Vol.15, Issue No 2, 2025

Preventing Evasion in SMS Spam Detection and Classification

[1] Chava Sravya, [2] Gadhi Jaya Narasimha Manikanta, [3] Vaara Pavan Kumar, [4] Addanki Ricky Paul Adams, [5] Dr. B. V. S. Varma

[1,2,3,4] Students, Dept. of CSE, DNR College of Engineering & Technology, Balusumudi, Bhimavaram, India.

[5] Professor, Dept. of CSE, DNR College of Engineering & Technology, Balusumudi, Bhimavaram, India.

ABSTRACT

Research into creating systems that can effectively handle the evasive tactics employed by spammers is necessary since the permanence of SMS spam is still a big problem. To protect the public from the harmful effects of SMS spam, research like this is crucial. The purpose of this research is to provide light on the difficulties inherent in the present state of SMS spam filtering and detection. In light of these difficulties, we provide a fresh SMS dataset consisting of over 68,000 messages, 61% of which are genuine (ham) SMS and 39% of which are spam. It is worth mentioning that this dataset is the biggest publicly accessible dataset on SMS spam that we have released so far for the sake of future study. By following the development of spam over time, we are able to describe the dataset. We then assess and contrast the efficacy of popular machine learning-based SMS spam detection algorithms, including both basic and sophisticated deep neural networks, by extracting syntactic and semantic data. We look at how well-known anti-spam services and current methods for detecting SMS spam hold up against spammers' evasion strategies. Our research shows that when it comes to correctly identifying spam SMS messages, most anti-spam services and methods based on shallow machine learning perform poorly. It has come to our attention that every machine learning method and anti-spam service is vulnerable to the many evasive tactics used by spammers. In order to overcome these constraints, our work encourages more research into these areas so that anti-spam services and SMS spam detection may progress.

Keywords: anti-spam services, evasive tactics, machine learning robustness analysis, spam dataset, development of SMS spam, detection of spam.

INTRODUCTION

The identification of SMS spam1 has been the subject of study for about twenty years [1, 2, 3, 4, 5, 6, 7], yet it remains a difficult and significant problem for our contemporary digital society. Recent years have seen worrisome levels of SMS spam; in the United States alone, victims were conned out of an estimated $330 million in 2022—more than twice the amount lost in 2021 [8]. Similarly, the ScamWatch committee of the Australian Competition and Consumer Commission (ACCC) estimated a near-doubling of yearly losses from 175 million Australian dollars in 2020 to 323 million Australian dollars in 2021. The number of complaints of SMS fraud increased to 67,180 in 2021 from 32,337 in 2020; the largest number of reports of SMS scams, with more than 8,835 in February 2022, was recorded for all scam delivery methods combined. We highlight four key obstacles to preventing SMS spam in this work: Data Availability: Due to a lack of big, real-world, annotated datasets, developing models for SMS spam detection has proven to be rather difficult. Many previous studies [4, 7, 10, 11, 12, 13] used small, unbalanced datasets with just a few hundred spam messages, which are obsolete and not representative of the real world. To the best of our knowledge, the two most current datasets on SMS spam are SpamHunter Dataset [14] and SMS Spam Collection [4]. Despite only having 747 spam messages, the SpamHunter Dataset has 947 annotated spam messages, making the SMS Spam Collection outdated (released in 2012). Their usefulness in fighting SMS spam is called into question by the fact that they do not include current data and only capture a small number of SMS spam messages.This constraint makes the model less generalizable and less effective on unknown data, and thus raises the likelihood of overfitting [15]. Missing Datasets for Benchmarks: The lack of a standardized benchmark dataset for thorough comparisons [18], [21] has caused research in the domain of SMS spam detection to remain fragmented, despite the several strategies that have been suggested for this purpose [5, 16, 17, 18, 19, 20]. It is difficult to assess the effectiveness of many suggested detection approaches due to the absence of consistent datasets, which in turn leads to an ambiguous description of the models' performance. Resistance to Deceptive Methods: The inadequacy of

current anti-spam ecosystems and suggested machine learning (ML) models to withstand fraudsters' evasion tactics is another significant obstacle to reducing SMS spam. Spammers are always coming up with new, sneaky ways to get past spam filters. Although ML and deep learning models have the potential to enhance SMS spam detection, new research using NLP has shown that evasive techniques can trick different ML-based spam models with small text changes, making the threat of SMS spam even greater [7, [22], [23], [24]. There seems to be a lack of study into the efficacy of the SMS anti-spam ecosystem in combating evasive approaches, even though they pose a significant danger to SMS spam models. Also, most of the machine learning models that have been published so far have been tested using old-fashioned evasive methods, completely disregarding the most recent dangers that have emerged due to the explosion of web-based bulk SMS providers. Problem with Concept Drift: Scammers' ever-changing tactics provide a challenge to SMS spam filters. There are more options and methods for spammers to use these evasive strategies due to the convergence of mobile networks and the internet. Modern SMS spam may be undetectable by ML-based spam detectors trained on older datasets (such as Table 3) or with a small number of spam cases. Concerning the effect of idea drift on SMS spam filtering, no research has been published. Furthermore, there is a notable lack of studies about the development of spam SMS. In order to create effective defenses, it is essential to get insights on the evolution of the underlying spam. The following are some of the contributions made by this study that deal with the difficulties of automated SMS spam detection: Publication of a Massive SMS Spam Database: With 67,018 English texts culled from a variety of sources spanning more than a decade, we are proud to provide a new, massive SMS scam dataset. It contains both valid and spam communications; the former accounts for 60.9% and the latter for 39.1%. Sources like ScamWatch2 and Action Fraud contribute to the dataset. 3 As far as we are aware, this tagged SMS spam dataset is the biggest one available. With an average of 5,574 messages, it far outstrips the capabilities of existing datasets like [4], [5], [10], [11], [16], and [26]. As part of our work, we are consolidating and aggregating data collected from both online sources and volunteers. Also, as mentioned in section III, we carried out preparatory operations such as deduplication, removing non-English messages, converting SMS photos to text, and categorizing more than 60,000 SMS messages. We have made our dataset and source code available to the research community at https://github.com/smspamresearch/spstudy (Anonymous) in the hopes that it would encourage further studies in this field. Determination of SMS Spam by Means of Machine Learning: Using several methods such as supervised text classification, deep learning, one-class learning, and positive unlabeled learning, we assess how well various important model architectures for SMS filtering perform (see to §VI-A). Neither of these latter two methods has ever been used for the purpose of detecting spam in SMS messages. The Bag of Words method obtained the maximum F1 score of 45%, indicating that one-class learning strategies did not perform well. The Word2Vec approach, which is often associated with supervised learning, obtained an F1 score of 79%. This is comparable to the 83% F1 score attained by two-class SVM with Word2Vec, and other positive unlabeled techniques also produced respectable results. In addition, we assess ML models that make use of various feature sets and word embeddings (§V-B). These models encompass a range of approaches, including static and dynamic versions of semantic context-based vector space models, non-semantic count-based vector space models (e.g., Count Vectorization, TF-IDF), and semantic non-contextbased vector space models (e.g., Word2Vec, fastText, GloVe). With the exception of one-class learning, we discovered that every model performance was enhanced by using semantic-based word embedding. Methods for Avoiding ML-Based SMS Spam Detection: To avoid detection in Smish [28] or Punycode [29] spam messages and embedded URLs, spammers could use obfuscation or perturbation techniques, for example, [27]. The strength of ML models is being tested against four different kinds of evasive perturbations, which are described in section VI-B: char-level, word-level, sentence-level, and multi-level evasive strategies. We discovered that all ML and DL models are susceptible to various forms of evasive manipulation of SMS texts and offer novel evasive ways to counter them. As far as we are aware, no previous research has pitted ML models against the evasive tactics used by spam SMS attackers in their investigation. Analysis of Concept Drift: We examine the features and new spammer strategies of SMS spam, as well as longitudinal trends, using our dataset of spam SMSes from 2012 to 2023. We also provide a study of the evolution of SMS spam (cf. §IV). Furthermore, we examine how idea drift affects SMS spam filtering by testing machine learning classifiers on a dataset that is time-stamped by year (see to §VII). The findings show that the efficacy of models for detecting SMS spam is significantly inversely related to idea drift. Testing the SMS Anti-Spam Ecosystem's Robustness: We test the SMS Anti-Spam ecosystem in the real world against various evasive scenarios to see how it holds up. Popular text messaging apps with spam filtering and APIs for third-party anti-spam web services are the two most essential parts of this ecosystem that we show here. No prior research has, as far as we are aware, tested the SMS anti-spam ecosystem in a real-world setting to see how well it fares against spammers' covert tactics. We found that third-party services are not good for SMS filtering and that the anti-spam text messaging software is susceptible to evasive approaches as well.

