SMS Spam Detection Model

Sunny Shin, Karsyn Lee, Raymond Tang
W266: Natural Language Processing
UC Berkeley School of Information
{sunnyshin, karsyn, raymond.tang}@ischool.berkeley.edu

Abstract
Spam sent through SMS (short message service) poses harm to society, yet the current available models and algorithms for detecting SMS spam are limited in their efficiency. This research addresses SMS spam detection by exploring various machine learning frameworks, including logistic regression, convolutional neural networks (CNN), long short-term memory networks (LSTM), BERT, and BERTweet. By leveraging these extremely diverse techniques, we constructed multiple models and adopted an ensemble approach to improve the performance of our model. Through experimentation and training, the proposed ensemble model demonstrated exceptional performance, achieving an impressive F1 score of 99.89%. The findings of this research contribute valuable insights into SMS spam detection, paving the way for more effective measures to combat spam through SMS and protect mobile phone users from these messages.

1 Introduction

The method of communicating through SMS started in 1992 and has been used as a communication tool that has rapidly grown in popularity across the world. SMS spam refers to any messages that are unwanted and unsolicited, typically sent out in bulk by SMS. [1]. In 2022, Americans received more than 225 billion spam text messages, a 157 percent increase over 2021. [2]. Spam through SMS can lead to identity theft, malware, and unwarranted charges on your phone bill.

There are a few key reasons why this new area of interest is underdeveloped. There was previously no profit or incentive for telecom companies to block SMS spam. In March 2023, due to the rapid increase in SMS spam in the past few years, the Federal Communications Commission (FCC) is cracking down on spam SMS messages with new rules for telecom companies, citing a surge of consumer complaints in recent years tied to unwanted robo texts. [3]. The FCC has recently passed new guidance, imposing fines of up to $1,500 per message on telecom companies liable for allowing excessive SMS spam. The research that is performed through this project has the potential to be a model of interest to telecom companies that are striving to be compliant with the government to avoid reaping the financial consequences from the FCC. There is real novelty in SMS spam detection due to telecom interest.

It is also important to note that SMS spam detection is a challenging industry to break into for a few reasons. The first reason is that the SMS spam data available is difficult to obtain. Due to privacy measures, there is a lack of updated, genuine, public, and robust SMS Spam datasets. [4].

Another challenge that makes SMS spam detection a difficult model to create in natural language processing specifically, is the inherent nature of an SMS message. These messages are typically ambiguous, short in length, consist of limited header information, contain emojis, and contain abbreviations in their text. Therefore, previously established spam filters, most commonly for email, may have their performance seriously degraded when directly employed to deal with mobile Spam. [4]. Various for profit companies produce a spam detection service in the email setting including Avanan, Proofpoint, Webroot, and Mimecast. The most successful deployed email spam detection models in industry rely on features such as header information, emojis, abbreviations, special characters, and spelling errors to detect spam. Due to the key differences in the etiquette of language used in the email and sms settings, current industry spam detection techniques are not appropriate to handle SMS spam in industry yet.

The objective of this project is to leverage and learn from models similarly built for email spam detection to perform SMS spam detection and overcome the challenges that SMS language characteristics and the privacy of the data present. Spam detection in SMS using natural language processing is a new frontier of possibility due to the increased attention from the FCC and the new penalties faced by telecoms companies that fail to block SMS spam. By leveraging advanced machine learning techniques and training the model on a diverse dataset of both malicious and benign SMS, the team aims to create an intelligent system capable of detecting patterns, anomalies, and other indicators of spam vs non-spam. This NLP model will also play a crucial role in enhancing SMS security and informing users of potential spam.

2 Project Overview

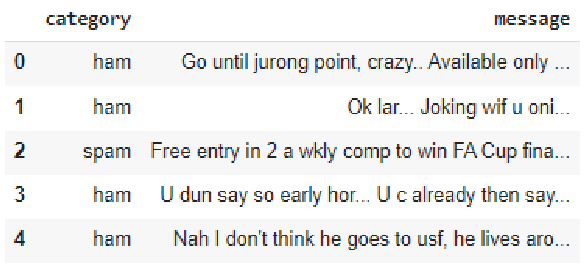
2.1 Dataset used

We explored the SMS Spam Collection data posted by University of California, Irvine (UCI) Machine Learning Repository in 2012 [5], which includes 5,574 total instances. The breakdown of the datasets include the followings:

- 425 SMS spam messages that were extracted from the Grumbletext Website. [6].
- A subset of 3,375 SMS were randomly chosen from the NUS SMS Corpus (NSC) which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at National University of Singapore. [7].
- A list of 450 SMS ham(non-spam) messages were collected from Caroline Tagg's PhD thesis project. [8].
- Finally, 1,002 SMS ham messages and 322 spam messages were added from the SMS Spam Corpus v.0.1 Big. [9].

