ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

Exploring Deceptive Methods in SMS Spam Filtering: A Multi-Model Comparison

[1] M. Nageshwarappa, [2] P. Akanksha,

[1]Assistant Professor, Megha Institute of Engineering & Technology for Women, Ghatkesar.
[2] MCA Student, Megha Institute of Engineering & Technology for Women, Ghatkesar.

Article Info

Received: 30-04-2025 Revised: 16-06-2025 Accepted: 28-06-2025

ABSTRACT

There has to be study into creating systems that can successfully handle the evasive tactics employed by spammers since SMS spam is still a big problem. If we want to protect people from the harmful effects of SMS spam, we need to fund studies like this. Our goal in doing this research is to shed light on the difficulties that are currently present in the field of SMS spam identification and filtering. Our new SMS dataset has over 68,000 messages, 61% of which are valid (ham) SMS and 39% of which are spam, in an effort to tackle these issues. The most publicly accessible dataset on SMS spam to this point is this one, which we are releasing for more study. We use a longitudinal study of spam evolution to define 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 test how well the most popular anti-spam services and current models for detecting SMS spam hold up against the tactics used by spammers to avoid detection. According to our research, most anti-spam systems and methods that rely on shallow machine learning perform poorly when it comes to correctly identifying spam SMS messages. 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.

INDEX TERMS

spam dataset, anti-spam services, evasive approaches, machine learning robustness analysis, SMS spam evolution, and methods for detecting spam.

I. INTRODUCTION

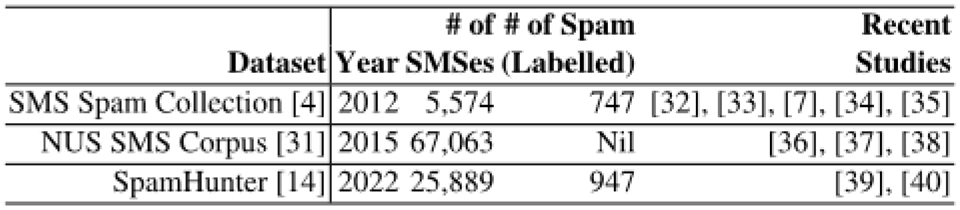
Research on SMS spam1 identification has been ongoing for about twenty years [1, 2, 3, 4, 5, 6, 7], yet it remains a significant challenge for our contemporary digital society. With a projected USD$330 million (more than twice the 2021 total) wasted in the US to SMS fraudsters in 2022, the amount of SMS spam has reached worrisome levels in recent years [8]. Similarly, yearly losses almost doubled from 175 million Australian dollars in 2020 to 323 million Australian dollars in 2021, according to the Australian Competition and Consumer Commission's (ACCC) ScamWatch committee [9]. There were 67,180 instances of SMS fraud in 2021, up from 32,337 the previous year; in February 2022 alone, more than 8,835 SMS scams were recorded, making it the most common means of scam delivery. Our research in this area has shown four significant obstacles to reducing SMS spam: Access to Information: Due to a lack of big, real-world, annotateddatasets, developing algorithms to identify SMS spam is quite difficult. Previous research[4,7,10,11,12,13] often makes use of small, unbalanced datasets that are many years old and include just a few hundred spam messages. We are aware of two up-to-date SMS spam datasets: SMS Spam Collection [4] and SpamHunter Dataset [14].

The SpamHunter Dataset has 947 annotated spam messages, whereas the out-of-date SMS Spam Collection (released in 2012) only has 747. Due to the lack of up-to-date and comprehensive data on SMS spam texts casts doubt on their effectiveness in preventing SMS spam. This constraint reduces the model's generalizability and its performance on unknown data, and thus raises the likelihood of overfitting [15]. The lack of a standardized benchmark dataset for thorough comparisons [18], [21] has caused research in the field of SMS spam detection to remain fragmented, despite the many strategies that have been suggested for this purpose [5, 16, 17, 18, 19, 20]. The performance of these models is not well described due to the absence of consistent datasets, which makes it difficult to assess the effectiveness of various suggested detection approaches. Resilience in the Face of Evasive Attacks: The inadequacy of current anti-spam ecosystems and suggested machine learning (ML) models to withstand fraudsters' evasion strategies is another big obstacle to reducing SMS spam. Spammers are always coming up with new, sneaky ways to get past spam filters. New research using advances in Natural Language Processing (NLP) has shown that evasive techniques can fool different ML-based spam models, which means that the threat of SMS spam is on the rise, even though traditional ML and deep learning models may improve spam detection. (7), (22) and (23), and (24). There is a lack of study on how well the SMS anti-spam ecosystem counters evasive approaches, which is concerning since evasive techniques pose a significant threat to SMS spam models. Additionally, the most recent dangers stemming from the explosion of Web-enabled services for bulk SMS messaging are disregarded as the machine learning models presented in the literature are assessed using traditional evasive methods.

Text Message Spam Databases Research on SMSspamdetection was made possible by the availability of several SMS datasets. Nevertheless, there aren't a ton of spam messages in these databases. An overview of the most prominent SMS spam datasets, beginning in 2012 and ending in 2022, is shown in table1. Since its 2012 release, the SMS Spam Collection [4] has been grossly skewed and contains spam messages from before 2010. It is also grossly out of date. There are 5,574 messages altogether, with just 747 being spam. The spam messages were collected from two sources: the now-defunct Grumbletext website (a UK forum) and the SMS Spam Corpus v.0.1 Big [30]. With the most current update being on March 9, 2015, the National University of Singapore (NUS) SMS Corpus [31] has 67,093 SMS messages. Not long ago, someone suggested the "SpamHunter" framework as a way to gather and extract spam data from publicly published SMS images on Twitter [14]. SpamHunter was used to collect and publish 25,889 tweets in various languages from 2018 to 2022. A large number of benign and awareness messages were mistakenly crawled and included in the SMS spam dataset produced by the SpamHunter framework, which is a shame since the method is innovative. The collection also includes many instances of duplicate messages and optical character recognition mistakes. The ML model might be severely misled if fed this noise, necessitating a human review to eliminate noise and mistakes. Welcome to 4grumbletext.co.uk!They removed 53 non-spam communications that were mistakenly included in the dataset after randomly sampling and reviewing 1,000 messages from the dataset for their own investigation. In this work, we provide a new massive SMS spam dataset that was compiled from several sources, such as public datasets, fraud observatories, Twitter (formerly known as X), and a group of specialists that manually labeled a huge number of SMS messages.