RELATED WORK
SMS SPAM DATASETS

Research on SMS spam identification was made possible by the availability of several SMS datasets. These datasets are small, however, and they don't include many spam messages. Table 1 summarizes the most important SMS spam datasets, starting with the most recent one from 2022 and going all the way back to 2012. With spam messages dating back to before 2010 and an uneven dataset, the 2012-released SMS Spam Collection [4] is severely outdated. There were 5,574 mails altogether, with just 747 being spam. The SMS Spam Corpus v.0.1 Big [30] and the now-defunct Grumbletext website4 (a UK forum) were used to collect the spam messages. The most recent update was made on March 9, 2015, and there are 67,093 SMS messages in the NUS SMS Corpus [31]. To gather and extract spam data from publicly published SMS images on Twitter, the "SpamHunter" framework was recently suggested [14]. Over the course of five years (2018–2022), SpamHunter was used to collect and publish 25,889 tweets in various languages. Although the SpamHunter method is unique, the SMS spam dataset it produces is quite noisy due to the inclusion of many benign and awareness messages that were crawled and added inadvertently. There are a lot of duplicate messages and optical character recognition mistakes in the collection. Incorporating this noise into the ML model might lead to severe mistraining, necessitating a human review to eliminate mistakes and noise. They examined 1,000 messages at random from the dataset for their own investigation and found 53 messages that weren't spam but were mistakenly included. In this work, we provide a new massive SMS spam dataset that was compiled from many sources, such as public datasets, scam observatories, Twitter (formerly known as X), and volunteers, in addition to a team of professionals who manually tagged a huge number of SMS for accuracy.

TABLE 1. Spam SMS datasets used in the literature.

Dataset	Year	# of SMSes	# of Spam (Labelled)	Recent Studies
SMS Spam Collection [4]	2012	5,574	747	[32], [33], [7], [34], [35]
NUS SMS Corpus [31]	2015	67,063	Nil	[36], [37], [38]
SpamHunter [14]	2022	25,889	947	[39], [40]

Assessing the Long Term Anti-Spam System

The use of SMS Spam Collection for spam prevention in well-known Android text messaging applications was studied by Akshay Narayan et al. [19]. Unfortunately, most of the apps that were tested in their research are no longer in production. Similarly, Tang et al. [14] examined a collection of widely used applications for both iOS and Android, as well as bulk SMS providers and anti-spam tools. But all they can look at is the spam that Twitter users have sent and received. Crucially, they failed to conduct testing to ascertain how well these apps and services resisted spammers' evasive tactics. In order to determine how resilient certain apps and services are, our review covers more ground than just performance; it also examines evasive strategies. In addition, as mentioned in §III, our test collection includes examples from several sources.

DATA COLLECTION AND AUGMENTATION

We gather and enhance a dataset of SMS spam messages that spans over ten years in order to tackle the first two obstacles (see to §I).A. Gathering Information We scoured the web for publicly accessible SMS statistics and compiled them using an exhaustive poll. We used search phrases like "SMS," "SMS messages," "SMS dataset," "Text Messages," and "Short Message Service" to scour different online resources and GitHub projects. Only scholarly publications, GitHub repositories, and search results that referenced publicly accessible datasets were included in our filtering process. Using a variety of public and free-for-research sources, we were able to compile a corpus of 179,440 SMS examples in numerous languages. In Table 3 you can see all of the sources that were used. To find tweets on SMS spam, we used the aforementioned sources as well as targeted searches on Twitter. These tweets were made public in the form of screenshots or photographs. In order to be sure that we covered all the bases, we crawled and gathered these tweets from 2012–2017 and again from 2022–2023, stretching beyond the

SpamHunter dataset. In addition, we retrieved public photos and screenshots of victims' reported SMS frauds from scam observatory websites such as Scamwatch and Action Fraud. In addition, college students were asked to participate in our research by sending any text messages they received to a specific number that was utilized to gather data. The time frame for this endeavor was from October 2021 to June 2023. We made sure the volunteers knew their efforts would be available to everyone. The following data sets were compiled: 1,130 spam messages from Twitter, 203 spam SMS messages from observatories, and 3,712 SMS messages (1,387 spam, 2,325 ham) from volunteers. We have included these freshly acquired communications into our corpus; they were previously unreported.Section B: Data Augmentation Our main goal is to identify English-language SMS spam. Initiating the data preparation step, we remove duplicate and non-English messages from our combined dataset in order to achieve this. For this aim, we install a filtering system with two passes. We initially use the langdetect [49] package in Python to identify the language of each SMS. Messages that are determined to be not in English are quickly removed. Second, the remaining SMS texts are filtered using the Googletrans [50] library. Any communications that are not in English will also be removed from the dataset by this further filtering phase. Because there is a limit on the amount of free API requests that users may make, we utilize the Googletrans API for the second round of filtering. Additionally, we used Python's pytesseract [51] module to transform pictures—screenshots of SMSes—found on Twitter and scam sites into text. After filtering out duplicates, non-English messages, and labels, we were left with a dataset consisting of 4,904 unique English language SMSes, down from 62,114 in the aggregated dataset. We manually identify 60,032 of these SMSes as either Ham (legal) or Spam using a series of criteria (see Table 2). Our manual classification of SMS messages as spam or ham is governed by a set of nine criteria, which are shown in Table 2. After reviewing several sorts of scams on scam observatories (scam watch, Action Fraud), analyzing labeled SMS corpora, and having a discussion between the writers, the recommendations were formed. Three researchers worked together to examine the aggregated dataset for unlabeled SMSes and assign labels based on predetermined criteria. A consensus was reached after talks on disagreements over given labels. The "Super Dataset" (Table 4) is the end product of merging the "augmented dataset," which include SMSes from scam observatory, Twitter, and volunteers with the original dataset. A total of 67,018 SMS messages are included in the Super Database. Out of them, 40,837 (60.9%) are legal messages and 26,181 (35.1%) are spam. C. SUMMARY Among the more than 60,000 SMS messages we gathered and annotated were 4,904 brand-new messages from a variety of sources. With our

TABLE 2. Rules for labeling spam SMSes in our dataset.

Rule	Label	Description
Rule1	Spam	Promotional or unwanted messages (advertising, prose-lytizing, etc)
Rule2	Spam	Containing "unknown" URLs in the text message
Rule3	Spam	Asking users to contact on email within text message
Rule4	Spam	Asking users to contact back on the same number or another contact number within the text message.
Rule5	Spam	Asking users for personal or sensitive information
Rule6	Spam	Asking or requesting users for the payment
Rule7	Spam	Asking users to forward or circulate the message
Rule8	Spam	Asking users to download or install a file
Rule9	Ham	Containing text, details of "well-known" services/URLs

TABLE 3. Overview of SMS Spam datasets consolidated to generate an augmented dataset.

Dataset	# of SMSes	Language	Labeling	Year
UCI [4], [5]	5,574	English	Labelled	2012
NUS [31]	67,063	Multi	Unlabelled	2015
Github1 [52]	77,039	Multi	Unlabelled	2019
Github2 [53]	557	English	Labelled	2018
Gupta [6]	3,318	Multi	Labelled	2018
SpamHunter [14]	25,889	Multi	Partial	2022
Consolidated	179,440	Multi	Partial	-
Consolidated [Augm.]	**62,114**	**English**	**Partial**	-

TABLE 4. Characterisation of super dataset.

Dataset	# of SMSes	# of Spam SMSes	
		2012-2017	2018-2023
Consolidated [Augm.] (cf. Table 3)	62,114	1,190	22,071
DS7 [Volunteers]	3,712	Nil	1,387
DS8 [Scamwatch, ActionFraud]	203	Nil	203
DS9 [Twitter]	1,330	223	1107
Super Dataset	**67,018**	**1,413**	**24,768**

contains SMS data from a wide variety of sources, making it the most comprehensive collection available. Compared to other research efforts, our contribution of 2,920 classified spam messages is a considerable increase (for comparison, see Table 1).

EVOLUTION OF SMS SPAM: AN ANALYSIS OF CHANGING CHARACTERISTICS

Here we will examine the characteristics of SMS spam and the strategies used by spammers using our longitudinal dataset that covers the years 2012–2023. Our goal is to discover longitudinal patterns and learn how SMS spam has changed over the years. Based on their release date, we time-stamped the full SMS collection. We next separated the dataset according to its publication date and created two sets: "DS legacy" with 37,615 SMS messages from datasets released between 2012 and 2017 (including Twitter spam messages collected during this time) and "DS latest" with 29,403 SMS messages from datasets and other sources published between 2018 and 2023.

The results show that spammers' efforts to avoid detection and target people successfully have changed, and that there have been major alterations in SMS spam tactics. A. Text-Based Analysis In addition, we check the dataset for SMSes that are grammatically, semantically, and spelling correctly. 1) Errors in Spelling As a strategy to evade spam filters, spammers intentionally use misspelled words [54]. Using the pyspellchecker [55] module in Python, we added a mistake_to_ tokens_ratio to properly evaluate the spelling problems in SMSes of different lengths, and we were able to quantify this phenomena. From 2012–2017, the average mistake_to tokens_ratio was 18%; from 2018–2023, it jumped to 33% (see Figure 1). The increasing frequency of misspellings highlights spammers' dogged pursuit of ways to evade spam filters, even though auto-correction tools are standard on most new mobile phones.

TABLE 5. Lexical analysis of the super dataset.

Data	Avg Chars	Avg Words	Avg Sent	Avg Sent Length	Avg Word Length	Richness	ARI	Flesh Score
Overall	96.05	17.24	1.92	28.06	4.51	0.95	6.31	80.1
Ham	65.14	13.0	2.58	23.29	4.13	0.97	2.13	93.70
Spam	144.26	23.86	0.90	35.50	5.12	0.93	12.81	60.95

FIGURE 1. CDFs of ratios of spelling mistake to words (also termed as tokens or keyword) in Spam SMSes.