The next image is a screenshot of the dataset that has been collected for spam research which contains all the mobile messages. The collection is composed of just one text file, where each line has the correct class followed by the raw messages. These messages are tagged accordingly as ham or spam. There are 5,574 rows with two columns in the dataset.

Fig. 1. Preview of the Dataset



	category	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

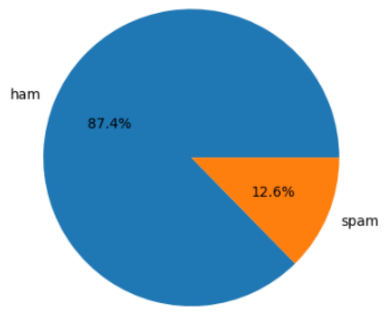
2.2 Challenges

As we conducted exploratory data analysis (EDA) to better understand the dataset, we observed a few prominent challenges. As discussed in the introduction, SMS inevitably contains a wide range of writing styles, languages, and grammatical variations, making it difficult to create a universal set of rules or patterns for identifying malicious content. SMS text can be even harder to classify than other forms of communications because it uses more informal, abbreviated, and slang language. The model needs to be trained on diverse and representative datasets to effectively capture these variations.

As discussed in section 2.1 above, the dataset was compiled from multiple sources, which can play a role in the consistency and quality of the dataset. The ham and spam examples might have some systematic differences (we will look into this during the pre-processing step) that models could overfit to without actually detecting whether they are spam or not. To avoid this, we implemented example misclassifications and fed in some selectively engineered test cases.

We also acknowledge that there is an imbalance in the original dataset as shown in Fig. 2. The spam category was significantly underrepresented, which made us consider and implement different ways to mitigate this challenge, discussed in further detail in section 2.4. It is worth noting that the original distribution of ham and spam in our dataset is a real-life depiction of the ratio of SMS ham to spam messages the general population receives. We found that in 2021, the spam rate was 10% of an individual's SMS messages. [10]. The spam rate in our dataset, at 13%, is not too far from this average, making it a reasonable representation of real-world conditions.

Fig. 2. The Percentage of Labels in the Original Dataset



We were able to counteract the imbalance of data through the resample method. We extracted the class with fewer observations, upsampled the subset and shuffled it back into the data. After the resampling was complete, we had a much more evenly distributed dataset, with a 47/53 split of data between the two classes. The entire process of how the resampling method was chosen is discussed in the data pre-processing Section 2.4.

2.3 Background

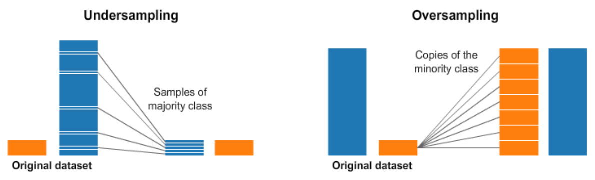
We analyzed the pre-existing machine learning models to distinguish spam from non-spam in email. Delany et al. (2012) provided a survey of existing works for filtering Spam SMS. [11]. They mostly covered articles that relied on traditional machine learning approaches but not deep learning. For example, Mathew & Issac (2011) compared Bayesian classifiers with other classification algorithms and found that the former was better to classify Spam text messages. [12]. Other models used K-Nearest Neighbor (KNN) and structural features, achieving good performance in spam classification. [13]. Bollam Pragna (2019) implemented a Support Vector Machines (SVM) classifier with Bag of Words (BoW) produced a high accuracy of 98.49%, which proved the effectiveness of a simple classifier. [14]. Only recently, researchers have started to use deep neural networks such as CNN and LSTM model for spam filtering. These models have shown high accuracy in detecting SMS spam. For example, Popovac et al. (2018) proposed a CNN-based architecture with one layer of convolution and pooling to filter SMS spam. They achieved an accuracy of 98.40%. [15]. Jain et al. (2019) used LSTM network (a variant of recurrent neural network) to filter SMS spam. With the help of 6,000 features and 200 LSTM nodes, their model achieved an accuracy of 99.01%. [16]. This paper extends these related works by achieving an even higher performance of 99.89% as shown in Section 4.

