represents the set of stem words in message m_j. Notably, for each word w_i in $\mathbb{S}_r$, the spam score $\bar{s}_i$ has been set to a constant value. Assume each message follows Bernoulli trial and we randomly label m_j as a spam with the probability of p_j. If all words from m_j are not in the seed corpora, m_j is designated as a spam with the probability of 0.5.

Step 2: Word Distribution Initialization. With the labels from *Step 1*, the target dataset splits into two datasets, with one for spam and the other one for ham. In each one, we count the total number of included messages, denoted as C_0 and C_1 for ham and spam, respectively. Meanwhile, we count the frequency of each word occurring in each dataset and use the vector $\boldsymbol{F}_L$ sized V to record all words' frequencies in dataset labeled as L, where each element corresponds to one word. With $\boldsymbol{F}_L$, we can initialize $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$.

Step 3: Updating Labels. This step repeats. In each iteration, we select one message m_j with the label of l. We count the frequency of each word in m_j and update vector $\boldsymbol{F}_l$ by subtracting such a frequency value from the corresponding term. We remove m_j from dataset at hand and update C_l by subtracting the message count by 1, i.e., $C_l = C_l - 1$. Before relabeling this message, we calculate the likelihood of such a message as spam or ham, respectively, without considering the spam/ham seed, denoted as $v_L (L = 1 \text{ or } L = 0)$:

$$v_L = \frac{C_L + \beta_{\pi L}}{C_0 + C_1 + \beta_{\pi 1} + \beta_{\pi 0} - 1} \prod_{w_i \notin \mathbb{S}_r,\, w_i \in \mathbb{S}_j} (\theta_L(i))^{\sigma_{ji}}, \tag{5}$$

where $\beta_{\pi 1}$ and $\beta_{\pi 0}$ represent the initialized hyper parameters of Beta distribution corresponding to spam and ham datasets, respectively. Referred to as the shape parameters of the Beta distribution, $\beta_{\pi 1}$ and $\beta_{\pi 0}$ are set to the uniform distribution. $\theta_L(i)$ represents the respective value of w_i in θ_L, i.e., the current distribution of word w_i in the dataset labeled as L. σ_{ji} is the frequency of word w_i occurring in message m_j. Since each message is assumed to follow the Bernoulli trial [26] for designating as a spam or ham, the probability of one message m_j to be a spam is calculated as follows:

$$p = \frac{v_1 \cdot \prod_{w_i \in \mathbb{S}_r \cap \mathbb{S}_j} \sqrt[2t]{\bar{s}_i}}{v_1 \cdot \prod_{w_i \in \mathbb{S}_r \cap \mathbb{S}_j} \sqrt[2t]{\bar{s}_i} + v_0 \cdot \prod_{w_i \in \mathbb{S}_r \cap \mathbb{S}_j} \sqrt[2t]{(1 - \bar{s}_i)}}, \tag{6}$$

where t is the number of sampling iterations. Then, we can assign a new label $\bar{l}$ to m_j with the probability of p being designated as a spam. Then, we add this message to the respective dataset according to the new label and increase the total count of messages in this dataset by 1, i.e., $C_{\bar{l}} = C_{\bar{l}} + 1$. $\boldsymbol{F}_l$ and $\boldsymbol{\theta}_l$ are also updated accordingly. This step is iteratively executed till all messages are selected.

Step 4: Updating Word Distribution Vectors. We assume all words distribution can be modeled as the Dirichlet distribution in both spam and ham datasets [25]. Assume there is a total of V words in the target dataset, then the Dirichlet distribution can be considered as the V dimensional distributions. Let's define a V dimensional vector t_L with each entry $t_L(i) = F_L(i) + \gamma_{\theta_L(i)}$, where $F_L(i)$ is the frequency value corresponding to w_i in $\boldsymbol{F}_L$ and $\gamma_{\theta_x(i)}$ is a hyperparameter of word w_i. Note that $\gamma_{\theta_x(i)}$ is added to the number of observed cases to avoid 0 observation in the dataset labeled with L, with its value set to 1 in general [36]. Then, we can sample $\boldsymbol{\theta}_L$ as $\boldsymbol{\theta}_L \sim \text{Dirichlet}(\boldsymbol{t}_L)$, where Dirichlet represents the Dirichlet distribution. We denote $\langle y_L(1), ..., y_L(V) \rangle$ as the sample of V words' frequencies and draw such V independent random samples from Gamma distribution, each with the density of

$$\text{Gamma}(t_L(i), 1) = \frac{y_L^{t_L(i)-1}(i) e^{-y_L(i)}}{\Gamma(t_L(i))}, \tag{7}$$

where Γ represents the Gamma function. The values of $\langle y_{x,1}, ..., y_{x,V} \rangle$ can be sampled via above Gamma distribution. Each value $\theta_L(i)$ in the sampling vector $\boldsymbol{\theta}_L$ can be calculated by

$$\theta_L(i) = \frac{y_L(i)}{\sum_{j=1}^{V} y_L(j)}. \tag{8}$$

Step 5: Recording $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$. We store the current sampling vectors $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ and go back to *Step 3. Step 3, Step 4,* and *Step 5* repeat until the average values of all elements in $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ converge.

3.2.2 Scoring Each Word. The aforementioned procedure can iteratively generate a series of samples and score them. We have stored the corresponding $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$. Let $\boldsymbol{\theta}_L^k$ denote the stored vector $\boldsymbol{\theta}_L$ in the k-th iteration. These sampling values can be used to calculate the spam score of each word by averaging the overall probability of a word in $\boldsymbol{\theta}_0(i)$ and $\boldsymbol{\theta}_1(i)$ from all iterations, i.e., for each word $w_i \notin \mathbb{S}_r$, the spam score s_i is calculated as follows:

$$s_i = \frac{1}{K} \sum_k \frac{\theta_1^{(k)}(i)}{\theta_1^{(k)}(i) + \theta_0^{(k)}(i)}, \tag{9}$$

where K represents the total number of sampling iterations.

3.2.3 Reconstructing Spam and Ham Word Corpora. After getting the spam scores of all words, we reconstruct the spam and ham word corpora that can better capture the spam and ham patterns, respectively. A word with a higher spam score means that any message containing it has a higher probability of being a spam. We rank the spam scores of all words and set a threshold to reconstruct the spam and ham corpora. In particular, we set a spam threshold to be 0.8 and collect all words with spam scores higher than 0.8 to serve as the new spam corpus. Meanwhile, the ham threshold is set to 0.4, with all words having the spam scores below 0.4 chosen to serve as the new ham corpus. When scoring words in seed corpora, we may initialize a word with higher score if it is deemed spam-prone; otherwise, we set it with a lower score. In the late case, the sampling process in Section 3.2 can help to find some other spam words with higher confidence. Up to this point, we have addressed reconstructing two more confident spam and ham corpora, with each word in the two corpora being associated with a spam score.