READABILITY

The next step is to extract the word count, sentence length, punctuation, non-letters (like emoticons), lexical richness, Automated Readability Index (ARI) and Flesch score from each SMS in our dataset [56]. The ARI is a popular reading measure that evaluates the comprehensibility of a text corpus. Lexical richness is the ratio of unique words to total words, which shows obvious repeats. It is calculated as stated in Equation 1. A higher score indicates that the text is simpler to read. Our findings are summarized in Table 5. According to the data, spam SMS users had a lower Flesch score (93.7 vs. 60.9) and lexical richness (97% vs. 93%), but a higher ARI (2.13 vs. 12.81). This suggests that while ham SMSes have a broader vocabulary, spam SMSes have inferior readability. From 2012–2017, the average readability index of spam SMSes was 80.2, but from 2018–2023, it dropped to 60.4.

$$\text{ARI} = 4.71 \times \text{average word length} + 0.5 \times \text{average sentence length} - 21.43 \quad (1)$$

SPAM SMS Length

In order to avoid detection and maximize the delivery of their messages, SMS spammers use a range of message lengths. We observed that the average length of spam SMSes was relatively constant, at 137 characters from 2012 to 2017 and 143 characters from 2018 to 2023, while the length of spam SMSes remained high compared to Ham SMSes (see Figure 2). This regularity in duration indicates that spammers have discovered the sweet spot between being undetectable and getting their points across.

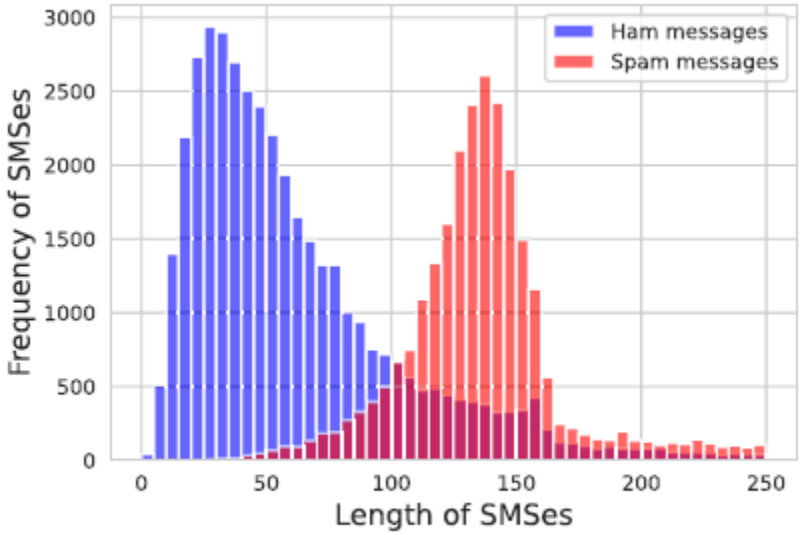


FIGURE 2. Length (in characters) vis-'a-vis Frequency distributions of SMSes in our dataset.

PREVALENCE OF URLS AND URL-SHORTENING SERVICES

We manually identify the URL-shortening provider used in our collection of spam SMSes and use regular expressions to extract URLs from the messages. From 2012–2017, only 13% of spam messages included URLs; from 2018–2023, however, that number jumped to 37%, indicating a dramatic rise in the use of URLs in SMS spam. The trend here shows that spammers are becoming craftier at using URLs to spread malware and trick people into clicking on dangerous links. There has been a noticeable uptick in the creation of spam URLs via URL-shortening providers. It is possible that these services allow spammers to conceal the true destination of URLs, monitor click-through rates, circumvent spam filters, and fit long or harmful links within the character constraints of SMS messages [57].

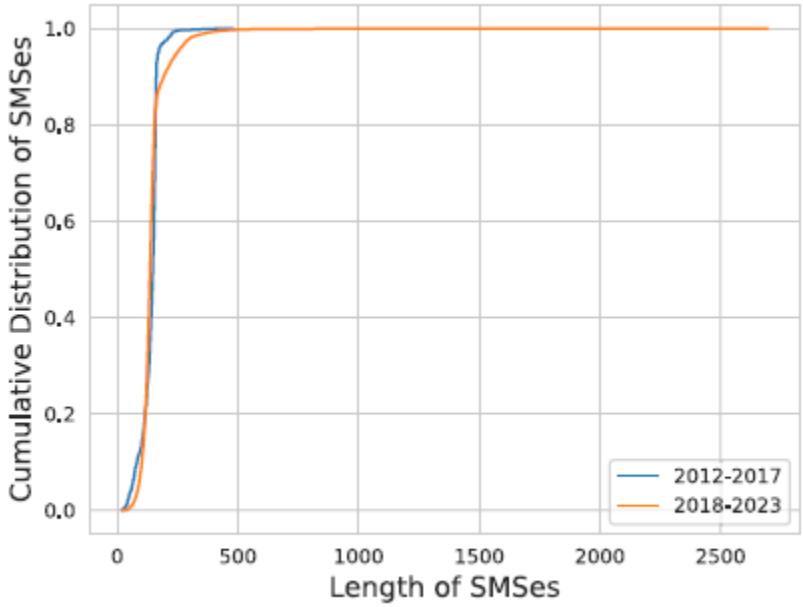


FIGURE 3. Distribution of SMS length (in characters) in Spam messages.

The five best URL shortening providers found in our Super dataset are shown in Table 6. Fewer than one percent of spam operations used URL shortening providers between 2012 and 2017. The percentage of spam operations that

used URL shorteners is far lower, at 9.76%. The increasing use of URL shorteners highlights their function in enabling the efficient and anonymous spread of spam.

TABLE 6. Top 5 URL shortner services identified in spam SMS in super dataset.

Service Name	# of Unique Occurrences	
	2012-2017	2018-2023
bit.ly	11	1154
goo.gl	2	990
tinyurl.com	1	80
cutt.ly	-	71
wa.me	-	41

ANALYZING ML-BASED SPAM DETECTION TECHNIQUES

Preprocessing the combined dataset, comparing feature models, choosing appropriate machine learning techniques, and testing the effects of various evasive techniques on the ML models are all parts of our experimental methodology, as shown in Figure 4. In what follows, we'll go into further detail about these procedures. Division and processing of data Preprocessing is applied to the combined dataset in order to eliminate stop words and superfluous letters. To do this, we use the NLTK library [58]. In addition, we used the scikit-learn [59] software to divide the dataset into three parts: train (80%), test (20%), and hold-out (Tables 7 and 8, respectively, provide the lexical analysis of the train and test splits). The 225 randomly chosen spam SMS that make up the hold-out set are there only for validation reasons.

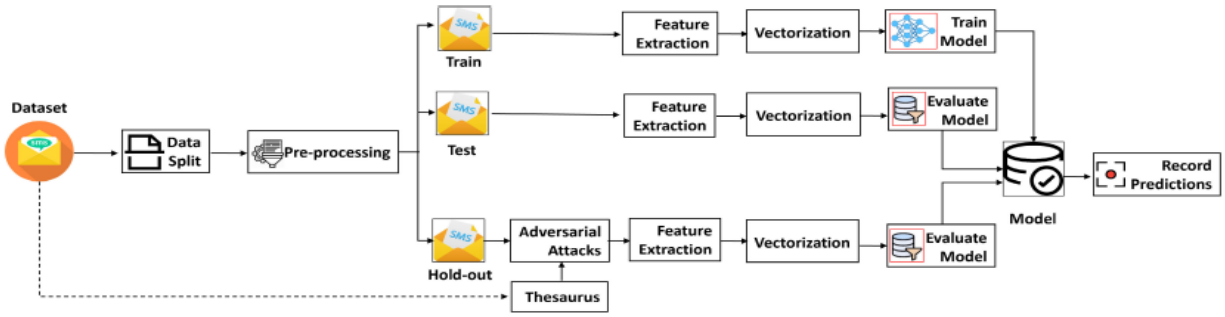


FIGURE 4. Overview of evaluation methodology.

TABLE 7. Lexical analysis of data set used for *training* classifiers.

Data	Avg Chars	Avg Words	Avg Sent	Avg Sent Length	Avg Word Length	Richness	ARI	Flesh Score
Overall	96.60	17.35	1.90	27.89	4.51	0.95	6.36	80.75
Ham	65.83	13.14	2.58	23.33	4.12	0.97	2.15	93.63
Spam	144.22	23.85	0.86	34.95	5.11	0.92	12.88	60.83

TABLE 8. Lexical analysis of data set used for *testing* classifiers.

Data	Avg Chars	Avg Words	Avg Sent	Avg Sent Length	Avg Word Length	Richness	ARI	Flesh Score
Overall	97.52	17.48	1.96	28.68	4.52	0.95	6.36	80.59
Ham	66.74	13.29	2.64	23.52	4.14	0.97	2.18	93.40
Spam	144.32	23.86	0.93	36.54	5.12	0.92	12.72	61.12

messages. The machine learning models' performance is assessed using this subset; Section V-D will go into more detail on this topic. The ML models are trained on the train set and then tested on the test set to determine how well they do on new, unknown data.