II. RELATED WORK

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]

B. SCAM DETECTION BY SMS A number of traditional ML and DL methods have been suggested to combat the problem of SMS spam. While Almeida et al. discovered that SVM outperformed other ML

ISSN:0976-0172

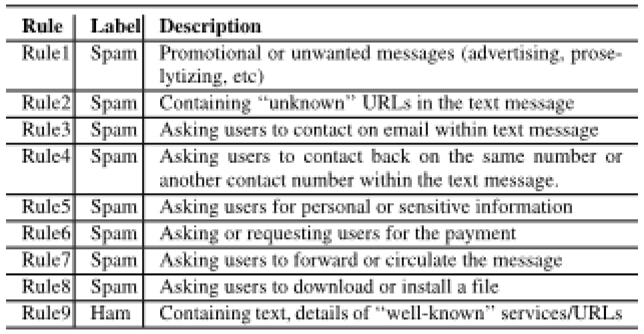
Journal of Bioscience And Technology
www.jbstonline.com

classifiers when evaluating word frequency alone, they neglected to assess DL models in their comparisons [4], [41]. In a similar vein, Gupta et al.[6] examined one DL classifier and seven traditional ML classifiers using TF-IDF features alone. These classifiers included CNN and SVM. The researchers Royetal.[42] discovered that when it came to SMS spam identification employing different model stack architectures, CNN and LSTM performed better than the usual two-class classifiers. But they didn't test any cutting-edge transformer-based models; they stuck to old-fashioned two-class algorithms. Although Jain et al. [43] examined LSTM with Word2Vec, they did not employ contextualized word embeddings or state-of-the-art transformer-based models. Instead, they just used LSTM with Word2Vec. One side of the coin is that research on SMS spam identification has been disjointed due to earlier studies using just a handful of characteristics and models. Classifiers that use PU learning and one-class approaches, on the other hand, have gotten less attention. Our work stands out because no other research has examined such a wide variety of machine learning models before. We evaluate their effectiveness and resilience in fighting SMS spam by using detailed syntactic and semantic aspects. What makes our work unique and important is that no prior study has used modern datasets to train or evaluate classifiers.

III. DATA COLLECTION AND AUGMENTATION

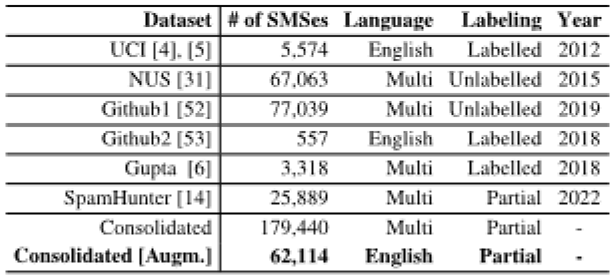
We gather and enhance a dataset of SMS spam messages that spans more than ten years in order to tackle the first two obstacles (see to §I). Collecting Data In order to find, gather, and combine publically accessible SMS datasets, we ran an extensive survey. We used search phrases like "SMS," "SMSmessages," "SMSdataset," "TextMessages," and "ShortMessageService" to scour the internet and GitHub projects for pertinent information. Only findings, GitHub repositories, and articles citing publicly accessible datasets were considered for inclusion in our filter. Our work collected 179,440 SMS occurrences from a variety of public and free-for-research sources across numerous languages. Table 3 presents the detailed sources. We also searched Twitter specifically for mentions of SMS spam in addition to the previously indicated sources. Screenshots or photos of these tweets were made public. To make sure we covered more ground than just the SpamHunter dataset, we crawled and gathered these tweets from 2012–2017 and again from 2022–2023. Scamwatch and Action Fraud are two websites that we used to get public photographs and screenshots of victims' reported SMS frauds.In addition, we asked college students to help us out by sending any text messages they got to a certain number we provided. The time frame for this endeavor was from 2020–2023. The contributors were made aware that their work will be available to the public. Through these data gathering efforts, we were able to get a total of 3,712 SMS messages (1,387 spam and 2,325 ham) from volunteers, 203 spam SMS messages from observatories, and 1,130 spam messages from Twitter. We have included these freshly acquired communications into our corpus; they were not previously reported.

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



Rule	Label	Description
Rule1	Spam	Promotional or unwanted messages (advertising, proselytizing, 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.

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

Dataset	# of SMSes	# of Spam SMSes 2012-2017	2018-202
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**

dataset contains SMS from several sources and is the most comprehensive and varied. We significantly increased the amount of tagged spam messages compared to earlier research projects, contributing 2,920 messages (for comparison, see Table 1).