3.3 Training Spam Word Model

The reconstructed spam and ham corpora as well as their accompanying spam scores serve as the labeled dataset for training a spam word model. Such a model can be put to use for encoding each future message with a set of spam scores. To this end, we adopt a neural network (NN) model for training the latent patterns by

means of encoded words. As the NN model takes the high dimensional vectors as its input, we first have to encode each word in the reconstructed corpora via a high dimensional vector. Here, we employ the natural language processing (NLP) model to encode each word into a high dimensional vector for richly mining latent patterns. For example, for any two words having similar syntactic or semantic patterns, the NLP model will output alike vectors. This is critically important in capturing drift spammer patterns.

NLP Model Selection. There exist a cluster of NLP models that aim to get word encoding vectors, like word2vec [27], GloVe [33], and others, which are well trained for directly embedding all words into the high dimensional vector space. The embedded vectors then represent the syntactic (structural) and semantic (meaning) patterns of words from the respective words corpus. Such NLP models are in an unsupervised manner, therefore applicable to train our model for encoding the word vectors. However, this way is time-consuming unnecessarily without any performance guarantee. Instead, we rely on pre-trained models to get the encoded high dimensional vector of each word. There are lots of NLP models trained specifically for different platforms, such as SMS [57], Email [20], and Twitter[22]. Our empirical study has unveiled that these NLP models can be used across different platforms. Here, we select the model trained from Twitter corpus [33] to serve as our NLP model, which covers huge amounts of both normal and spam words.

Word Vector Extraction. The selected NLP model has a dictionary-like format, allowing us to look up the corresponding vector of a given word. This vector is then labeled with the spam score associated to the respective word. Each returned vector is denser, which thus covers a set of words that have a similar structure or meaning. In the case for a spammer to substitute one spam word with another similar one in order to evade detection, the dense vector can still cover it due to vector similarity. If the selected NLP model fails to recognize some words present in the reconstructed word corpora, we encode them with the "unknown" vectors and reassign high scores to them, since they tend to be spam words.

Training Spam Word Model. The dense vectors (denoted as $\mathbf{v}$) of all words in the reconstructed word corpora and their associated spam scores can be considered as the labeled ground-truth dataset to train our employed neural network for in-depth learning on inherent relationships among those dense vectors, with an aim at generating a spam word scoring model. We consider a simple five-layer neural network model, including *one Input layer*, *three Hidden layers*, and *one Output layer*. The *Output layer* adopts the standard *sigmoid* function, with a cross-entropy loss minimized by the gradient descent on the function output.

3.4 Outlier Detection

With the trained spam word model, we are ready to employ it for predicting all words' spam scores in the target dataset and to encode each message for the use of outlier detection.

Pre-processing Target Dataset. In Section 3.1, we have mined the stem words from each message and stored them in $\mathbb{S}_i = \{w_1, w_2, \cdots\}$. We then use the selected NLP model to look up the dense vector of each word in $\mathbb{S}_i$. If a word does not appear in the NLP model, it is encoded with an "unknown" vector and assigned to a high spam score. Each message is thus represented in the following format, $\mathcal{M} = \{\mathbf{v}_1, \mathbf{v}_2, ...\}$, where $\mathbf{v}_i$ represents the dense vector of a corresponding word w_i. The trained spam word model from Section 3.3 is used to predict the spam score of each dense vector $\mathbf{v}_i$, say s_i. The message is represented by a list of scores, with each element holding the value of corresponding word's spam score. We rank all spam scores in this vector and truncate each list to 10 elements, with 0 as padding values if the list includes less than 10 elements, . Then, a message is converted in the form of scores list $\mathbf{s} = \{s_1, s_2, ..., s_{10}\}$.

Outlier Detection. With encoded spam scores for each message, we employ the *Magnitude outlier* from the prominent outlier detection method [7], to expose the outlier property of spam messages. Let $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_N\}$ denote the encoded functional data for all messages in the target dataset. To reveal the magnitude outlier characteristic of a message with its encoded vector of $\mathbf{s}$, we calculate the intercept (denoted as $\hat{\alpha}_j$) and slope (denoted as $\hat{\beta}_j$) of the linear regression model of $\mathbf{s}$ in discrete version over each of other messages' encoding $\mathbf{s}_j \in \mathcal{S}$. The intercept $\hat{\alpha}_j$ is expressed by $\hat{\alpha}_j = \bar{s} - \hat{\beta}_j \bar{s}_j$, where $\bar{s}$ and $\bar{s}_j$ are the average values of all spam scores in $\mathbf{s}$ and $\mathbf{s}_j$, respectively, and $\hat{\beta}_j$ is the slope, defined as: $\hat{\beta}_j = \frac{\mathrm{Cov}(\mathbf{s}, \mathbf{s}_j)}{\mathrm{Var}(\mathbf{s}_j)}$, where $\mathrm{Cov}(\mathbf{s}, \mathbf{s}_j)$ is the covariance between $\mathbf{s}$ and $\mathbf{s}_j$ and $\mathrm{Var}(\mathbf{s}_j)$ is variance of $\mathbf{s}_j$. The magnitude index of each message can be calculated by:

$$I_m(\mathbf{s}, \mathcal{S}) = \left| \frac{1}{n} \sum_{j=1}^{n} \hat{\alpha}_j \right|. \tag{10}$$

After deriving the magnitude indices of all messages, we can rank index values from the largest to the lowest. The spam messages' property will be exposed to appear with larger values than non-spam messages in general.

4 EXPERIMENTS

We implement our platform-oblivious detection system and conduct extensive experiments to evaluate its performance. The main goal of this section is twofold. First, we run our system on different datasets from three social platforms and classify their respective spams to show its effectiveness in exposing spams' outlier property. Second, we compare our system with existing supervised and unsupervised methods in terms of spam classification performance. Besides, the necessity of each design component and the impacts of various parameters are also evaluated.