FEATURE EXTRACTION

Using non-semantic approaches such as a bag of words (BoW) [60] and n-grams (bigrams, trigrams) [61] and term frequency-inverse document frequency (TF-IDF) [62], we transform raw text into numerical characteristics. After that, we create a TF-IDF corpus from the whole BoW and n-grams dataset. Word order is not taken into account by BoW, but it does capture word frequency in the corpus. We solve this problem by counting the number of word pairings in an n-word sequence using n-grams.

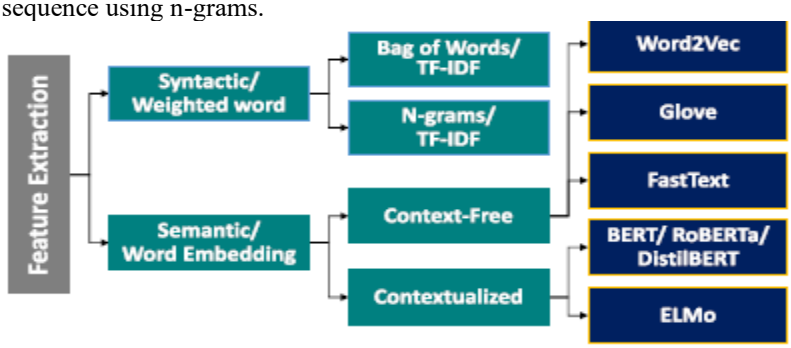


FIGURE 5. Feature extraction or representation techniques.

Nevertheless, a sparse matrix is still the outcome of even n-grams. Lastly, we evaluate the document and corpus terms' relevance using TF-IDF. With a high TF-IDF score, uncommon words stand out. This assignment is implemented using the scikit-learn module in Python.

EVASION TECHNIQUES

There are two main categories of evasion techniques: black-box and white-box, as well as targeted and non-targeted [22]. The attacker's knowledge of the targeted ML model determines which strategy is used. There is a current trend in using traditional black-box evasive strategies for evaluating SMS spam detection algorithms [7], [22]. Though it may come as a surprise, spammers may easily switch up their tactics to trick mobile consumers just as easily as they can internet users. For example, as shown in Table 9, many spam SMS campaigns are able to evade spam detection services by using the Punycode attack, a homograph technique (use non-English characters that visually resemble others) [81] commonly used in URL phishing, as well as in Smishing [82]. We investigate evasive tactics like the Punycode assault and others that are detailed below in our study. We want to see how well they work in avoiding machine learning-based

TABLE 9. Instances of evasion techniques (Spam messages) colored red. The changes in the text are also highlighted red.

Evas. Tech.	SMS Text
Actual SMS	You may get a $750 Economic Support Payment. For more details, Click https://xxx.info/covid/
Paraphrasing	A $750 Economic Support Compensation may be available to you. For further details https://xxx.info/covid/
Charswap	You Qmay gjt a $70 Economic Support Payment. For fruther detials https://XXXXX.info/COVID/?r=xxxx
EDA	You may get a $750 Support Economic Payment. For details https://XXXXX.info/COVID/?r=xxxx
Homograph	You may get a $750 Economic Support *payment* For more details, *Click* https://XXXXX.info/COVID/?r=xxxx
Spacing	You may get a $750 Economic Support p a y m e n t. For more details, C l i c k https://xxx.info/covid/
Hybrid	You may get a $750 Economic Support p a y m e n t. For more details, C1ick https://xxx.info/covid/

spam detection and anti-spam services. Our goal is to learn more about how these evasive strategies work so we can better understand how they might be used to avoid SMS spam detection. Our main emphasis is on black-box settings, which mimic real-life situations where the spammer is unaware of the procedures used to identify SMS spam. To accommodate for the wide variety of potential alterations to SMS text, we use a two-stage technique to produce evasive instances. We first use the concept of imperceptibility to save the original message's semantic content, and then we build a thesaurus of the 200 most common terms in spam communications. Table 11 displays the results of our lexical analysis, after which we create several evasive instances using methods like paraphrase, simple data augmentation [27], spacing, homograph (punycode), and hybrid evasive strategies.

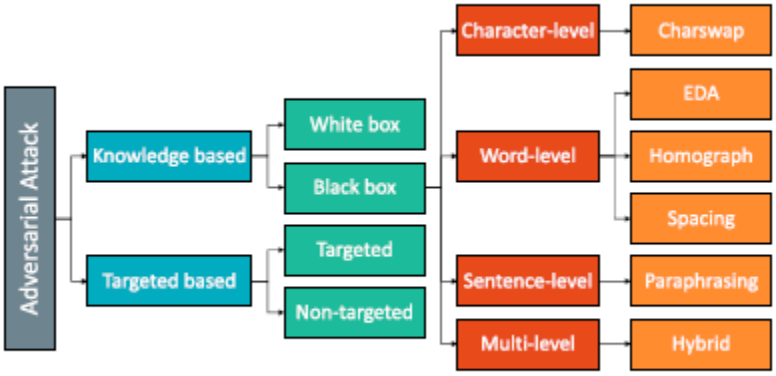


FIGURE 6. Taxonomy of evasion techniques employed in this study.

EVALUATION OF MACHINE LEARNING MODELS

We assess the ML models using a variety of metrics, such as how well they function, how much computing power they use, how easily they can be understood, and

TABLE 10. Explanation of evasion techniques employed in this study.

Technique	Explanation
Paraphrasing	A popular attack for fooling Spam filters by replacing vocables with synonyms or similar phrases, rendering the message unrecognizable to the model. We carried out this attack in two steps. In the first step, we restructure the sentences. In the second fold, we replaced all of the Spam keywords from the thesaurus with their synonyms in each SMS message.
EDA	This tactic generates adversarial examples by randomly removing, swapping, replacing, or adding a word's synonym in a sentence. We carried out this attack with the help of `TextAttack` [27]
Charswap	This attack randomly substitutes, deletes, inserts, and swaps adjacent characters of Spam keywords in the message. We utilize the `TextAttack` to generate adversarial examples of this type.
Homograph	This attack involves replacing the thesaurus's keywords found in SMS messages with their Punycode. We generated the homoglyphs for each character in the corresponding Spam word with the help of an online Punycode attack generator tool [83]
Spacing	In this technique, we add spaces between the characters of each spam keyword in the thesaurus.
Hybrid	This attack involves using a combination of spacing and visually similar alphanumeric characters to generate an adversarial example. The top 100 spam keywords from the thesaurus found in the SMS messages are replaced with spacing tactics, whereas other spam keywords are replaced with their alphanumeric equivalents (i.e., l is replaced with 1, a is replaced with @, etc.).

TABLE 11. Lexical analysis of data set used for *testing* classifiers (cf § VI-B) against different categories of evasive techniques (cf § V-D).

Data	Avg Chars	Avg Words	Avg Sent	Avg Word Length	Avg Sent Length	Richness	ARI	Flesh Score
Original	126.59	21.08	2.24	35.22	5.19	0.96	9.56	72.3
Para	148.25	24.32	2.76	55.81	5.29	0.94	9.19	67.90
EDA	125.88	20.78	2.23	35.01	5.24	0.96	9.72	70.87
Homo	126.63	20.99	2.24	35.24	5.22	0.95	9.67	84.79
Spacing	148.09	42.06	2.23	40.56	2.68	0.72	3.42	84.16
Charswap	126.22	20.97	2.21	35.26	5.20	0.97	9.64	73.59
Hybrid	145.25	38.94	2.23	39.95	2.89	0.79	3.55	84.21

sturdy construction. For this reason, we run two sets of experiments: (i) one to see how well the models distinguish between spam and real SMS, and (ii) another to see how well they hold up against evasive ways. For this unbalanced dataset, we assess the model's performance using the following metrics: Precision (PR), Recall (RE), Accuracy (ACC), and F1-score (F1) [84]. In this unbalanced setting, these measures provide a fair evaluation of the model's performance [85]. The following equations define them:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \quad (5)$$

If an SMS is spam, we mark it as a positive; if it is not, we mark it as a negative. We use the symbols P and N to denote the overall quantity of spam SMSes and ham SMSes, respectively. The number of correctly classified messages as spam is represented by TP in the above equations, while the number of incorrectly classified messages as spam is shown by FP. In the same vein, TN and FN are used to represent true negatives and false negatives, respectively. Tn shows how many SMSes were accurately labeled as Ham, whereas FN shows how many were incorrectly labeled as Ham.

SUMMARY

By capturing semantic information, automatically learning relevant features from the raw text data, and handling nonlinear connections, deep learning models with word embeddings outperformed shallow ML models in this experiment. Traditional machine learning algorithms struggle to decipher the hidden patterns in SMS texts due to the complex and non-linear word associations found inside. Alternatively, deep learning algorithms may extract more complex characteristics from their input data and learn complex non-linear word associations. Because of this flexibility, the DL models used in our experiment were able to accurately forecast the likelihood of SMS spam messages despite their complexity.

TABLE 12. Performance evaluation of *shallow* ML classifiers with the Super dataset (cf. Table 4). Here PR, RE, F1, ACC, and CM represent precision, recall, F1-score, accuracy, and confusion matrix, respectively. The top 3 models with the highest F1 and Acc are shown in bold.