2.4 Pre-processing

The pre-processing stage was completed in various steps including but not limited to label encoding, tokenization, removing of stop words, special characters and punctuations.

As mentioned in Section 2.2, one of the challenges we faced was dealing with the imbalance of spam and ham in the dataset. Out of many different techniques, we considered the resampling method using sklearn library's resample package as a strategy. As shown in Fig. 3 below, resample allows us to either undersample the majority class or oversample the minority class.

Fig. 3. Resampling Mechanisms

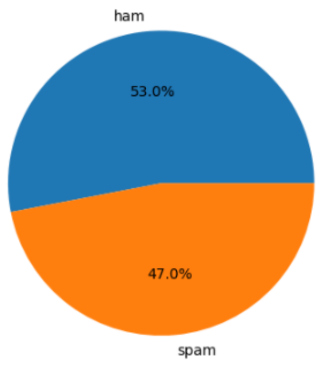


We also considered Synthetic Minority Oversampling Technique (SMOTE) which is another strategy to oversample the minority class by adding new instances or datasets that are synthesized from the existing dataset. However, due to the fact that our dataset is text-based, there is a possibility of a significant data distortion and noise as SMOTE interpolates existing data to create synthetic samples.

Given the smaller size of the dataset we were working with, we decided to implement oversampling the minority class technique in order to avoid working with even further downsized data. However, we acknowledge that this technique comes with its own limitations. Because the oversampling method duplicates random records from the minority class, it can lead to overfitting. Nonetheless, we determined that handling the imbalance by resampling or creating synthesized data points outweighs the risk of generalization.

The pre-processing steps allow us to have a more balanced dataset adequate for our machine learning model. You can see the progress from Fig.1 to Fig. 4 below, which highlights the impact of the resampling by showing the 47/53 split of data between the two classes instead of 13/87.

Fig 4. The Percentage of Labels after Resampling



3 Model

3.1 Methods

We decided to implement different models and compare the results in order to understand why certain models perform better than others. These are the five progressive approaches we took to perform a rigorous testing and comparison of the performance:

- Baseline model using logistic regression model
- RNN-LSTM
- CNN
- BERT (pooler token vs. CLS token)
- BERT-CNN ensemble
- BERTweet (pooler token vs. CLS token)
- BERTweet-CNN ensemble

4 Results

For a classification model, false positives and false negatives can have significant consequences. Misclassifying a legitimate SMS message as malicious could result in important communications being missed or delayed, while failing to identify a malicious SMS message could lead to security breaches or phishing attacks. Accuracy score is used when the true positives and true negatives are more important while F1-score is used when the false negatives and false positives are crucial. F1 score depicts the success of detecting spam and ham from SMS messages across the entire dataset through combining precision and recall. While we are reporting both scores, we decided that F1 score is our primary metric of success for the following reasons.

First, F1 score is a reliable metric commonly used to evaluate the performance of a binary classification model and is a more harmonic means of precision and recall. It is calculated as:

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

Second, F1 score is an informative measure that ranges from 0 to 1, with 1 being the best possible F1 score and 0 indicating poor performance. Because it takes both false positives and false negatives into consideration, F1 score penalizes the model more heavily for imbalanced performance in either precision or recall, thereby encouraging the classifier to be balanced and make accurate predictions for both classes.

4.1 Baseline Result

Our baseline model consists of a basic logistic regression model to classify the dataset from spam vs. ham. The F1 score is 0.9385, which was an excellent place for our team to start. This F1 score depicts that across the entire dataset, combining precision and recall, in relation to detecting spam and ham from SMS messages, the model had 93.85% accuracy. After achieving this F1 baseline score, we depicted that our model would be successful if we could beat this score by slightly over 1%, establishing an evaluation metric of an F1 score of 0.9500. We came to settle on this metric because our baseline was quite successful and we believed that a model producing an F1 score of 0.9500 would be valuable to users wishing to block spam as well as telecom companies looking to be compliant with spam laws. Dedeturk and Akay developed a machine learning model in 2020 that achieved a spam detection rate of 98% in accuracy score in email. [17]. Our goal of exceeding 0.95 F1 score holds us to an extremely high standard that is comparable to email spam detection rates, a field highly studied and practiced in industry.