4.1 Implementation

System Settings. We screen the spam and ham seed corpora from the messages with top 1% ALER and MCER measures, respectively, in the target dataset. Notably, the selection of 1% shall be validated in Section 4.5. The spam and ham scores are set to be 0.8 and 0.4, respectively, in the seed corpora. Such two thresholds are also applied to identify the spam and ham words in the reconstruction of new word corpora. The dimension of embedded word vectors is the same as pre-trained NLP model [33], i.e., 25. The neural network model parameters adopted in Section 3.3 are given in Table 1. In the output layer, the mean squared error (MSE) is employed as the loss function, i.e., $\mathrm{MSE} = \frac{1}{n} \sum_i^n (s_i - \hat{s}_i)^2$, where n is the total number of words in both reconstructed spam and ham corpora, s_i is the spam score of a word w_i, and $\hat{s}_i$ represents its predicted score.

Table 1: Parameters of the neural network model

Layer Type	Drop off rate	# of Neurons
Input		25
Fully Connected ReLU	0.5	64
Fully Connected ReLU	0.3	32
Fully Connected ReLU	0.1	8
Output		1

Table 2: The spams, hams, and spam ratios of four datasets

dataset	Size	Spam	Ham	Spam Ratio
Kaggle SMS	5,572	747	4,825	13.4%
Metsis Email	20,681	4,146	16,545	20.0%
Twitter Trending	677,938	108,470	569,468	16.0%
Twitter Normal	5,823,230	355,217	5,468,013	6.1%

Datasets. We conduct experiments on 4 real-world datasets from three platforms, i.e., SMS, Email, and Twitter, depicted as follows.

- *Kaggle SMS dataset [5]* is a set of human-labeled cell-phone messages collected for research. There are a total of 5, 574 messages included, with each message labeled by a "ham" (legitimate) or "spam" tag. Our experiment uses all messages in this dataset for evaluation.
- *Metsis Email Dataset [26]* is a dataset including the email sources from Enron dataset, SpamAssassin corpus and others. We take the labeled 16, 545 ham and 4, 136 spam emails for our experiments.
- *Twitter Trending Dataset [4]* includes a total of 677, 938 tweets, collected in 2019, with focus on users who posted trending topics. A total of 108, 470 tweets are labeled as spams via the diversified approaches [61]of checking suspended accounts, clustering, and the rule-based method to pre-process the raw dataset and then performing manual checking.
- *Twitter Normal Dataset [3]* covers 2.5 million users collected from Twitter networks in 2019. There are a total of 5, 823, 230 labeled tweets, in which 355, 217 of them are labeled as spams via the same approach as in [61] with the combination of several approaches and the manually checking is finally conducted.

Table 2 summarizes the statistical information of the four datasets.

Compared Methods. We compare our solution to the existing both unsupervised and supervised methods on spam detection. The unsupervised methods include *Alien-l and Alien-s [30], OUSLD [34], JSF [58], Hashing [12], and Gibbs [15]*, in which Alien-l and Alien-s are outlier-based methods. The supervised methods include *Bayesian Inference [37], C4.5 [16], AdaBoost [60], SVM [48]*, and *Neural Netowrk (NN) [10]*.

Evaluation Metrics. We evaluate the performance of spam detection by using such standard metrics as recall ($Rec = \frac{DS'}{TS}$), precision ($Prs = \frac{DS'}{DS}$), and F_1 score ($F_1 = 2 \times \frac{Rec \times Prs}{Rec + Prs}$), where TS represents the number of spams in the dataset, DS denotes the number of detected spams, DS' means the number of detected spams that are indeed spams (i.e. true spams).

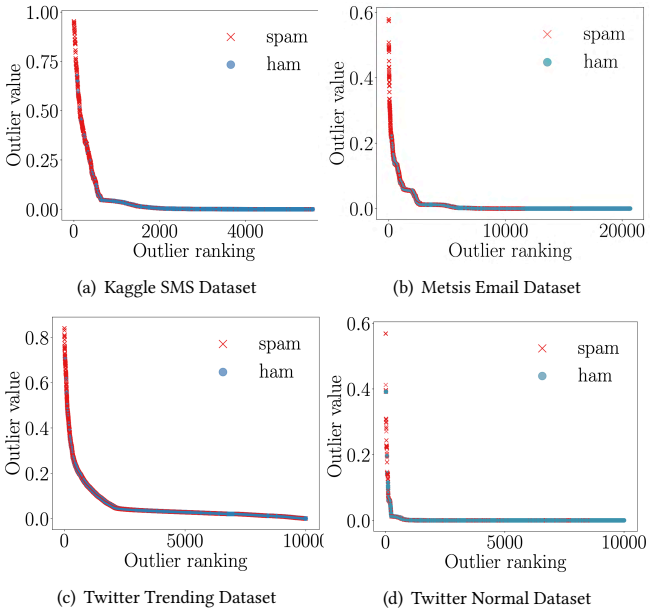


Figure 7: The ranking of outlier scores of all messages.

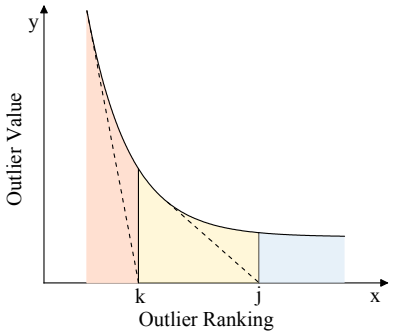


Figure 8: The Lorenz curve and its three zones, colored in orange, yellow and blue to indicate the *Spam Zone*, *Uncertain Zone*, and *Ham Zone*, respectively.

4.2 Outlier Exposure

We implement our proposed system with the setting in Section 4.1 and run it on four target datasets. The outlier results are ranked from the largest values to the lowest. Figures 7(a), (b), (c), and (d) show the ranked results from the datasets of Kaggle SMS, Email, Twitter Trending, and Twitter Normal, respectively, where the x-axis represents the ranking indices and the y-axis represents the outlier scores from our system. Red cross and blue points signify spam and ham, respectively. These figures exhibit our method can expose the outlier property of spam messages in all four datasets, where most of the spam messages have higher scores than ham messages, resided in the leftmost parts.