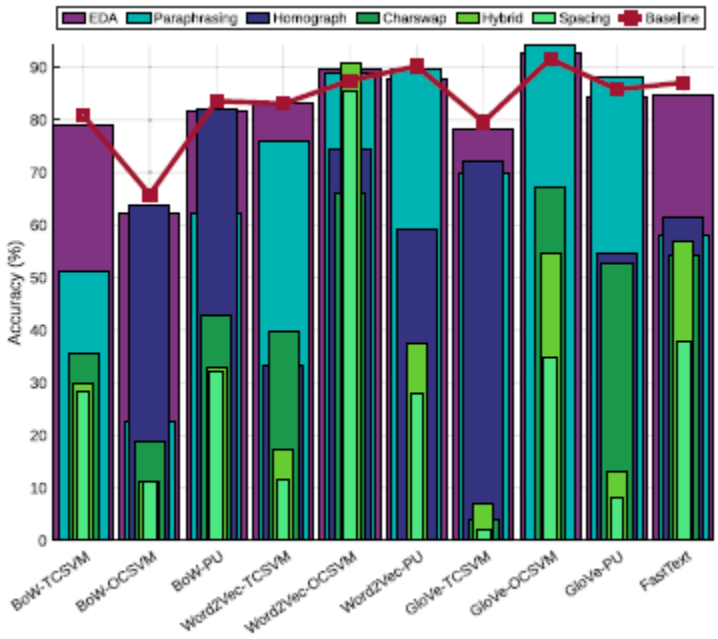
Feature Model	Classifier	Performance Metrics				
		PR	RE	ACC	F1	CM $\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix}$
BoW (TF-IDF)	TCSVM	99%	70%	74.2%	82%	$\begin{pmatrix} 1869 & 44 \\ 3267 & 7667 \end{pmatrix}$
	OCSVM	98%	50%	56.5%	66%	$\begin{pmatrix} 1809 & 109 \\ 5610 & 5619 \end{pmatrix}$
	PU	99%	81%	83.7%	90%	$\begin{pmatrix} 1871 & 47 \\ 2095 & 9134 \end{pmatrix}$
Bigram (TF-IDF)	TCSVM	100%	26%	37.3%	42%	$\begin{pmatrix} 1909 & 4 \\ 8052 & 2882 \end{pmatrix}$
	OCSVM	67%	27%	26.2%	38%	$\begin{pmatrix} 424 & 1494 \\ 8204 & 3025 \end{pmatrix}$
	PU	100%	38%	47.0%	55%	$\begin{pmatrix} 1911 & 7 \\ 6963 & 4266 \end{pmatrix}$
Trigram (TF-IDF)	TCSVM	100%	2%	16.6%	4%	$\begin{pmatrix} 1913 & 0 \\ 10711 & 213 \end{pmatrix}$
	OCSVM	67%	27%	26.2%	38%	$\begin{pmatrix} 424 & 1494 \\ 8204 & 3025 \end{pmatrix}$
	PU	100%	7%	20.4%	13%	$\begin{pmatrix} 1918 & 0 \\ 10462 & 767 \end{pmatrix}$
Word2Vec	**TCSVM**	**99%**	**98%**	**97.8%**	**99%**	$\begin{pmatrix} 1845 & 70 \\ 187 & 9353 \end{pmatrix}$
	OCSVM	90%	94%	85.7%	92%	$\begin{pmatrix} 743 & 1157 \\ 721 & 10503 \end{pmatrix}$
	PU	**97%**	**95%**	**93.1%**	**96%**	$\begin{pmatrix} 1599 & 316 \\ 592 & 10634 \end{pmatrix}$
GloVe	TCSVM	99%	81%	83.3%	89%	$\begin{pmatrix} 1800 & 116 \\ 2082 & 9144 \end{pmatrix}$
	OCSVM	90%	96%	87.2%	93%	$\begin{pmatrix} 676 & 1221 \\ 465 & 10760 \end{pmatrix}$
	PU	97%	93%	90.9%	95%	$\begin{pmatrix} 1564 & 352 \\ 841 & 10385 \end{pmatrix}$
fastText	**fastText**	**100%**	**92%**	**92.6%**	**95%**	$\begin{pmatrix} 1884 & 34 \\ 939 & 10300 \end{pmatrix}$

TABLE 13. Performance evaluation of DL classifiers with "Super Dataset" (cf. Table 4). The rows highlighted in red show the models with the highest accuracy and F1-score.

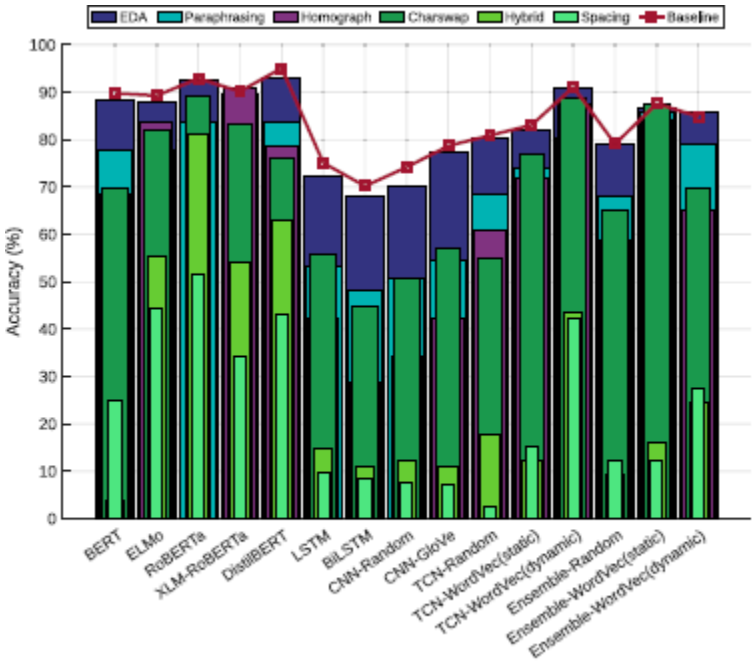
Classifier	Embedding	PR	RE	ACC	F1	CM $\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix}$
BERT	bert-base-uncased	100%	89%	90.3%	94%	$\begin{pmatrix} 1899 & 19 \\ 1252 & 9978 \end{pmatrix}$
ELMo	ELMO	100%	89%	90.4%	94%	$\begin{pmatrix} 1909 & 18 \\ 1241 & 9979 \end{pmatrix}$
RoBERTa	**roberta-base**	**99%**	**98%**	**97.4%**	**98%**	$\begin{pmatrix} 1819 & 99 \\ 242 & 10988 \end{pmatrix}$
XLM-RoBERTa	xlm-roberta-base	100%	95%	95.3%	97%	$\begin{pmatrix} 1908 & 10 \\ 613 & 10617 \end{pmatrix}$
DistilBERT	distilbert-base-uncased	100%	85%	87.1%	92%	$\begin{pmatrix} 1903 & 15 \\ 1686 & 9544 \end{pmatrix}$
LSTM	WE-Random	99%	95%	97.4%	97%	$\begin{pmatrix} 7880 & 69 \\ 274 & 4925 \end{pmatrix}$
BiLSTM	WE-Random	99%	96%	97.8%	97%	$\begin{pmatrix} 7879 & 70 \\ 219 & 4980 \end{pmatrix}$
CNN	WE-Random	98%	97%	97.7%	97%	$\begin{pmatrix} 7821 & 128 \\ 176 & 5023 \end{pmatrix}$
	GloVe (static)	98%	97%	97.8%	97%	$\begin{pmatrix} 7840 & 109 \\ 181 & 5018 \end{pmatrix}$
TCN	WE(Random)	99%	88%	89.2%	93%	$\begin{pmatrix} 1856 & 62 \\ 1361 & 9869 \end{pmatrix}$
	Word2Vec (static)	100%	88%	89.6%	94%	$\begin{pmatrix} 1894 & 24 \\ 1341 & 9889 \end{pmatrix}$
	Word2Vec (dynamic)	100%	88%	89.4%	93%	$\begin{pmatrix} 1891 & 27 \\ 1364 & 9866 \end{pmatrix}$
Ensemble (CNN-BiGRU)	WE-Random	100%	88%	89.6%	94%	$\begin{pmatrix} 1895 & 23 \\ 1345 & 9885 \end{pmatrix}$
	Word2Vec (static)	100%	92%	92.7%	96%	$\begin{pmatrix} 1900 & 18 \\ 938 & 10292 \end{pmatrix}$
	Word2Vec (dynamic)	99%	91%	91.6%	95%	$\begin{pmatrix} 1819 & 99 \\ 1004 & 10226 \end{pmatrix}$

ROBUSTNESS OF SHALLOW ML-BASED DETECTION TECHNIQUES

We find that spacing evasion is still the most successful method for avoiding most shallow ML models; eight models were completely dodged using this strategy. Furthermore, it should be mentioned that this method is very effective in eluding BoW/TF-IDF trained classifiers (such as TCSVM, OCSVM,



(a) Shallow ML Models.



(b) DL Models.