4.2 RNN-LSTM Result

The RNN-LSTM model shows further improvement with an F1 score of 0.9540 in comparison to the baseline model. LSTM models are specifically designed to handle sequential data such as text data, so it can effectively capture the patterns across sequences due to its ability to retain information over extended periods through its memory cells.

4.3 CNN Result

The CNN model shows improvement when compared to the RNN-LSTM F1 score, producing an F1 score of 0.9915. Because CNN models are designed to capture local patterns in data using convolutional filters, the application of these techniques can better detect the patterns for our text-based data by understanding the context and meaning in sentences. Additionally, the word embeddings in a form of dense vectors are directly used as inputs for the CNN model, thereby extracting the context from sentences more efficiently than a logistic regression model.

4.4 BERT Results

The first approach we took was to test using the Pooler token method. This is the output representation of a CLS token after passing through the pooler layer. Another approach we took was using the Classification Task (CLS) token to run the BERT model. This is used to extract the representation of the CLS token from the entire output tensor. For the purpose of developing a SMS detection model, we learned that extracting the representation of the CLS token from the entire output tensor worked the best for us.

One of the hyperparameters we focused on fine-tuning was the maximum character length. After having it set for 150 initially, we learned that this specific parameter did not take into consideration how short the SMS texts are. We then changed the parameter to be 40 and the model performance improved significantly.

After training for 5 epochs, the BERT model with pooler token approach produced an F1 score of 0.8936 and CLS token at 0.9610. They were slightly lower than the F1 score of CNN model, which led us to believe that the CNN model was working more effectively than the traditional BERT model alone.

4.5 BERT-CNN Ensemble Results

After evaluating the performance and structure of our models until this point, we acknowledged the limitations of a traditional BERT, while also making note of the improved performance of the CNN model. Based on these observations, we created an ensemble model that leveraged the complementary features of BERT's contextualized word embeddings and CNN's ability to capture local patterns.

We designed the model architecture to incorporate the CNN layer after the BERT layer. Specifically, we reshaped the output from the BERT layer to fit the input requirements of a CNN layer. The performance increased slightly from the traditional BERT model with an F1 score of 0.9986. This ensemble model combined the strengths of both BERT and CNN models by providing a semantic understanding of the input text and capturing local patterns of the data.

4.6 BERTweet Result

As a variant of the BERT model, we decided to implement the BERTweet model, which was originally designed for processing and understanding tweets and other social media texts. Given how the SMS show very similar patterns and informality as tweets or other social media data. It uses the same transformer-based architecture as BERT, but is trained on a domain-specific corpus, allowing it to capture the peculiarities and nuances of informal language, slang and other characteristics commonly found in tweets.

Similar to the BERT model, we trained the BERTweet model using both the pooler token and CLS token separately. The Pooler token BERTweet model's F1 score was 0.9566 and CLS token BERTweet model's F1 score was 0.9478.

4.7 BERTweet-CNN Ensemble Result

In the BERTweet-CNN Ensemble approach, the model performed better than the traditional BERT model. We deduced that the CNN model worked the best with our dataset, given the strengths of CNN model and features. The length of the texts were short which made it suitable for detecting patterns for our text-based data. Accordingly, we built a BERTweet-CNN ensemble to confirm our hypothesis.

The performance of the BERTweet-CNN certainly was aligned with what we expected with the F1 score of 0.9989, which was the highest score we had seen out of all the models and the result of past research. [18]. Specifically, the latest research paper by Jain et al. (2022) implemented a similar SMS detection model and their SVM model resulted in an F1 score of 0.9538, which suggests that our BERTweet-CNN model's performance is outstanding. [19].