Although the outlier property is visible, how to set a threshold for effectively separating spams from hams is still a challenging

Table 3: Spam counts and ham counts in the *Spam Zone*, *Uncertain Zone*, and *Ham Zone* from each dataset

Dataset	Spam Zone		Uncertain Zone		Ham Zone	
Label	Ham	Spam	Ham	Spam	Ham	Spam
Kaggle SMS	21	311	339	320	4465	116
Metsis Email	394	1966	893	1376	15258	1492
Twitter Trending	7640	33424	40625	42591	532082	21576
Twitter Normal	20573	110154	86992	137683	5360448	107380

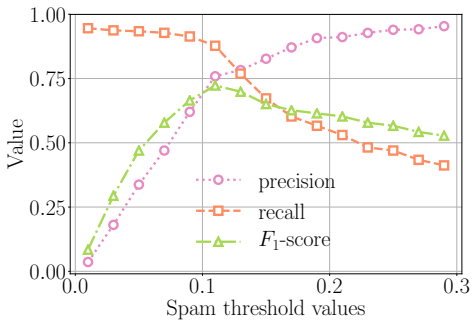


Figure 9: The performance of our system in Kaggle SMS dataset with various spam threshold values.

problem. Here, we propose a novel solution for intelligently identifying a suitable spam threshold value to single out spam messages. As the curves of all ranked values in Figure 7 follow exponential ones, we regress ranked values according to an exponential function, i.e., $L = e^{-\frac{a-F}{b}}$, where a and b are the parameters to fit the ranked outlier values. F is the ranking value for a given outlier score. This regression curve is also called Lorenz curve [14]. The Lorenz curve of ranked outlier scores depicted in Figure 7(a) is illustrated in Figure 8. With this curve, we take the leftmost point that has the highest score and draw a tangent line at this point, which intersects with the x-axis at a point k. Any point left to k will be considered as an outlier. We define the region left to point k as the *Spam Zone*, colored in orange in Figure 8. We draw another tangent line of the Lorenz curve with the slop of -1, and assume it intersects with the x-axis at point j. The region between the points of k and j is defined as the *Uncertain Zone*, while the region at the right of j is denoted as the *Ham Zone*. *Uncertain Zone* and *Ham Zone* are colored in yellow and blue, respectively. Table 3 shows the number of spams and hams in each region of every dataset. For the messages in *Spam Zone* and *Ham Zone*, we are confident to tell them as spams and hams, respectively. However, the messages in *Uncertain Zone* are hard to be classified surely. If we assume 50% of messages in the *Uncertain Zone* are spams, the spam ratio in a target dataset, denoted as R, can be estimated by: $R = \frac{n_1+0.5*n_2}{N}$, where n_1 and n_2 refer to the message counts in *Spam Zone* and *Uncertain Zone*, respectively. N is the total number of messages in the target dataset. Based on our estimation, the spam ratios of four target datasets are 11.9%, 16.9%, 12.2%, and 4.1%, respectively. When compared to the true spam ratios of target datasets, i.e., 13.4%, 20.0%, 14.6%, and 6.1%, respectively, from Table 2, our estimated spam ratios are very close to the true values.

Figure 9 shows the values of precision, recall, and F_1 score under different threshold values for the outlier ranking curve, for the Kaggle SMS dataset. It also exhibits the best performance to result from the threshold of around 11% to 13%, very close to the true spam ratio (i.e., 13.4%) and the estimated spam ratio (i.e., 11.9%). It implies the estimated spam ratios from our proposed solution can be safely employed by our system to filter out outlier spams effectively. In the following experiment, the threshold values for distinguishing spams among outlier ranking values are set to be 11.9%, 16.9%, 12.2%, and 4.1%, respectively, for datasets of Kaggle SMS, Metsis Email, Twitter Trending, and Twitter Normal.

4.3 Performance Comparison

We conduct extensive experiments on the four aforementioned datasets to compare our method with the existing unsupervised and supervised methods listed in Section 4.1.

Comparing to Unsupervised Methods. We run the existing unsupervised methods and our outlier method for 10 times on each dataset and calculate the averaged values of precision, recall, and F_1 score. Table 4 lists the complete results of our method and of existing unsupervised solutions on four datasets. We observe our method has the precision and recall of 89.6%, 79.4%, of 85.2%, 71.2%, of 88.7%, 82.6%, and of 85.1%, 78.4%, respectively, for the Kaggle SMS, Metsis Email, Twitter Trending, and Twitter Normal datasets. When comparing to other unsupervised solutions, our solution is observed to clearly outperform in terms of three performance metrics, except for the recall measure of Gibbs under Metsis Email dataset. This demonstrates the advantage of our method in terms of performance improvement. The reason is that our method explores the generating process of data, by iteratively refining data for use. Since a generative model renders a better learning capacity over a discriminative model [31], our method is thus able to capture spam patterns more precisely.

Comparing to Supervised Methods. We run the supervised methods listed in Section 4.1 on all four datasets via varying the ratio of training set over test set (denoted as r) from $1:10$ to $1:1$ for comparison with our method. Due to the page limit, we only include the detailed results from the Twitter Normal dataset under various ratios of training set over test set, as listed in Table 5. In the experiment, we keep the test set size as 1 million, and increase the training set from 0.1 million, 0.2 million, 0.33 million, 0.5 million to 1 million. The corresponding ratios of training set size over to test set size (i.e., r) are $1:10$, $1:5$, $1:3$, $1:2$ and $1:1$. From this table, we observe the performance of all supervised methods improves with the increasing of r, i.e., the increasing of training set size. However, the results of our solution do not change since it is an unsupervised method, without relying on the training dataset. It is also observed that when $r = \frac{1}{10}$, $r = \frac{1}{5}$ and $r = \frac{1}{3}$, our method can beat all supervised methods in terms of all three performance metrics. When $r = \frac{1}{2}$, our solution still outperforms all supervised solutions in terms of recall. For precision, all the supervised methods except SVM outperform our method. When $r = 1$, all supervised methods outperform ours in at least two metrics. These results demonstrate that all supervised methods highly rely on the training set size, with their performance level rising for a bigger training set size. However, in a large dataset, it is impractical to

Table 4: Comparisons of precision and recall under our solution and under unsupervised methods for all four datasets

Dataset	Kaggle SMS		Metsis Email		Twitter Trending		Twitter Normal	
Metric (%)	*Prs*	*Rec*	*Prs*	*Rec*	*Prs*	*Rec*	*Prs*	*Rec*
Alien-l [30]	79.4	51.1	82.2	59.4	79.4	53.3	74.1	60.6
Alien-s [30]	78.9	52.9	81.8	51.5	78.5	56.3	77.9	58.5
OUSLD [34]	54.4	42.5	58.2	57.9	61.1	51.4	60.0	52.5
JSF [58]	50.7	44.6	70.7	34.9	60.6	44.4	57.7	38.7
Hashing [12]	39.7	41.4	39.5	37.1	41.5	44.1	46.3	47.2
Gibbs [15]	64.5	55.0	63.9	72.0	45.2	67.1	51.5	66.2
Our Method	89.6	79.4	85.2	71.2	88.7	82.6	85.1	78.4