FIGURE 7. Analysis of robustness (in terms of accuracy) of ML models against adversarial attacks.

These models, which show the best performance (see §VI-A for details), include GloVe (TCSVM, OCSVM, PU), fastText, and the TCSVM-Word2Vec model. Table 14 shows that of the two models tested, Charswap is the second most effective approach. The OCSVM with Word2Vec model comes in at number two, with an accuracy of 66.1% (which is far lower than its baseline accuracy of 87.5%), while the PU with Word2Vec model fails miserably. All models, with the exception of those trained with BoW, are effectively targeted by the Homograph/Punycode assaults.

EVALUATION OF THE REAL-WORLD SMS ANTI-SPAM ECOSYSTEM

Next, we evaluate the robustness of the real-world SMS anti-spam ecosystem. Our analysis focuses on the two key

TABLE 14. Robustness/adversarial evaluation of *traditional (shallow)* ML classifiers on the holdout set. The cells highlighted in red show the most successful attack against the respective model.

Feature Model	Classifier	Baseline		Paraphrasing		EDA		Homograph		Spacing		Charswap		Hybrid	
		ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
Bow (TF-IDF)	TCSVM	80.9%	94%	51.1%	68%	79.1%	88%	-	-	**28.4%**	44%	35.6%	52%	29.8%	46%
	OCSVM	65.8%	79%	22.6%	37%	62.2%	77%	63.6%	78%	**11.1%**	20%	19%	32%	**11.1%**	20%
	PU	83.6%	91%	62.2%	77%	81.8%	90%	82.2%	90%	**32%**	48%	42.7%	60%	32.9%	49%
Word2Vec	TCSVM	83.1%	91%	76%	86%	83.1%	91%	33.3%	50%	**11.6%**	**21%**	39.6%	57%	17.3%	30%
	OCSVM	87.5%	93%	88.9%	94%	89.8%	95%	74.4%	85%	85.3%	92%	**66.1%**	**80%**	90.7%	95%
	PU	90.2%	95%	89.8%	95%	87.6%	93%	59.2%	74%	28%	44%	**0**	0%	37.3%	54%
GloVe	TCSVM	79.6%	89%	69.8%	82%	78.2%	88%	72.0%	84%	**2.2%**	**4%**	40%	57%	7.1%	13%
	OCSVM	91.6%	96%	94.2%	97%	92.8%	96%	-	-	**34.7%**	**51%**	67.1%	80%	54.7%	71%
	PU	85.8%	92%	88%	94%	84.4%	92%	54.7%	71%	**8%**	**15%**	52.9%	69%	13.3%	24%
fastText	fastText	87.1%	93%	58.2%	74%	84.9%	92%	61.3%	76%	**37.8%**	55%	54.2%	70%	56.9%	73%

TABLE 15. Robustness/adversarial evaluation of *deep* ML classifiers on the holdout set.

Classifier	Embedding	Baseline		Paraphrasing		EDA		Homograph		Spacing		Charswap		Hybrid	
		ACC	*F1*	*ACC*	*F1*	*ACC*	*F1*	*ACC*	*F1*	*ACC*	*F1*	*ACC*	*F1*	*ACC*	*F1*
BERT	bert-base-uncased	89.8%	95%	77.8%	88%	88.4%	94%	68.4%	81%	**24.9%**	**40%**	69.8%	82%	40.0%	57%
ELMo	ELMo	89.3%	94%	77.7%	88%	87.9%	94%	83.9%	91%	**44.6%**	**62%**	82.1%	90%	55.4%	71%
RoBERTa	roberta-base	92.9%	96%	83.6%	91%	92.4%	96%	-	-	**51.6%**	**68%**	89.3%	94%	81.3%	90%
XLM-RoBERTa	xlm-roberta-base	90.2%	95%	78.2%	88%	89.8%	95%	91.1%	95%	**34.2%**	**51%**	83.1%	91%	54.2%	70%
DistilBERT	distilbert-base-uncased	95.1%	97%	83.6%	91%	92.9%	96%	78.7%	88%	**43.1%**	**60%**	76.0%	86%	63.1%	77%
LSTM	WE-Random	75.1%	86%	53.3%	70%	72.4%	84%	42.2%	59%	**9.8%**	**18%**	56.0%	72%	14.7%	26%
BiLSTM	WE-Random	70.2%	83%	48.4%	65%	68.0%	81%	28.7%	45%	**8.4%**	**16%**	44.9%	62%	11.1%	20%
CNN	WE-Random	74.2%	85%	50.7%	67%	70.2%	83%	34.1%	51%	**7.6%**	**14%**	50.7%	67%	12.4%	22%
	GloVe (static)	78.7%	88%	54.7%	71%	77.3%	87%	42.2%	59%	**7.1%**	**13%**	56.9%	73%	11.1%	20%
TCN	TCN	80.9%	89%	68.4%	81%	80.4%	89%	60.9%	76%	**24.0%**	**39%**	55.1%	71%	17.8%	30%
	Word2Vec (static)	83.1%	91%	73.8%	85%	82.2%	90%	72.0%	84%	**15.1%**	**26%**	76.9%	87%	12.4%	22%
	Word2Vec (dynamic)	91.1%	95%	80.4%	89%	90.7%	95%	87.6%	93%	**42.2%**	**59%**	88.9%	94%	43.6%	61%
Ensemble (CNN-BiGRU)	WE-Random	79.1%	88%	68.0%	81%	79.1%	88%	58.6%	74%	12.4%	22%	64.9%	79%	**9.3%**	**17%**
	Word2Vec (static)	87.6%	93%	85.8%	92%	85.8%	92%	84.0%	91%	**12.4%**	**22%**	87.6%	93%	16.0%	28%
	Word2Vec (dynamic)	84.9%	92%	79.1%	88%	85.8%	92%	65.3%	79%	27.6%	43%	69.8%	82%	**24.4%**	**39%**

TABLE 16. Concept Drift Analysis of *shallow* or *traditional* ML classifiers. Based on ACC and F1 in both the folds, the row in green highlights the most successful model.

Feature	Classifier	Performance Metrics - First Fold					Performance Metrics - Second Fold				
		PR	RE	ACC	F1	CM($\begin{smallmatrix}TN & FP\\ FN & TP\end{smallmatrix}$)	PR	RE	ACC	F1	CM($\begin{smallmatrix}TN & FP\\ FN & TP\end{smallmatrix}$)
BoW (TF-IDF)	TCSVM	100%	5%	15.0%	10%	$\begin{pmatrix}2869 & 1\\ 23740 & 1310\end{pmatrix}$	7%	93%	73.1%	13%	$\begin{pmatrix}26891 & 10114\\ 54 & 757\end{pmatrix}$
	OCSVM	95%	28%	34.4%	44%	$\begin{pmatrix}2511 & 359\\ 17956 & 7094\end{pmatrix}$	4%	5%	95.4%	4%	$\begin{pmatrix}36053 & 952\\ 774 & 37\end{pmatrix}$
	PU	99%	57%	61.4%	73%	$\begin{pmatrix}2755 & 115\\ 10654 & 14396\end{pmatrix}$	51%	61%	97.9%	55%	$\begin{pmatrix}36521 & 484\\ 316 & 495\end{pmatrix}$
Bigram (TF-IDF)	TCSVM	100%	1%	11.0%	2%	$\begin{pmatrix}2870 & 0\\ 24847 & 203\end{pmatrix}$	6%	78%	73.1%	11%	$\begin{pmatrix}26999 & 10006\\ 178 & 633\end{pmatrix}$
	OCSVM	84%	57%	52.0%	68%	$\begin{pmatrix}137 & 2733\\ 10661 & 14389\end{pmatrix}$	1%	21%	24.3%	1%	$\begin{pmatrix}9015 & 27990\\ 638 & 173\end{pmatrix}$
	PU	100%	0%	10.3%	0%	$\begin{pmatrix}2870 & 0\\ 25045 & 5\end{pmatrix}$	75%	6%	97.9%	11%	$\begin{pmatrix}36989 & 16\\ 762 & 49\end{pmatrix}$
Trigram (TF-IDF)	TCSVM	100%	0%	10.3%	0%	$\begin{pmatrix}2870 & 0\\ 25050 & 1\end{pmatrix}$	22%	16%	97.0%	19%	$\begin{pmatrix}36545 & 460\\ 679 & 132\end{pmatrix}$
	OCSVM	89%	97%	87.1%	93%	$\begin{pmatrix}3 & 2867\\ 737 & 24313\end{pmatrix}$	2%	90%	2.3%	3%	$\begin{pmatrix}290 & 36759\\ 68 & 580\end{pmatrix}$
	PU	100%	0%	10.3%	0%	$\begin{pmatrix}2870 & 0\\ 25050 & 1\end{pmatrix}$	100%	0%	97.9%	0%	$\begin{pmatrix}37005 & 0\\ 810 & 1\end{pmatrix}$
Word2Vec	TCSVM	100%	51%	56.2%	68%	$\begin{pmatrix}2840 & 11\\ 12218 & 12829\end{pmatrix}$	14%	92%	88.0%	25%	$\begin{pmatrix}32177 & 4425\\ 68 & 743\end{pmatrix}$
	OCSVM	92%	98%	90.2%	95%	$\begin{pmatrix}660 & 2251\\ 493 & 24673\end{pmatrix}$	4%	79%	67.4%	8%	$\begin{pmatrix}24019 & 11756\\ 144 & 546\end{pmatrix}$
	PU	97%	97%	94.3%	97%	$\begin{pmatrix}2134 & 794\\ 820 & 24348\end{pmatrix}$	91%	80%	99.5%	85%	$\begin{pmatrix}36468 & 57\\ 135 & 555\end{pmatrix}$
GloVe	TCSVM	100%	32%	39.3%	49%	$\begin{pmatrix}2846 & 8\\ 16924 & 8123\end{pmatrix}$	9%	92%	80.0%	17%	$\begin{pmatrix}29199 & 7424\\ 65 & 746\end{pmatrix}$
	OCSVM	93%	97%	90.4%	95%	$\begin{pmatrix}947 & 1965\\ 722 & 24445\end{pmatrix}$	5%	79%	70.9%	9%	$\begin{pmatrix}25269 & 10448\\ 144 & 546\end{pmatrix}$
	PU	96%	95%	92.4%	96%	$\begin{pmatrix}2044 & 888\\ 1236 & 23932\end{pmatrix}$	84%	80%	99.4%	82%	$\begin{pmatrix}36442 & 103\\ 137 & 553\end{pmatrix}$
fastText	fastText	100%	54%	58.7%	70%	$\begin{pmatrix}2937 & 12\\ 11598 & 13573\end{pmatrix}$	22%	89%	93.9%	35%	$\begin{pmatrix}34714 & 2212\\ 78 & 612\end{pmatrix}$