IV. EVOLUTION OF SMS SPAM: AN ANALYSIS OF CHANGING CHARACTERISTICS

Using our longitudinal dataset that covers the years 2012–2023, we will examine the features of SMS spam as well as the methods used by spammers to get a better understanding of the evolution of this kind of spam. Based on the publishing 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. Our research provides new insight into the approaches used by spammers to evade detection and reach their intended audiences. Section A: Lexeme Analysis of Short Message Service To go a step further, we check our dataset of SMSs for semantic, grammatical, and spelling errors. 1) Errors in Spelling One strategy that spammers use to evade spam filters is intentional misspellings [54]. We used the pyspellchecker module in Python and implemented a mistake_to_ tokens_ratio to objectively measure spelling mistakes in SMSes of different lengths in order to put a numerical value on this issue. The average mistake-to-tokens ratio rose significantly from 18% (2012–2017) to 33% (2018–2023), according to our data (see Figure 1). Spammers' constant attempts to avoid spam detection are highlighted by the increased usage of spelling mistakes, even though auto-correction tools are prevalent in current 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

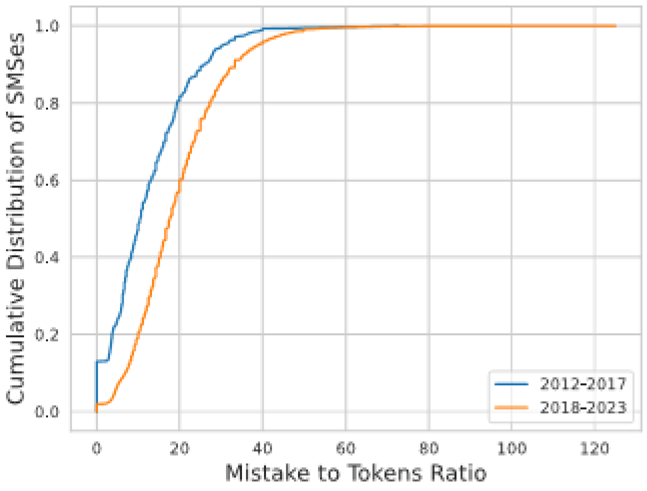


FIGURE 1. CDFs of ratios of spelling mistake to words (also termed as tokens or keyword) in Spam SMSes.

(2) EASILY SHARED After that, we get the lexical richness, Automated Readability Index (ARI), and Flesch score from each SMS in our dataset, together with the quantity of words, sentences, punctuation, and non-letters (like emoticons) [56]. The ARI, a popular reading measure, computes the comprehensibility of a text corpus, whereas lexical richness is the ratio of unique terms to total words, which shows obvious repeats (as shown in Equation1). A higher score indicates that the text is easier to read. For a summary of our findings, see Table 5. In contrast to, say, ham Spam SMSes have a lower Flesch score (93.7 vs. 60.9) and lexical richness (97% vs. 93%) than ham SMSes, but a higher ARI (2.13 vs. 12.81), therefore even though ham SMSes have a wider vocabulary, they are less readable. Between 2012 and 2017, the average readability index of spam SMSes was 80.2, but between 2018 and 2023, it dropped to 60.4.

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

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

3) SMS SPAM Length In order to avoid detection and maximize the delivery of their messages, SMS spammers use a range of message lengths. Figure 2 shows that the length of spam SMSes is much longer than that of Ham SMSes. Figure 3 shows that the average length of spam SMSes was generally constant, at 137 characters from 2012 to 2017 and 143 characters from 2018 to 2023. This regularity in duration indicates that spammers have discovered the sweet spot between being undetectable and getting their messages across.

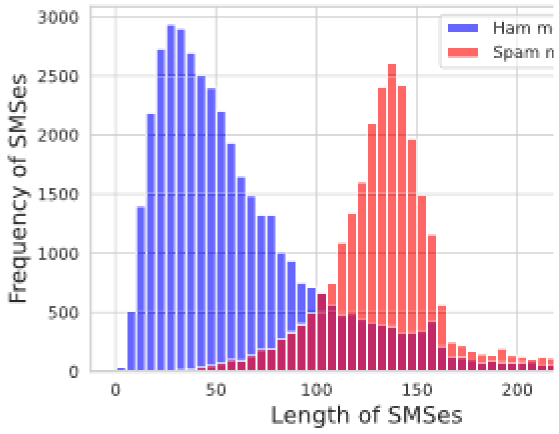


FIGURE 2. Length (in characters) vis-'a-vis Frequency distributions of SMSes in our dataset.

B. URLS AND URL-SHORTING SERVICES CONTINUE TO BE USED We carefully identify the URL-shortening service used in our spam SMS dataset and extract URLs from SMS using regular expressions. URLs in SMS spam have become much more common in recent years: While only 13% of spam messages included URLs between 2012 and 2017, that number jumped to 37% between 2018 and 2023. This shows that URLs are being used more and more by spammers to spread hazardous material and trick receivers into clicking on dangerous links. The use of URL-shortening services to generate spam URLs has also been on the rise. Maybe the services make it easier for spammers to conceal the true destination of their communications by including long or harmful URLs inside the character constraints of SMS messages.

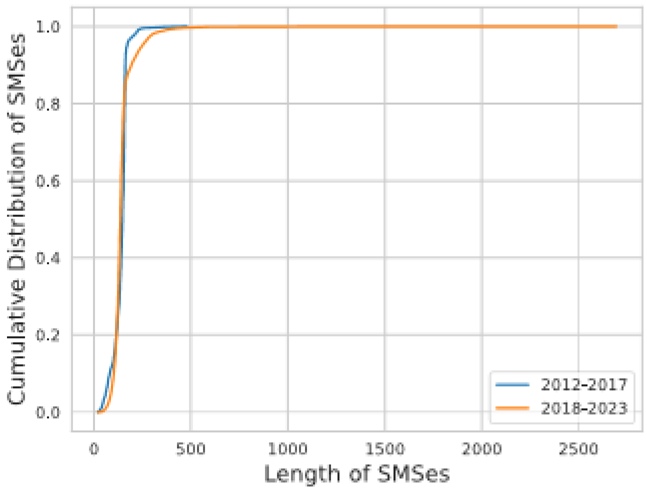


FIGURE 3. Distribution of SMS length (in characters) in Spam messages.

navigate spam filters, monitor CTRs, and URLs [57]. The top five URL shortener services found in our Super dataset are shown in Table 6. When looking at spam efforts from 2012–2017, just 1.97 percent made use of URL shortening providers. But URL shortener services were used by 9.76% of spam efforts. One reason URL shorteners are becoming more popular is because they make it easier to distribute spam anonymously and efficiently.