Table 5: Comparisons of supervised methods and our solution with various ratios of training set over test set, with the size of test set fixed to 1 million

r	Metric (%)	Bayes	C4.5	Ada	SVM	NN	Ours
$\frac{1}{10}$	*Prs*	80.2	73.2	76.5	73.9	77.4	**84.4**
	Rec	32.0	37.4	48.6	10.3	24.3	**77.4**
$\frac{1}{5}$	*Prs*	81.8	79.6	81.0	75.3	78.9	**84.4**
	Rec	38.5	44.9	52.5	10.3	23.8	**77.4**
$\frac{1}{3}$	*Prs*	82.5	80.8	83.1	78.4	79.3	**84.4**
	Rec	48.1	50.2	58.2	13.5	44.7	**77.4**
$\frac{1}{2}$	*Prs*	84.6	87.0	87.4	83.3	85.5	84.4
	Rec	63.6	61.5	60.1	66.7	71.2	**77.4**
$\frac{1}{1}$	*Prs*	88.0	84.7	91.5	85.1	90.6	84.4
	Rec	74.5	72.1	81.3	74.4	78.4	77.4

Table 6: Comparisons of the supervised solutions and our solution with various sizes of test dataset

Size	Metric (%)	Bayes	C4.5	Ada	SVM	NN	Ours
10k	*Prs*	89.4	89.2	90.5	87.2	90.6	85.5
	Rec	71.5	71.1	72.2	65.7	73.5	71.6
20k	*Prs*	85.1	83.9	83.7	83.5	85.5	85.3
	Rec	64.4	67.5	66.2	59.3	71.2	70.3
30k	*Prs*	84.7	83.1	82.6	83.2	85.2	**85.3**
	Rec	63.5	60.2	62.4	53.6	70.9	**71.0**
50k	*Prs*	83.2	81.3	82.2	81.4	83.1	**85.7**
	Rec	61.2	55.8	61.2	43.5	67.2	**72.2**

reliably label 50% (i.e., $r = \frac{1}{1}$) of dataset for training, since doing so may bring excessive human overhead. In contrast, our method does not require any ground truth labeling, but can still achieve moderate performance.

We next conduct experiments by fixing the training set size and varying the test set sizes to compare the performance of supervised solutions and our method. We take the Twitter Trending dataset as an example and select 10,000 labeled tweets, i.e., 1,446 spam and 8,554 ham messages, to serve as the training set. Table 6 shows the results of all supervised solutions and our method, with the test set size growing from 10,000 to 50,000. We observe the performance of our method fluctuates only slightly as the test set size grows, whereas all supervised counterparts suffer from fast performance degradation on all metrics with an increase in the test dataset size. This demonstrates the robustness of our proposed solutions.

Table 7: Results of Ablation Studies

Metrics (%)/Model	$\overline{Seed}$	$\overline{Gibbs}$	$\overline{NLP}$	Ours
Prs	62.6	48.3	68.5	**85.2**
Rec	71.7	51.4	70.1	**71.2**

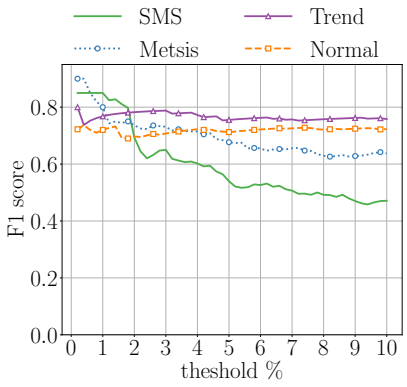
Specifically, when the test set size exceeds 30k, our method almost beats all examined supervised solutions.

4.4 Importance of Each Design Component

We conduct the ablation studies to evaluate the necessity and importance of each component in our design. In particular, the components of *Seed Collection, Word Copra Reconstruction,* and *Spam Word Model*, corresponding to Sections 3.1, 3.2, and 3.3, respectively, will be removed in turn, as design variants to evaluate the performance of the remaining system. The three corresponding variants are denoted as $\overline{Seed}$, $\overline{Gibbs}$, and $\overline{NLP}$, respectively. Table 7 presents the evaluation results (i.e., Precision and Recall) of $\overline{Seed}$, $\overline{Gibbs}$, and $\overline{NLP}$ under the Metsis Email dataset. From this table, we observe that all three variants perform worse than our complete system in terms of precision. Regarding the recall, our system performs markedly better than $\overline{Gibbs}$, slightly better than $\overline{NLP}$, but marginally worse than $\overline{Seed}$. However, considering both precision and recall metrics, which are important to signify the overall classification performance, we can conclude that all design components are necessary and important in contributing to our system performance.

4.5 Impact of Seed Threshold

We conduct experiments to show the impact of various threshold values for ALER and MCER on our system performance. Figure 10 shows the F1 scores of our system under various ALER and MCER thresholds, ranging from 0.2% to 10%. Notably, ALER and MCER are always set to the same value. From this figure, we can see our system performance on Twitter Trend and Twitter Normal datasets fluctuates only slightly when the threshold values increases from 0.2% to 10%, with most F1 scores being higher than 0.75. The reason is that the two datasets are large, so even when the threshold values rise to 10%, most of selected seeds are indeed spam or ham words. On the other hand, for the two small datasets Metis and SMS, an increase in threshold values will significantly degrade our system performances, since a large threshold value leads to a high false positive rate of spam seed corpus, thus substantially misleading



(a)

Figure 10: F1 scores of our system on four experimented datasets when varying the thresholds values of ALER and MCER measurement from 0.2% to 10%.

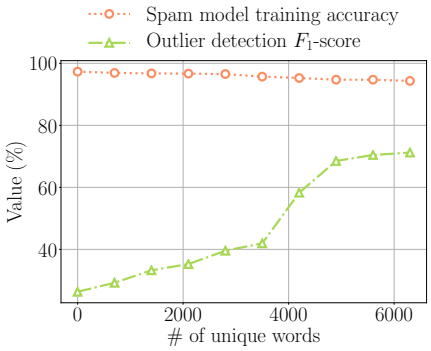
the remaining processing in our system. But from the four datasets, whose sizes range from 5, 574 to 5, 823, 230, our system can always achieve high F1 score values when setting the threshold value to 1%. As such, considering the dataset size may vary widely in practice, we believe it is safe to uniformly set the threshold values of ALER and MCER as 1%.