elements within this ecosystem: (i) well-known messaging apps that filter SMS spam, and (ii) third-party APIs that provide anti-spam solutions for SMS.Just as in the evaluation of machine learning models in Section VI, our study focuses solely on content-based SMS filtering. We zero in on two main areas to evaluate the antispam ecosystem's efficacy and robustness. I will start by saying,

TABLE 17. Concept Drift Analysis of *deep* ML classifiers.

Classifier	Embedding	Performance Metrics - First Fold					Performance Metrics - Second Fold				
		PR	RE	ACC	F1	CM $\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix}$	PR	RE	ACC	F1	CM $\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix}$
BERT	bert-base-uncased	100%	0%	10.5%	0%	$\begin{pmatrix} 2950 & 0 \\ 25171 & 1 \end{pmatrix}$	14%	94%	89.1%	24%	$\begin{pmatrix} 32846 & 4079 \\ 38 & 652 \end{pmatrix}$
ELMo	ELMO	100%	9%	14.8%	17%	$\begin{pmatrix} 937 & 0 \\ 13284 & 1368 \end{pmatrix}$	4%	99%	59.3%	8%	$\begin{pmatrix} 21693 & 15356 \\ 4 & 643 \end{pmatrix}$
RoBERTa	roberta-base	100%	0%	10.5%	0%	$\begin{pmatrix} 2950 & 0 \\ 25171 & 1 \end{pmatrix}$	15%	97%	89.6%	25%	$\begin{pmatrix} 33026 & 3899 \\ 21 & 669 \end{pmatrix}$
XLM-RoBERTa	xlm-roberta-base	100%	49%	53.9%	65%	$\begin{pmatrix} 2935 & 15 \\ 12947 & 12224 \end{pmatrix}$	31%	94%	95.8%	46%	$\begin{pmatrix} 35403 & 1522 \\ 41 & 669 \end{pmatrix}$
DistilBERT	distilbert-base-uncased	100%	39%	45.1%	56%	$\begin{pmatrix} 2939 & 11 \\ 15432 & 9739 \end{pmatrix}$	14%	94%	89.5%	25%	$\begin{pmatrix} 33012 & 3913 \\ 38 & 652 \end{pmatrix}$
LSTM	WE-Random	99%	14%	22.7%	24%	$\begin{pmatrix} 2925 & 25 \\ 21725 & 3446 \end{pmatrix}$	4%	98%	57.3%	8%	$\begin{pmatrix} 20893 & 16032 \\ 13 & 677 \end{pmatrix}$
BiLSTM	WE-Random	99%	9%	18.8%	17%	$\begin{pmatrix} 2931 & 19 \\ 22827 & 2344 \end{pmatrix}$	3%	98%	49.8%	7%	$\begin{pmatrix} 18050 & 18875 \\ 11 & 679 \end{pmatrix}$
CNN	WE(Random)	99%	40%	45.6%	57%	$\begin{pmatrix} 2856 & 94 \\ 15202 & 9969 \end{pmatrix}$	4%	98%	61.2%	8%	$\begin{pmatrix} 22363 & 14562 \\ 14 & 676 \end{pmatrix}$
	GloVe (static)	99%	40%	45.9%	57%	$\begin{pmatrix} 2854 & 96 \\ 15113 & 10058 \end{pmatrix}$	5%	97%	69.3%	10%	$\begin{pmatrix} 25388 & 11537 \\ 21 & 669 \end{pmatrix}$
TCN	WE(Random)	100%	39%	45.6%	56%	$\begin{pmatrix} 2941 & 9 \\ 15293 & 9878 \end{pmatrix}$	15%	94%	90.0%	26%	$\begin{pmatrix} 33189 & 3736 \\ 43 & 647 \end{pmatrix}$
	Word2Vec (static)	100%	37%	43.8%	54%	$\begin{pmatrix} 2944 & 6 \\ 15807 & 9364 \end{pmatrix}$	100%	100%	99.9%	100%	$\begin{pmatrix} 2927 & 23 \\ 12 & 25159 \end{pmatrix}$
	Word2Vec (dynamic)	100%	30%	37.0%	46%	$\begin{pmatrix} 2944 & 6 \\ 17697 & 7474 \end{pmatrix}$	8%	97%	80.6%	15%	$\begin{pmatrix} 29655 & 7270 \\ 24 & 666 \end{pmatrix}$
Ensemble (CNN-BiGRU)	WE-Random	100%	0%	10.5%	0%	$\begin{pmatrix} 2950 & 0 \\ 25171 & 1 \end{pmatrix}$	17%	92%	91.7%	29%	$\begin{pmatrix} 33869 & 3056 \\ 53 & 637 \end{pmatrix}$
	Word2Vec (static)	100%	0%	10.5%	0%	$\begin{pmatrix} 2950 & 0 \\ 25171 & 1 \end{pmatrix}$	25%	94%	94.6%	39%	$\begin{pmatrix} 34932 & 1993 \\ 43 & 647 \end{pmatrix}$
	Word2Vec (dynamic)	100%	0%	10.5%	0%	$\begin{pmatrix} 2950 & 0 \\ 25171 & 1 \end{pmatrix}$	19%	94%	92.5%	31%	$\begin{pmatrix} 34144 & 2781 \\ 42 & 648 \end{pmatrix}$

Here, we take a look at how effective anti-spam solutions in the real world are in catching spam before it reaches its targets. In order to do this, we use real-world scenarios to test all messages in the hold-out split using SMS anti-spam tools and services. With this stage, we can evaluate how well these methods detect and filter out spam communications. Second, we look at how well anti-spam SMS applications and services do when faced with evasive methods. To do this, we run tests with customized SMS instances that use different evasive strategies (see to §V-D). The widely-used SMS anti-spam applications and services are tested by sending these malicious instances to them. We then evaluate their ability to identify and mitigate the effects of these evasive approaches. This extensive analysis sheds light on the robustness and efficacy of the SMS anti-spam ecosystem. We help shed light on the pros and cons of these applications and services for real-world spam avoidance by testing them with both typical spam messages and customized versions meant to avoid detection.

TABLE 18. Text messaging apps subject to our evaluation.

App	Platform	Downloads	CBF	Mechanism
Google Messages	Android	1B+	No	AI/Rules
Key	Android	1M+	Yes	Rules
Trend Micro Check	Android	100K+	Yes	AI/Rules/ ReputationCheck
iOS Message	iOS	-	No	-
VeroSMS	iOS	-	Yes	AI/Rules
SMS Shield	iOS	-	Yes	AI/Rules

TABLE 19. Robustness and Performance evaluation of *Text Messaging Apps* on the holdout set.

Apps	Baseline	Parap.	EDA	Homog.	Spac.	Chars.	Hyb.
			% Spam Detected				
Google Messages	9.8%	-	-	-	-	-	-
Trend Micro Check	63.6%	47.8%	63.6%	25.8%	23.6%	56%	25.8%
Key	75.6%	69.3%	76.9%	32.4%	40.4%	58.7%	44.9%
iOS Message	0	-	-	-	-	-	-
VeroSMS	84.4%	73.3%	84.4%	63.1%	68.44%	79.6%	68.4%
SMS Shield	83.6%	71.6%	83.6%	60.9%	61.8%	76.4%	67.1%

TABLE 20. Robustness and Performance evaluation of *Anti-spam Services* on the holdout set.