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

V. ANALYZING ML-BASED SPAM DETECTION TECHNIQUES

Figure 4 shows our experimental methodology, which consists of multiple steps: preprocessing the combined dataset, comparing feature models, choosing appropriate ML techniques, and testing the effects of different evasive techniques on the ML models. The sections that follow provide further detail on these procedures. Section A: Data Processing and Sorting A preprocessing step is used to eliminate stop words and extraneous characters from the combined dataset. Here, we make use of the NLTK library [58]. Furthermore, the dataset was divided into three parts: train (80%), test (20%), and hold-out (see Tables 7 and 8 for lexical analysis of the train and test parts, respectively) using the scikit-learn [59] module. The hold-out set is comprised of 225 randomly chosen spam SMS and is intended for validation purposes only.

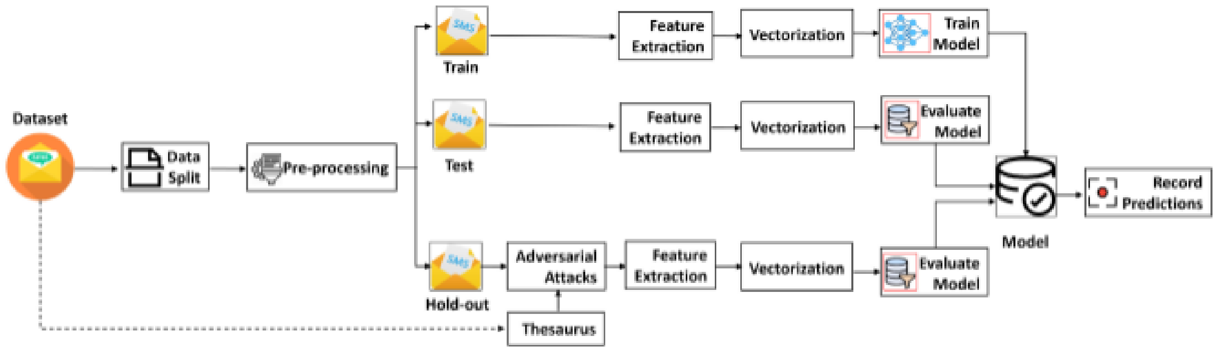


FIGURE 4. Overview of evaluation methodology.

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. In Sections V–D, we will go into detail about how this subset is used to assess the efficacy of the machine learning models. To train ML models, we use the train set, and to evaluate how well they do on new data, we use the test set. B. Extracting Features We prepare the dataset for ML models by transforming the SMSs into a structured feature space. To evaluate the effect on classifier accuracy, we convert the word lists in each message into a feature vector using different feature extraction methods (see Figure 5). This vector contains syntactic and semantic data. 1) Count-Based Vector Space Model for Syntactic and Non-Semantic Data As an alternative to semantic methods, we use n-grams (bigrams, trigrams) [61] and term frequency-inverse document frequency (TF-IDF) [62] to transform raw text into numerical characteristics. The next step is to convert the whole BoW and n-grams corpus to TF-IDF format. Word-of-mouth (BoW) records how often words appear in the corpus without taking word order into account. We solve this problem by counting the number of word pairings in an n-word sequence, which is a document's n-grams.

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

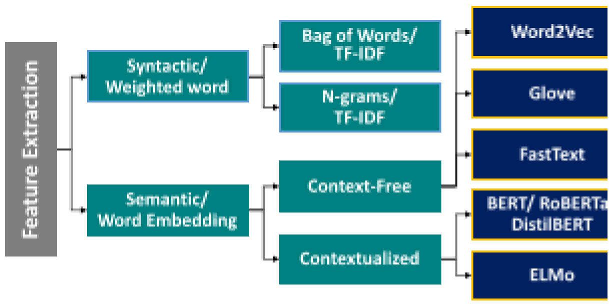


FIGURE 5. Feature extraction or representation techniques.

Nevertheless, a sparse matrix is still the outcome of even n-grams. Lastly, we determine the word significance in the text and corpus using TF-IDF. Rare words tend to have a higher TF-IDF score. To do this, we use the scikit-learn module in Python. 2) EMBEDDING INTO SEMANTIC Unfortunately, the semantic meaning of the words is not adequately represented when we extract syntactic word representations from SMSes, which makes interpretation challenging. To tackle this, we build a semantic feature vector for every word using word embedding [63]. This vector captures the word's semantics and context, which are further explained below.
a: SPACE MODEL FOR CONTEXT-INDEPENDENT VECTORS Classic word embeddings, a vector space model that is independent of context, is used for static or context-free embeddings. Word2Vec [64] and the Gensim package [65] are two popular word embeddings that we use to build static and dynamic models. The Word2Vec model employs a static method that involves pre-trained word embedding using 300-dimensional vectors. The dynamic method involves fine-tuning the vectors as they are being trained. To accomplish GloVe embeddings, we additionally employ the Python pretrained GLoVE model [66] released by the Stanford NLP Group, and for fastText embeddings, we use the Python module fastText [67]. See Appendix A for further information on each method.

b: A Vector Space Model Dependent on Context It is impossible to account for polysemy using the context-independent vector space paradigm. We address this issue by using contextualized word representations, which are able to capture word semantics in different contexts and eliminate the dependency on context for polysemous words. Embedding from Language Model (ELMo) [69] and Bidirectional Encoder Representations from Transformers (BERT) [68] are the contextualized word embeddings that we use. Unlike embedding representationsofindividualwords, BERT grasps the contextual link between several words. To produce BERT embeddings, we utilize the SimpleTransformers module in Python. To generate the associated embeddings, we access ELMo via TensorFlow Hub using Python. Appendix A

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, C l ick https://xxx.info/covid/

spam detection and anti-spam services. To further comprehend the possible influence of these evasive strategies on escaping SMS spam detection, we will examine their effectiveness. In our work, we focus on black-box settings, which mimic actual situations in which the spammer is unaware of the SMS spamdetectionmethods. We accommodate for the various potential changes in SMS context by using a two-step procedure to develop evasive samples. As part of our approach, we (i)compiled a spam dictionary using the 200 most common terms, and (ii)used the concept of imperceptibility to ensure that the modified version of the message retained all of its original semantic meaning. After that, we produce several instances of evasion strategies (lexical analysis in Table 11) by using methods like paraphrase, simple data augmentation [27], spacing, homograph (punycode), and hybrid evasive tactics. The study's evasion tactics, as demonstrated in Figure 6's taxonomy, are summarized in Table 10. We paid special attention to details like domain names, email

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

addresses, phone numbers, and URLs. Changing characters in these parts might result in new entities, which could alter the SMS's semantic meaning and affect its categorization.