4.6 Impact of Word Richness

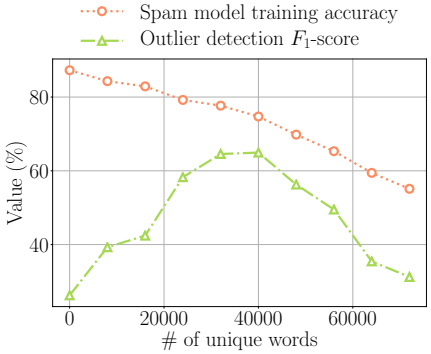
From Table 4, we observe that our solution performs differently across 4 target datasets. By analyzing the datasets, we find the difference resulting from unique words included in the datasets. We thus explore how the unique word count affects our system performance, by taking Kaggle SMS and Metsis Email datasets as the examples, which contain 6,300 and 78,000 unique words, respectively.

Figure 11 (a) and Figure 11(b) depict the trends of spam model training accuracy and our solution's F_1 scores as the unique word counts increase in Kaggle SMS and Metsis Email datasets, respectively. We observe the spam model training accuracy keeps decreasing with an increase in the unique word counts. The reason is that given more unique words, the regression task in the neural network model becomes more difficult. This degrades the accuracy of our spam word model.

On the other hand, it is found that when the number of unique words is small, the F_1 scores of our solution keep improving with an increase in the unique word count in both datasets. The reason is that if the number of unique words is small, the Gibbs sampling method employed in Section 3.2 is likely to *overfit* these words, causing the model to suffer from low generalization. It misleads to wrong spam scores in the spam model training phase, thus making our system perform poorly. With more words included, this overfitting problem can be alleviated to help improve the performance of our solution. However, when the unique word count becomes excessive, i.e., around 40,000 in Figure 11 (b), the Gibbs sampling method is likely to be saturated. In this situation, two words that



(a) Kaggle SMS dataset



(b) Metsis Email dataset

Figure 11: Impact of word richness in the target dataset.

are semantically or syntactically different may be forced to merge into the same distribution and thus to assign with similar spam scores. This will mislead our system, lowering the F_1 score.

5 RELATED WORK

Existing works in spam detection can be categorized into supervised, semi-supervised, and unsupervised methods.

Supervised Machine Learning methods rely on the ground truth dataset to let a machine learning classifier learn the latent patterns of spammers for classification. Extensive works are based on the fact that spammers and normal users behave differently, making it possible to extract effective features that can reveal such differences for the machine learning classifiers to learn latent patterns. These features include, but are not limited to, user profiles [13], user behaviors [23], message contents [21, 54], user relationships [9, 49, 62]. Feature extraction also becomes prevalent in other machine learning-based applications such as fake review or news detection [32, 40, 53], rumor detection [19, 63], etc. However, effective feature extraction has been well known as a challenging problem, especially if we aim to leverage them across different social platforms. Moreover, all of the aforementioned works require to have large-sized reliable ground truth datasets. It has been widely recognized that acquiring a large-sized reliable ground truth dataset

is a challenging and painful problem. More importantly, some research [29, 44, 52, 55] have realized that the supervised methods may encounter substantial performance degradation when classifying the imbalanced data (like spam messages only occupy a small portion).

Semi-Supervised Learning methods [11, 18, 24, 64] have been proposed to relax the reliance on the ground truth dataset. For example, Zhou *et al.* [64] explored the supervision power from multiple classifiers, where a labeling query occurs only if all classifiers are comparably confident on a disagreed unlabeled sample. Chen *et al.* [11] proposed an asymmetric self-learning approach that extracts "changed spams" from incoming tweets. Liu *et al.* [24] proposed a solution for extracting time-sensitive features to track the feature change, and Imam *et al.* [18] used unlabeled data to learn the structure of the feature space, helping to refine the result from supervised classifiers. SpamGAN [42] was also proposed by using both unlabeled and labeled datasets to train a GAN-based spam review classifier. However, they still rely on a certain amount of reliable ground truth datasets, which are not easy to acquire.

Unsupervised Methods aim to let the spam detection task free from labeling effort. Several categories of unsupervised methods have been explored, such as behavior-based, content-based, and graph-based ones, for spam detection. In the behavior-based category, Mukherjee *et al.* [28], and Wang *et al.* [51] have developed unsupervised solutions based on the observation that spammers and non-spammers behave differently, able to model such behavioral disparities by such features as frequency of activities, crawl actions, register duration, click behaviors, and others. In the content-based category, the hash values of the first k N-grams [56], the document complexity [46], locality-sensitive hashing [59], min-hash [12], and Natural Language Processing [34] have been investigated. In graph-based methods, the social network graph is leveraged and analyzed to find the differences of spams and hams [35, 43, 50] . However, existing unsupervised methods, in general, markedly underperform their supervised counterparts. Moreover, they are tailored to specific platforms, making them unable to adapt to multiple platforms.

Outlier Detection belongs to the unsupervised category as well and it relies on the fact that spammer's patterns have significant disparities versus those of the major data volume, thus possessing the outlier property potentially. Some works have been proposed to explore the outlier property through different technologies or approaches, i.e., data density [38], density-based clustering methods [8], principal component analysis (PCA) [39], combined artificial bee colony and k-nearest neighbors [6], and P-value [47]. However, the performance of all aforementioned approaches are unsatisfactory, especially when applying to different platforms.

6 DISCUSSION

This work relies on the fact that the spam messages account for a small portion in any social platform so that they can be exposed as the outliers. This holds true for almost all social platforms, where the existing anti-spam mechanisms have filtered the majority of spam messages, leaving out only a small portion of spam messages to represent an even smaller fraction of messages that pass anti-spam mechanism's checking, as also affirmed by earlier reports [1, 2, 41, 45]. This refers to the imbalanced data, where the supervised methods typically encounter dilemma [29, 44, 52, 55] when classifying the inside minority set (i.e., spam). Our work aims to overcome this dilemma by proposing an effective unsupervised solution, which can be integrated into the existing anti-spam mechanisms for further filtering spam messages.

Our solution can overcome the spammer feature drift problem to some extent due to the following two design strategies. First, we don't rely on feature extraction from any labeled ground truth dataset, and instead directly refine the spam and ham corpora existing in the target dataset. Second, for each word in the refined spam and ham corpora, we embed it into a high dimensional vector, which allows it to cover a richer set of words having similar syntactic or semantic patterns. Even spammers evolve by substituting some words with new ones in a spam, the resulting spam may still yield a vector for successful detection as an outlier.

This work only focuses on the English-based social platforms. It is interested for one platforms based on other languages, such as Spanish, Chinese, Japanese, Korean, and others, leaving as an open problem for future pursuit. On the other hand, our work targets text-only spam in linguistic-based social platforms. It is also necessary to develop a platform-oblivious solution to detect malicious images from social platforms like Facebook and Instagram, whereas the images take a large portion. This line of research will be deferred in our future work.