Service	Baseline	Parap.	EDA	Homog.	Spac.	Chars.	Hyb.
			% Spam Detected				
OOPSpam	33.8%	-	33.8%	10.7%	12.9%	23.6%	13.8%
Zyla Text Spam Checker	56.4%	46.7%	57.3%	25.8%	38.7%	43.6%	40.4%
Plino	80.9%	83.1%	88.0%	69.3%	69.3%	74.2%	71.6%

DISCUSSION & FUTURE RESEARCH DIRECTIONS

Here we provide a concise overview of the main points from our analysis of the antispam ecosystem and ML models on their performance and resilience. D. DL Models Are More Effective Than Traditional ML Models for Detecting SMS Spam We put 31 ML models through their paces in terms of training and performance evaluations, as well as robustness (the ability to withstand adversarial assaults). Even while the majority of standard ML models perform well when tested, only six of sixteen managed an 85% recall or greater, suggesting that they were not very good at detecting spam (see to Table 12 for details). The fifteen deep ML models, on the other hand, all hit 85% or above. recall, and each of them achieved an F1-score of 90% or more (see to Table 13 for details). This demonstrates how much better DL is than Conventional machine learning for the purpose of detecting spam in SMS messages. B. DL Models Outperform Traditional ML Models in Terms of Resilence Tables 14 and 15 illustrate the results of the robustness evaluation, which clearly demonstrate that deep ML models are less affected by adversarial instances compared to shallow ML models. Evidently, deep ML models are stronger in this regard. In addition, the findings show that deep learning models based on transformers may greatly enhance the resilience of the data. When compared to one-class learning, PU learning is on par with two-class learning. With the exception of Trigram, PU learning achieves better results than OCSVM in performance (see §VI-A) and adversarial (see §VI-B) evaluations, including a high F1-score and accuracy. When compared to fastText and TCSVM, its performance is on par. Because of this, PU learning may be a good substitute when working with tiny datasets or just positive examples.D. Deficiency in Generalization Skills Most models with fresh SMS campaigns perform poorly according to the findings of the most recent SMS spam messages in the idea drift study (refer to §VII). Most of the anti-spam text applications in §VIII had poor baseline performance (just two apps had an 80% spam detection rate) on original spam messages, which implies that their anti-spam algorithms aren't up to snuff with the evolving nature of SMS spam. This emphasizes the present dearth of research into methods for dealing with the notion drift issue in SMS spam screening.

CONCLUSION

We developed and analyzed a large new SMS dataset to provide insight on the evolving features of SMS spam. To find out how well various machine learning models and the anti-spam ecosystem identify SMS spam, we utilized this dataset as a benchmark. All of the models based on machine learning were able to correctly identify ham

SMSes, or authentic SMS messages. Nevertheless, out of all the anti-spam text applications and deep learning models tested, only a few managed to achieve an accuracy score of 80% or above when it came to spam message classification. Our examination of the SMS anti-spam ecosystem and the machine learning model reveals the shortcomings of existing anti-spam advancements and suggests future research paths. We contend that SMS spam is still a major problem and that further study is needed to create systems that can protect the public from SMS spam by tackling the evasion strategies used by spammers. You can get our dataset and analysis at https://github.com/smspamresearch/spstudy. We hope that this will draw attention to the shortcomings of existing anti-spam procedures and encourage further study into the development of better detection algorithms.

REFERENCES

[1] J. Buchanan and A. J. Grant, ''Investigating and prosecuting Nigerian fraud,'' *U.S. Att'ys Bull.*, vol. 49, pp. 39–47, Nov. 2001.
[2] L. Zhang, J. Zhu, and T. Yao, ''An evaluation of statistical spam filtering techniques,'' *ACM Trans. Asian Lang. Inf. Process.*, vol. 3, no. 4, pp. 243–269, Dec. 2004.
[3] G. L. Wittel and S. F. Wu, ''On attacking statistical spam filters,'' in *Proc. CEAS*, 2004.
[4] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, ''Contributions to the study of SMS spam filtering: New collection and results,'' in *Proc. 11th ACM Symp. Document Eng.*, Sep. 2011, pp. 259–262.
[5] T. Almeida, J. M. Hidalgo, and T. Silva, ''Towards SMS spam filtering: Results under a new dataset,'' *JiSS*, vol. 2, no. 1, pp. 1–18, 2013.
[6] M. Gupta, A. Bakliwal, S. Agarwal, and P. Mehndiratta, ''A comparative study of spam SMS detection using machine learning classifiers,'' in *Proc. 11th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2018, pp. 1–7.
[7] S. Rojas-Galeano, ''Using BERT encoding to tackle the mad-lib attack in SMS spam detection,'' 2021, *arXiv:2107.06400*.
[8] FCC. (2022). *The Top Text Scams of 2022*. Accessed: Oct. 8, 2023. [Online]. Available: https://www.ftc.gov/news-events/data-visualizations /data-spotlight/2023/06/iykyk-top-text-scams-2022
[9] ACCS. (2022). *Accs Scam Statistics*. [Online]. Available: https://www. scamwatch.gov.au/scam-statistics
[10] M. A. Abid, S. Ullah, M. A. Siddique, M. F. Mushtaq, W. Aljedaani, and F. Rustam, ''Spam SMS filtering based on text features and supervised machine learning techniques,'' *Multimedia Tools Appl.*, vol. 81, no. 28, pp. 39853–39871, Nov. 2022.
[11] I. Ahmed, R. Ali, D. Guan, Y.-K. Lee, S. Lee, and T. Chung, ''Semisupervised learning using frequent itemset and ensemble learning for SMS classification,'' *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1065–1073, Feb. 2015.
[12] C. Oswald, S. E. Simon, and A. Bhattacharya, ''SpotSpam: Intention snalysis-driven SMS spam detection using BERT embeddings,'' *ACM Trans. Web*, vol. 16, no. 3, pp. 1–27, Aug. 2022.
[13] S. Y. Yerima and A. Bashar, ''Semi-supervised novelty detection with one class SVM for SMS spam detection,'' in *Proc. 29th Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Jun. 2022, pp. 1–4.
[14] S. Tang, X. Mi, Y. Li, X. Wang, and K. Chen, ''Clues in tweets: Twitterguided discovery and analysis of SMS spam,'' in *Proc. ACMSIGSAC Conf.omput. Commun. Secur.*, Nov. 2022, pp. 2751–2764.
[15] A. van der Schaaf, C.-J. Xu, P. van Luijk, A. A. van't Veld, J. A. Langendijk, and C. Schilstra, ''Multivariate modeling of complications with data driven variable selection: Guarding against overfitting and effects of data set size,'' *Radiotherapy Oncol.*, vol. 105, no. 1, pp. 115–121, Oct. 2012.
[16] T. Xia and X. Chen, ''A discrete hidden Markov model for SMS spam detection,'' *Appl. Sci.*, vol. 10, no. 14, p. 5011, Jul. 2020.
[17] L. Duan, N. Li, and L. Huang, ''A new spam short message classification,'' in *Proc. 1st Int.Workshop Educ. Technol. Comput. Sci.*, 2009, pp. 168–171.
[18] S. M. Abdulhamid, M. S. A. Latiff, H. Chiroma, O. Osho, G. Abdul-Salaam, A. I. Abubakar, and T. Herawan, ''A review on mobile SMS spam filtering techniques,'' *IEEE Access*, vol. 5, pp. 15650–15666, 2017.
[19] A. Narayan and P. Saxena, ''The curse of 140 characters: Evaluating the efficacy of SMS spam detection on Android,'' in *Proc. 3rd ACMWorkshop Secur. Privacy Smartphones Mobile Devices*, Nov. 2013, pp. 33–42.
[20] A. A. Al-Hasan and E.-S.-M. El-Alfy, ''Dendritic cell algorithm for mobile phone spam filtering,'' *Proc. Comput. Sci.*, vol. 52, pp. 244–251, Jan. 2015.
[21] J. Li, S. Ji, T. Du, B. Li, and T.Wang, ''TextBugger: Generating adversarial text against real-world applications,'' 2018, *arXiv:1812.05271*.
[22] W.Wang, R.Wang, L.Wang, Z.Wang, and A. Ye, ''Towards a robust deep neural network in texts: A survey,'' 2019, *arXiv:1902.07285*.

[23] A. Huq and M. Tasnim Pervin, ‘‘Adversarial attacks and defense on texts: A survey,’’ 2020, *arXiv:2005.14108*.
[24] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, ‘‘Black-box generation of adversarial text sequences to evade deep learning classifiers,’’ in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2018, pp. 50–56.
[25] (2023). *Action Fraud*. Accessed: Oct. 6, 2023. [Online]. Available: https://www.actionfraud.police.uk/
[26] (2012). *UCI Machine Learning Repository—SMS Spam Collection Data Set*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/sms
[27] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, ‘‘TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP,’’ in *Proc. Conf. Empirical Methods Natural Lang. Processing: Syst. Demonstrations*, 2020, pp. 119–126.
[28] F. Salahdine and N. Kaabouch, ‘‘Social engineering attacks: A survey,’’ *Future Internet*, vol. 11, no. 4, p. 89, Apr. 2019.
[29] A. Costello, *Punycode: A Bootstring Encoding of Unicode for Internationalized Domain Names in Applications (IDNA)*, document RFC 3492, 2003.
[30] J. M. Gómez Hidalgo, G. C. Bringas, E. P. Sánz, and F. C. García, ‘‘Content based SMS spam filtering,’’ in *Proc. ACMSymp. Document Eng.*, Oct. 2006, pp. 107–114.