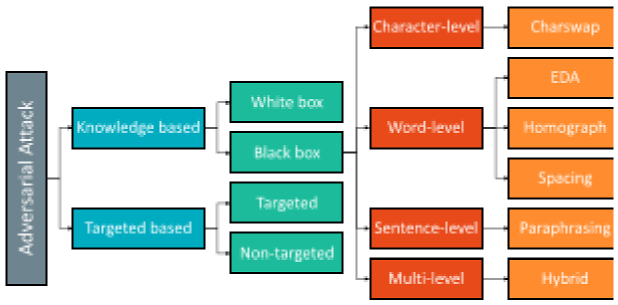


FIGURE 6. Taxonomy of evasion techniques employed in this study.

VI. EVALUATION OF MACHINE LEARNING MODELS

Performance, computational resources, interpretability, and other metrics are among those we use to assess ML models.

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

and resilience. To achieve this goal, 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 defend against sneaky tactics. When testing the model on an unbalanced dataset, we employ the following metrics: Precision (PR), Recall (RE), Accuracy (ACC), and F1-score (F1) [84]. These measures provide a fair evaluation of the model's performance in this unbalanced setting [85]. Here are the equations that define them:

ISSN:0976-0172

Journal of Bioscience And Technology
www.jbstonline.com

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

As used here, spam SMSes are considered positives while ham SMSes are considered negatives. We use the symbols P and N to denote the overall quantity of spam SMSes and ham SMSes, respectively. In contrast, TP stands for the number of properly categorized messages as spam and FP for the number of incorrectly classed messages as spam in the aforementioned formulae. Similarly, TN and FN are used to represent true negatives and false negatives, respectively. In this case, TN is the number of SMSes that were accurately identified as Ham, while FN is the number of SMSes that were incorrectly categorized as Ham. I. TRADITIONAL ML-BASED DETECTION TECHNIQUES PERFORMANCE EVALUATION First, we encode the SMS messages in the training and testing sets using multiple representation models (see Section V-B) to produce feature vectors. Then, we assess the ML models without adversarial instances. Following this, the classification algorithms are given these vectors to process. The algorithms are trained using the training set and then tested using the testing set. The next step is to implement a classic two-class support vector machine (TCSVM) with a number of features, such as Word2Vec, GloVe, n-grams, and BoW. Use the training set's Spam messages only to train the OCSVM and PU classifiers. The outcomes of our assessment are shown in Table 12. In general, we find that different classification algorithms and feature models provide different outcomes.

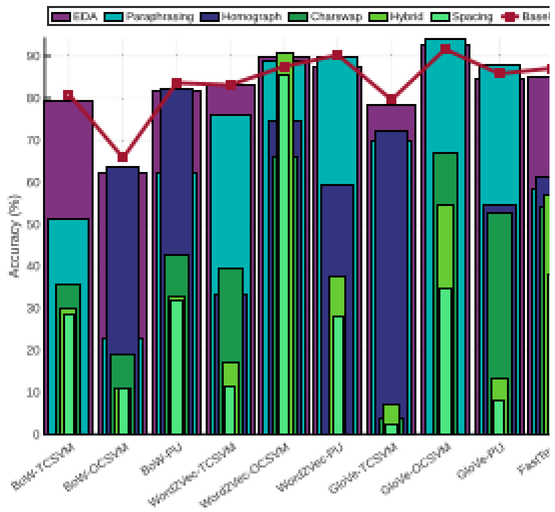
TABLE 12. Performance evaluation of shallow ML classifiers with the Super dataset (cf. Table 4). Here PR, RE, F1, ACC, and CM represent precision, recall, F1-score, accuracy, and confusion matrix, respectively. The top 3 models with the highest F1 and Acc are shown in bold.

Feature Model	Classifier	Performance Metrics				
		PR	RE	ACC	F1	CM$\left(\begin{smallmatrix} TN & FP \\ FN & TP \end{smallmatrix}\right)$
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%**	$\begin{pmatrix} 1884 & 34 \\ 939 & 10300 \end{pmatrix}$

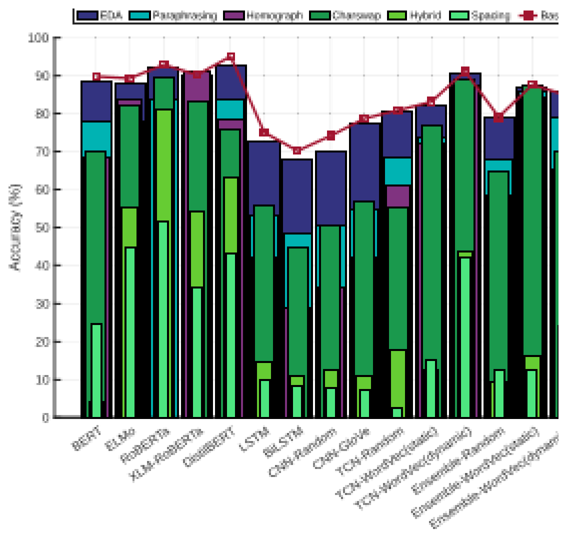
TABLE 13. Performance evaluation of DL classifiers with ''Super Dataset'' (cf. Table 4). The rows highlighted in red show the models with the highest accuracy and F1-score.