7 CONCLUSION

This paper has proposed a novel platform-oblivious spam detection framework that can work effectively across multiple social platforms to expose spams' outlier property for accurate spam detection. With new solutions developed to mine the spam and ham seed corpora from the target dataset and to reconstruct the refined corpora as the references for distinguishing spams from hams, our framework avoids reliance on laborious ground truth labeling and has the potential of capturing the spammer feature drift. Through employing the NLP and neural network models to train a spam word model, the framework can identify the words with similar semantic and syntactic information, without relying on the feature extraction employed by all previous supervised learning methods. It encodes all messages in the target dataset efficiently for effective processing by the outlier techniques to expose spams' outlier property. The results from extensive experiments demonstrate that our solution can indeed expose the outlier characteristics of the vast majority of spams. In addition, it is exhibited to outperform all examined unsupervised methods and can better supervised counterparts, with its performance kept consistently superior when applied to multiple platforms.

ACKNOWLEDGMENTS

This work was supported in part by NSF under Grants 1763620, 1948374, and 2019511. Any opinion and findings expressed in the paper are those of the authors and do not necessarily reflect the view of funding agency.

REFERENCES

[1] 2020. Facebook Transparent Report. https://transparency.facebook.com/community-standards-enforcement#spam.

[2] 2020. Twitter Transparent Report. https://transparency.twitter.com/en/platform-manipulation.html#platform-manipulation-jan-jun-2019.

[3] 2021. Twitter normal dataset. https://drive.google.com/open?id=1y-A0vOJ4amZ6P6oqjIryJKAoPItFbfWy.

[4] 2021. Twitter trending dataset. https://drive.google.com/open?id=1jfIdhjTUx_gtbYXhW1QwMzhfnIv7NgFe.

[5] Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of SMS spam filtering: new collection and results. In *11th ACM Symposium on Document Engineering*. 259–262.

[6] Reema Aswani, SP Ghrera, Arpan Kumar Kar, and Satish Chandra. 2017. Identifying buzz in social media: a hybrid approach using artificial bee colony and k-nearest neighbors for outlier detection. *Social Network Analysis and Mining* 7, 1 (2017), 38.

[7] Arturo Azcorra, Luis F Chiroque, Rubén Cuevas, A Fernández Anta, Henry Laniado, Rosa Elvira Lillo, Juan Romo, and Carlo Sguera. 2018. Unsupervised scalable statistical method for identifying influential users in online social networks. *Scientific Reports* 8, 1 (2018), 6955.

[8] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. 2006. Density-based clustering over an evolving data stream with noise. In *Proceedings of the SIAM International Conference on Data Mining*. 328–339.

[9] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the detection of fake accounts in large scale social online services.. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. 15–15.

[10] Patrick PK Chan, Cheng Yang, Daniel S Yeung, and Wing WY Ng. 2015. Spam filtering for short messages in adversarial environment. *Neurocomputing* 155 (2015), 167–176.

[11] Chao Chen, Jun Zhang, Yang Xiang, and Wanlei Zhou. 2015. Asymmetric self-learning for tackling twitter spam drift. In *Proceedings of the IEEE Conference on Computer Communications Workshops*. 208–213.

[12] Federico Concone, G LO RE, Marco Morana, and Claudio Ruocco. 2019. Twitter Spam Account Detection by Effective Labeling. In *3rd Italian Conference on Cyber Security, ITASEC 2018*, Vol. 2315.

[13] Emiliano De Cristofaro, Nicolas Kourtellis, Ilias Leontiadis, Gianluca Stringhini, Shi Zhou, et al. 2018. LOBO: Evaluation of generalization deficiencies in Twitter bot classifiers. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 137–146.

[14] Joseph L Gastwirth. 1971. A general definition of the Lorenz curve. *Econometrica: Journal of the Econometric Society* (1971), 1037–1039.

[15] Chris R Giannella, Ransom Winder, and Brandon Wilson. 2015. (Un/Semi-) supervised SMS text message SPAM detection. *Natural Language Engineering* 21, 4 (2015), 553–567.

[16] José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sánz, and Francisco Carrero García. 2006. Content based SMS spam filtering. In *Proceedings of the ACM Symposium on Document Engineering*. 107–114.

[17] John P Huelsenbeck and Fredrik Ronquist. 2001. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics* 17, 8 (2001), 754–755.

[18] Niddal Imam, Biju Issac, and Seibu Mary Jacob. 2019. Semi-supervised learning approach for tackling Twitter spam drift. *International Journal of Computational Intelligence and Applications* (2019).

[19] Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, and Jiebo Luo. 2017. Multimodal fusion with recurrent neural networks for rumor detection on microblogs. In *Proceedings of the 25th ACM International Conference on Multimedia*. 795–816.

[20] Diana Jlailaty, Daniela Grigori, and Khalid Belhajjame. 2017. Mining business process activities from email logs. In *Proceedings of the IEEE International Conference on Cognitive Computing*. 112–119.

[21] Stefan Kennedy, Niall Walsh, Kirils Sloka, Andrew McCarren, and Jennifer Foster. 2019. Fact or Factitious? Contextualized Opinion Spam Detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. 344–350.

[22] Jeongin Kim, Taekeun Hong, and Pankoo Kim. 2019. Word2Vec based spelling correction method of Twitter message. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 2016–2019.

[23] Bo Liu, Zeyang Ni, Junzhou Luo, Jiuxin Cao, Xudong Ni, Benyuan Liu, and Xinwen Fu. 2018. Analysis of and defense against crowd-retweeting based spam in social networks. *World Wide Web* (2018), 1–23.

[24] Jiaolong Liu. 2016. A time-sensitive spam filter algorithm dealing with concept-drift. In *Proceedings of the 4th International Conference on Machinery, Materials and Computing Technology*.

[25] Rasmus E Madsen, David Kauchak, and Charles Elkan. 2005. Modeling word burstiness using the Dirichlet distribution. In *Proceedings of the 22nd ACM International Conference on Machine Learning*. 545–552.

[26] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. 2006. Spam filtering with naive bayes-which naive bayes?. In *Proceedings of the CEAS*, Vol. 17. 28–69.

[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems*. 3111–3119.

[28] Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 632–640.

[29] Sankha Subhra Mullick, Shounak Datta, Sourish Gunesh Dhekane, and Swagatam Das. 2020. Appropriateness of performance indices for imbalanced data classification: An analysis. *Pattern Recognition* 102 (2020), 107197.