Classifier	Embedding	PR	RE	ACC	F1	CM$\left(\begin{smallmatrix} TN & FP \\ FN & TP \end{smallmatrix}\right)$
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%	$\begin{pmatrix} 1819 & 99 \\ 1004 & 10226 \end{pmatrix}$

the accuracy of these models by comparing their results to those of SMS messages without any changes. Table 15 shows the findings of DL model resilience against evasion tactics, whereas Table 14 shows the results of classic (or shallow) ML model robustness. The adversarial evaluation did not include the N-gram (bi-gram, trigram) based models because of their low performance. 1) ML-Based Detection Techniques' Robustness When it comes to shallow ML models, we find that the spacing evasion method is still the best at avoiding most of them; eight models were completely avoided. Furthermore, it should be mentioned that this method is very effective in eluding classifiers that have been trained using BoW/TF-IDF, 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.

The models that show the best performance are fastText, GloVe (TCSVM, OCSVM, PU), and TCSVM-Word2Vec (see §VI-A for details). Table 14 shows that of the two models tested, Charswap is the second most effective approach. The OCSVM with Word2Vec model comes up at number two, with a success rate of 66.1% (far lower than its baseline accuracy of87.5%), 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. 2) DL-Based Detection Techniques' Robustness Figure 7 shows how well DL models perform against stain removal methods. Applying the spacing evasion approach severely diminishes the performance of DL-based models, according to our study. For thirteen of the fifteen models tested, the spacing method proved to be the most effective evasion approach. In contrast to the spacing strategy, the hybrid evasion technique proves to be the second most successful when used to CNN-BiGRU Ensemble models that have been trained using WE-Random and Word2Vec (dynamic) embeddings.

VII. CONCEPT DRIFT ANALYSIS

The next step is to examine how well our classifiers perform in detecting SMS spam over time, taking idea drift into account. We execute a two-pronged experiment to evaluate the ML models' performance. The DS_legacy dataset, which is covered in §IV, is used as the training set in the first fold, whereas the DS_latest dataset is used as the test set. In the second fold, on the other hand, we train on the DS heritage dataset and utilize the DS newest dataset. The findings of the first experimental fold on shallow ML models are shown in Table 16, while those of the second experimental fold on DL models are shown in Table 17. Most shallow ML and DL classifiers perform poorly in the first fold when compared to the second fold; a few OCSVM and PU models are notable exceptions. Notably, these classifiers' accuracy and recall (the rate at which spam is detected) have dropped significantly, suggesting that the unique characteristics of contemporary real-world spam SMS messages are not well represented by the methods and datasets now in use. Still, the PU model constrained with Word2Vec and GloVe stands out from the others, showing off impressive results in every assessment measure in both folds. This highlights the fact that syntactic characteristics are

ISSN:0976-0172

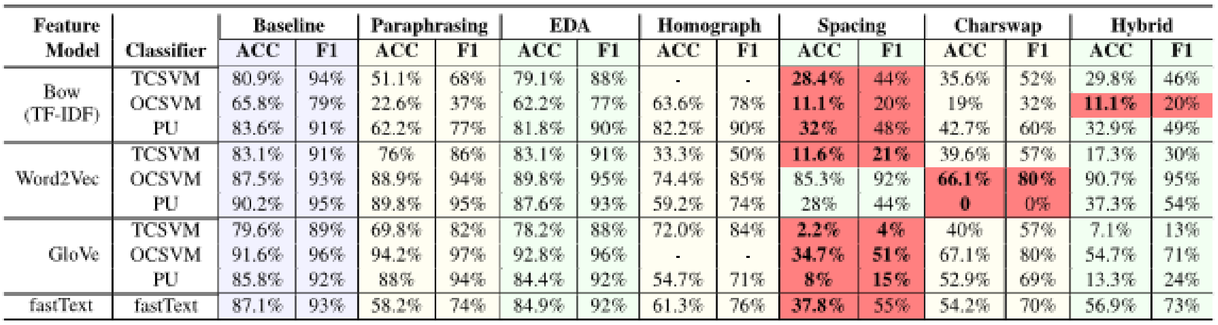
Journal of Bioscience And Technology
www.jbstonline.com

not as effective as word embeddings (i.e., semantic features) in detecting SMS spam.

VII. EVALUATION OF THE REAL-WORLD SMS ANTI-SPAM ECOSYSTEM

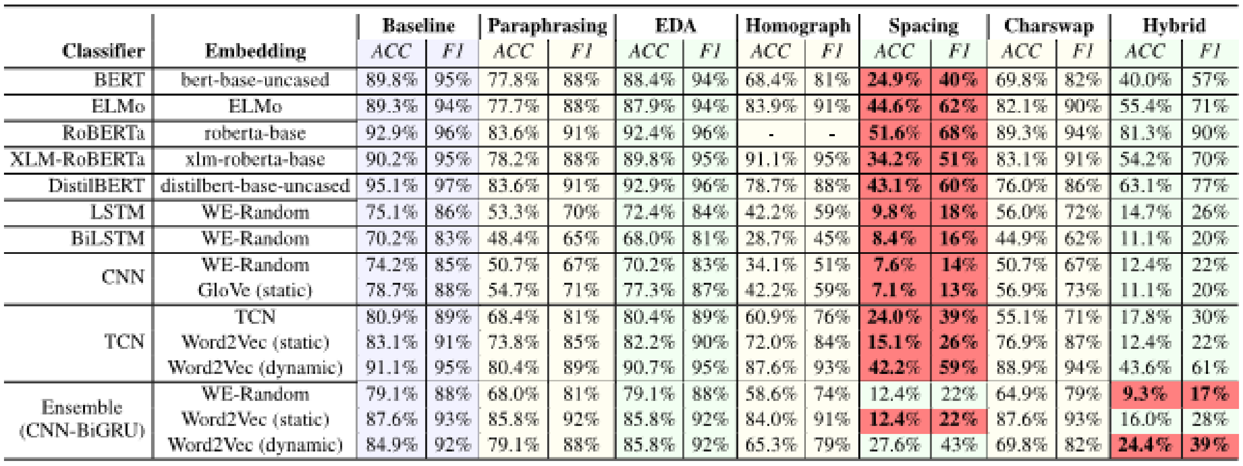
After that, we check how well the SMS anti-spam ecosystem works in practice. We have zeroed down on the two most important

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.