[30] Kazuyuki Narisawa, Hideo Bannai, Kohei Hatano, and Masayuki Takeda. 2007. Unsupervised spam detection based on string alienness measures. In *Proceedings of the International Conference on Discovery Science*. 161–172.

[31] Andrew Y Ng and Michael I Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems*. 841–848.

[32] Shirin Nilizadeh, François Labrèche, Alireza Sedighian, Ali Zand, José Fernandez, Christopher Kruegel, Gianluca Stringhini, and Giovanni Vigna. 2017. Poised: Spotting twitter spam off the beaten paths. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 1159–1174.

[33] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing*. 1532–1543.

[34] Feng Qian, Abhinav Pathak, Yu Charlie Hu, Zhuoqing Morley Mao, and Yinglian Xie. 2010. A case for unsupervised-learning-based spam filtering.. In *Proceedings of the SIGMETRICS*, Vol. 10. 367–368.

[35] Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 985–994.

[36] Philip Resnik and Eric Hardisty. 2010. *Gibbs sampling for the uninitiated*. Technical Report.

[37] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. 1998. A Bayesian approach to filtering junk e-mail. In *Proceedings of the Learning for Text Categorization: Papers from the 1998 workshop*, Vol. 62. 98–105.

[38] Markus Schneider, Wolfgang Ertel, and Fabio Ramos. 2016. Expected similarity estimation for large-scale batch and streaming anomaly detection. *Machine Learning* 105, 3 (2016), 305–333.

[39] Vatsal Sharan, Parikshit Gopalan, and Udi Wieder. 2018. Efficient Anomaly Detection via Matrix Sketching. In *Proceedings of the Advances in Neural Information Processing Systems*. 8069–8080.

[40] Saeedreza Shehnepoor, Mostafa Salehi, Reza Farahbakhsh, and Noel Crespi. 2017. NetSpam: A Network-Based Spam Detection Framework for Reviews in Online Social Media. *IEEE Transactions on Information Forensics and Security* 12, 7 (2017), 1585.

[41] Sri Harsha Somanchi. 2015. The mail you want, not the spam you don't. https://cloud.googleblog.com/2015/07/the-mail-you-want-not-the-spam-you-dont.html.

[42] Gray Stanton and Athirai A. Irissappane. 2019. GANs for Semi-Supervised Opinion Spam Detection. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 5204–5210.

[43] Enhua Tan, Lei Guo, Songqing Chen, Xiaodong Zhang, and Yihong Zhao. 2013. Unik: Unsupervised social network spam detection. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 479–488.

[44] Junjiao Tian, Yen-Cheng Liu, Nathaniel Glaser, Yen-Chang Hsu, and Zsolt Kira. 2020. Posterior re-calibration for imbalanced datasets. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 33. 8101–8113.

[45] TitanHQ. 2021. SpamTitan Anti Spam Solution - Block Over 99.9Spam. https://trust.titanhq.com/acton/media/31047/spamtitan-spam-filter-ma.

[46] Takashi Uemura, Daisuke Ikeda, and Hiroki Arimura. 2008. Unsupervised spam detection by document complexity estimation. In *Proceedings of the International Conference on Discovery Science*. 319–331.

[47] Maria Grazia Vigliotti and Chris Hankin. 2015. Discovery of anomalous behaviour in temporal networks. *Social Networks* 41 (2015), 18–25.

[48] V Vishagini and Archana K Rajan. 2018. An improved spam detection method with weighted support vector machine. In *Proceedings of the International Conference on Data Science and Engineering*. 1–5.

[49] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. 2017. SybilSCAR: Sybil detection in online social networks via local rule based propagation. In *Proceedings of the IEEE Conference on Computer Communications*. 1–9.

[50] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. 2018. Sybilblind: Detecting fake users in online social networks without manual labels. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses*. 228–249.

[51] Gang Wang, Xinyi Zhang, Shiliang Tang, Haitao Zheng, and Ben Y Zhao. 2016. Unsupervised clickstream clustering for user behavior analysis. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 225–236.

[52] Yiru Wang, Weihao Gan, Jie Yang, Wei Wu, and Junjie Yan. 2019. Dynamic curriculum learning for imbalanced data classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5017–5026.

[53] Kun Wu, Xu Yuan, and Yue Ning. 2021. Incorporating Relational Knowledge in Explainable Fake News Detection. In *Pacific-Asia Conference on Knowledge*

Discovery and Data Mining. 403–415.

[54] Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y Zhao. 2017. Automated crowdturfing attacks and defenses in online review systems. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 1143–1158.

[55] Jian Yin, Chunjing Gan, Kaiqi Zhao, Xuan Lin, Zhe Quan, and Zhi-Jie Wang. 2020. A novel model for imbalanced data classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 6680–6687.

[56] Kenichi Yoshida, Fuminori Adachi, Takashi Washio, Hiroshi Motoda, Teruaki Homma, Akihiro Nakashima, Hiromitsu Fujikawa, and Katsuyuki Yamazaki. 2004. Density-based spam detector. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 486–493.

[57] Lina You, Yujun Li, Yue Wang, Jie Zhang, and Yang Yang. 2016. A deep learning-based RNNs model for automatic security audit of short messages. In *Proceedings of the 16th International Symposium on Communications and Information Technologies*. 225–229.

[58] Jianfei Yu and Jing Jiang. 2015. A hassle-free unsupervised domain adaptation method using instance similarity features. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. 168–173.

[59] Qunyan Zhang, Haixin Ma, Weining Qian, and Aoying Zhou. 2013. Duplicate detection for identifying social spam in microblogs. In *Proceedings of the IEEE International Congress on Big Data*. 141–148.

[60] Xipeng Zhang, Gang Xiong, Yuexiang Hu, Fenghua Zhu, Xisong Dong, and Timo R Nyberg. 2016. A method of SMS spam filtering based on AdaBoost algorithm. In *Proceedings of the 12th World Congress on Intelligent Control and Automation*. 2328–2332.

[61] Yihe Zhang, Hao Zhang, Xu Yuan, and Nian-Feng Tzeng. 2019. Pseudo-honeypot: Toward efficient and scalable spam sniffer. In *Proceedings of the 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 435–446.

[62] Yihe Zhang, Hao Zhang, Xu Yuan, and Nian-Feng Tzeng. 2019. Tweetscore: Scoring tweets via social attribute relationships for twitter spammer detection. In *Proceedings of the ACM Asia Conference on Computer and Communications Security*. 379–390.

[63] Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*. 1395–1405.

[64] Zhi-Hua Zhou and Ming Li. 2010. Semi-supervised learning by disagreement. *Knowledge and Information Systems* 24, 3 (2010), 415–439.