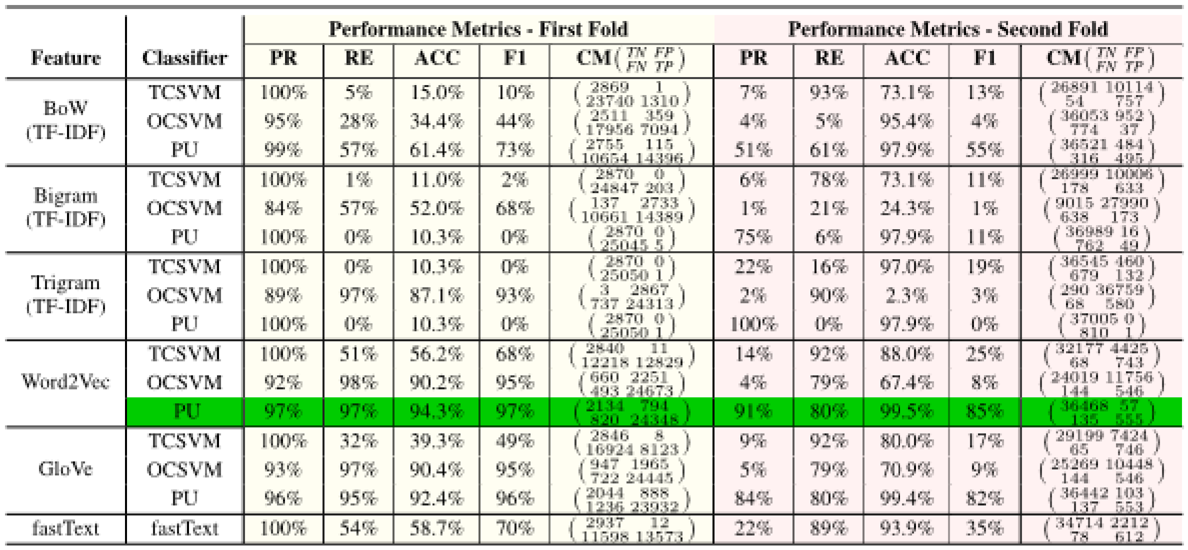
Feature Model	Classifier	Baseline		Paraphrasing		EDA		Homograph		Spacing		Charswap		Hybrid	
		ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
Bow (TF-IDF)	TCSVM	80.9%	94%	51.1%	68%	79.1%	88%	-	-	**28.4%**	44%	35.6%	52%	29.8%	46%
	OCSVM	65.8%	79%	22.6%	37%	62.2%	77%	63.6%	78%	**11.1%**	20%	19%	32%	**11.1%**	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%**

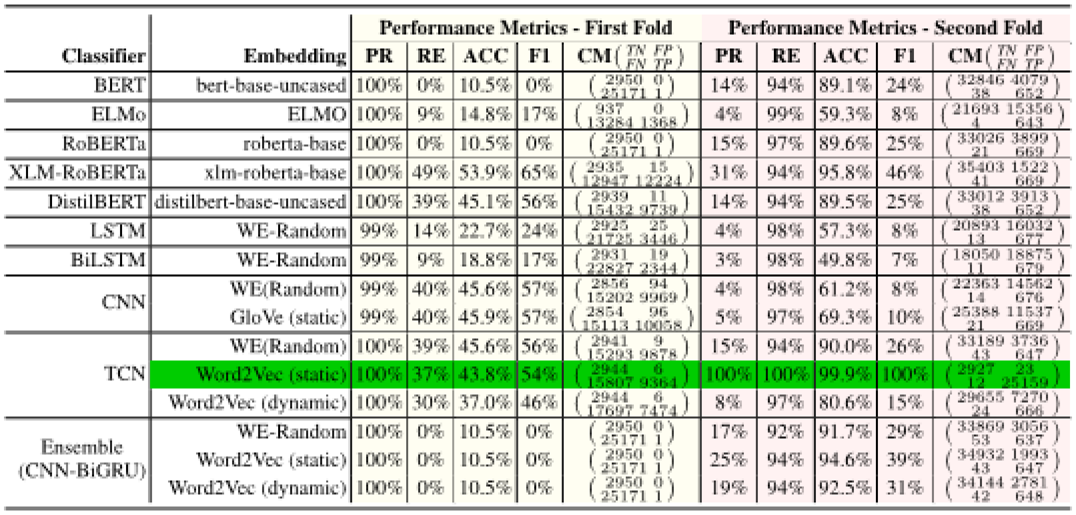
TABLE 16. Concept Drift Analysis of shallow or traditional ML classifiers. Based on ACC and F1 in both the folds, the row in green highlights the most successful model.



Feature	Classifier	Performance Metrics - First Fold					Performance Metrics - Second Fold				
		PR	RE	ACC	F1	CM$\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix}$	PR	RE	ACC	F1	CM$\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix}$
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%	$\begin{pmatrix} 34714 & 2212 \\ 78 & 612 \end{pmatrix}$

parties involved in this system: (i) widely used text messaging apps that provide spam filtering for SMS, and (ii) external web services that provide anti-spam solutions for SMS built by third parties. Like the examination of machine learning models in the preceding section (see to §VI), our study focuses solely on content-based SMS filtering. We concentrate on two main areas to evaluate the efficacy and robustness of the antispam ecosystem. To begin with,

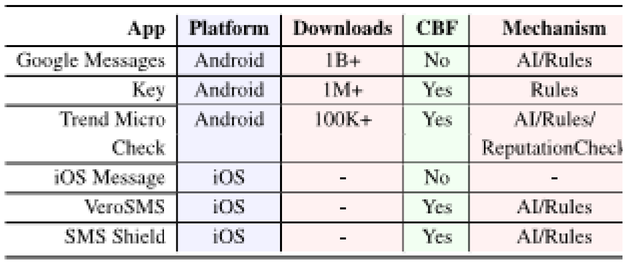
TABLE 17. Concept Drift Analysis of deep ML classifiers.



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%	$\begin{pmatrix} 34144 & 2781 \\ 42 & 648 \end{pmatrix}$

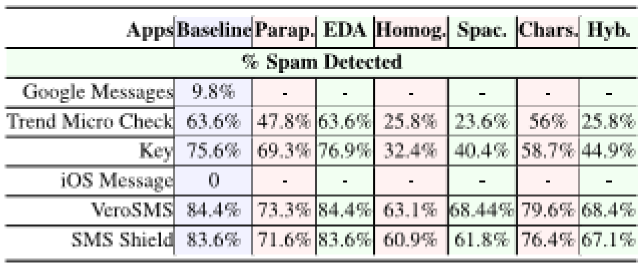
Below we take a look at how effective anti-spam solutions in the real world are at catching spam before it reaches its target audience. We do this by using real-world scenarios to submit all messages in the hold-out split to SMS anti-spam applications and services. in this point, we can see how well these methods worked in detecting and removing spam. Second, we look at how well SMS anti-spam applications and services defend against sneaky tactics. To do this, we run experiments using tweaked SMS instances that use different evasive strategies (see to §V-D). We test the robustness and efficacy of the widely-used SMS anti-spam applications and services by feeding them these malicious instances, and we see how well they handle these evasive tactics. Our understanding of the SMS anti-spam ecosystem's efficacy and robustness is enhanced by this thorough assessment. 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

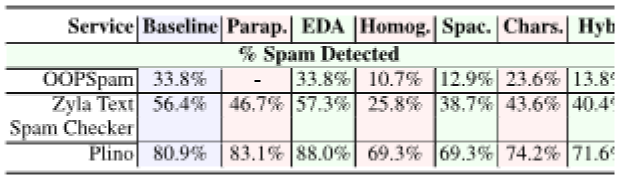
TABLE 19. Robustness and Performance evaluation of Text Messaging Apps on the holdout set.



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%

B. SMS Anti-Spam Services Evaluation We conducted a thorough search of the web to identify third-party online anti-spam SMS services that rely on content-based filtering and use machine learning to identify spam text. There does not seem to be any anti-spam SMS service, either publicly or privately, at this time. Nonetheless, OOPSpam[88], Text Spam Checker[89], and Plino [90] are only a few of the general-purpose text spam detection services that are accessible. For our analysis, we used the hold-out set to assess the efficacy of these three anti-spam programs. In order to get the detection result, these services provide APIs that accept text as input. One of the features that TextSpamChecker and Plino provide is the ability to classify text as "spam" or "ham," among other services. However, OOPSpam provides a spam chance score between zero and six.

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

We calculated the probability of mistakenly identifying legal communications by looking at how well the leading anti-spam service Plino classified benign SMS messages, much like the anti-spam text applications. We provided the same 225 harmless SMS texts into the Python API. Therefore, 139 out of 225 (61.78%) of the SMS messages that were not malicious were reported wrongly. Typical everyday communications that are mistakenly categorized include things like "Now They either use the football ground or the one near the faculty club" and "Why?" I need a vehicle for local usage, ideally an automatic, and I was wondering if you might suggest a good rental service. Plus, I was wondering if you wanted to stop by the Tuc store. D. ACCOUNTABLE DISCLAIMER Four months prior to publication, we notified app and anti-spam service providers of our review findings. We disseminated our results to provide perspective and criticism for the ongoing improvement of the application. They still haven't gotten back to us.Section IX: Discussion and Directions for Future Research Here, we provide a concise overview of the main takeaways from our performance and robustness examination of the antispam ecosystem and ML models. A. DETECTING SMS SPAM IS EASIER WITH DL MODELS THAN WITH TRADITIONAL ML MODELS We train and assess the robustness (ability to withstand adversarial assaults) and performance of 31 ML models. A recall of 85% or greater was reached by just six models out of sixteen, suggesting inadequate spam detection, even though most classical ML models show outstanding accuracy during performance assessment (see Table 12). On the other hand, only fifteen All of the deep ML models obtained an F1-score higher than 90% and attained a recall of 85% or higher (see to Table 13 for details). In terms of SMS spam detection, this demonstrates that DL is far better than conventional ML.B. DL Models Outperform Conventional ML Models in Terms of Resilence Tables 14 and 15 illustrate the results of the robustness study, which clearly demonstrate that deep ML models are less affected by adversarial instances than shallow ML models. The results demonstrate the superiority of deep ML models. Results also show that deep learning models based on transformers may greatly enhance model robustness.

X. CONCLUSION

We developed and analyzed a large new SMS dataset to provide insight on the evolving features of SMS spam. We tested several machine learning models and the anti-spam ecosystem on the dataset to see how well they identify SMS spam and how resilient they are. Every one of the models that relied on machine learning to detect authentic SMS messages (ham SMSes) performed admirably. 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. The shortcomings of existing anti-spam advancements and possible future research paths are brought to light by our examination of the machine learning model and the SMS anti-spam ecosystem. We maintain that SMS spam is still a major problem and call for further study into how to protect the public from SMS spam by creating systems that can counter the evasion tactics used by spammers. Accessible at https://github.com/smspamresearch/spstudy, our dataset and analysis highlight the shortcomings of existing anti-spam techniques and call for further study to create 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 studyofspamSMSdetectionusingmachinelearningclassifiers,’’inProc. 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 classSVMforSMSspamdetection,’’inProc.29thInt.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: TwitterguideddiscoveryandanalysisofSMSspam,’’inProc.ACMSIGSACConf. Comput. 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 effectsofdatasetsize,’’RadiotherapyOncol.,vol.105,no.1,pp. 115–121, Oct. 2012.