4.8 Analysis of the Results

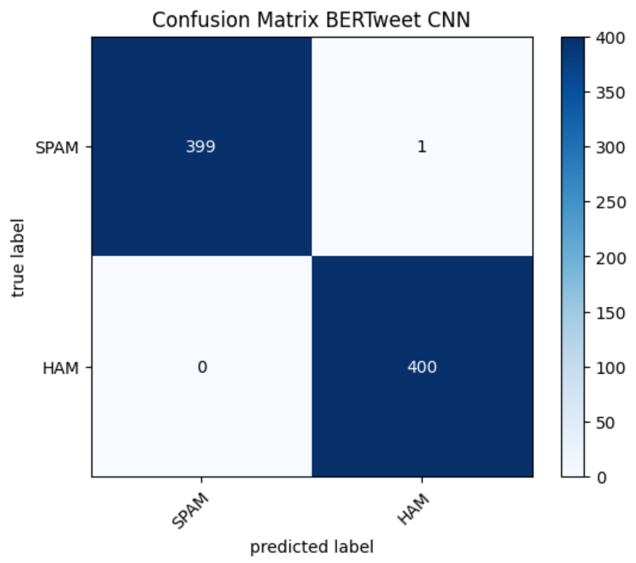
Below table shows a comparison of the performance from all the models we tested.

Table 1. Result Summary

Models and Configuration	Accuracy	F1 Score	Loss
Baseline	0.9425	0.9385	0.0575
RNN - LSTM	0.9542	0.9545	0.1137
CNN	0.9918	0.9915	0.0194
BERT - pooler token	0.9061	0.8936	0.3035
BERT - CLS token	0.9660	0.9610	0.1053
BERT-CNN Ensemble	0.9988	0.9986	0.0078
BERTweet - pooler token	0.9624	0.9566	0.1794
BERTweet - CLS token	0.9589	0.9478	0.1400
BERTweet-CNN Ensemble	0.9988	0.9989	0.0071

Based on the BERTweet-CNN ensemble model, we produced a confusion matrix below to visualize and understand the model's performance by comparing the predicted results to the actual outcomes. The result shown in Table 2 depicts that out of 400 instances in the training dataset, the BERTweet-CNN ensemble managed to predict 0.9989 of the cases correctly (both the true positives and true negatives).

Table 2. BERTweet-CNN Ensemble Outcome using the Confusion Matrix



Although the experimental results show an improved performance of the proposed spam BERTweet-CNN model compared to other models, the false predictions also indicate the drawbacks of this proposed model. We analyzed the content of the false prediction samples, including false positive and false negative samples, and found that there were a great number of unknown (UNK) data passed to the model, which is produced because the words were never seen in the training dataset. In other words, the unknown words obstruct the model from understanding the messages.

One unexpected trend in our evaluation metrics was a high loss in the BERT-pooler token model. After thorough analysis of this occurrence, we deduced that the BERT model returns two outputs that can be exploited for classification purposes, the last_hidden_state layer and the pooler output. Since we did not unfreeze any of the pretrained BERT layers and update their weights, the weight information is from the original data where the corresponding BERT models were pretrained. In our finding, BERT-Pooler had an F1 of 0.8936, which did not perform better than the baseline model whereas the BERTweet-Pooler had the F1 score of 0.9566. This indicates the BERT model, which was pre-trained using the next sentence prediction task, might not capture the entire context for our SMS classification. The BERTweet model that was pre-trained with tweets is more suitable for SMS sentences because data consists of message bodies that are more tweet-like in contrast to the Wiki-like sentences. One proposed way to fix this inconsistency would be to unfreeze some of the layers or even all the 12 BERT transformer layers for trainable embeddings. We found that our dataset was small in size and oversampled, so we chose not to go this route. As a solution, we experimented with the last_hidden_state output of BERT instead of the pooler output. First, we extracted the CLS token-level embeddings to feed that into a sigmoid classifier. The BERT-CLS model with an F1 score of 0.9610 versus the BERTweet-CLS model with an F1 score 0.9478. This result showed us that the model beat the baseline and the BERT-CLS model has improved from its pooler variant. We then fed the entire last_hidden_state as a sequence of vectors into a CNN to capture contextual information for every token in the SMS. With this configuration, the BERT-CNN model achieved an F1 score of 0.9986 versus the BERTweet-CNN model that achieved an F1 score of 0.9989. Both models improved dramatically from their non-convolutional variants with BERTweet-CNN outperforming BERT-CNN, and becoming our highest performing model.

Additionally, the SMS messages are usually very short, which increases the influence of every single word and makes the unknown words more influential. This explains why the RNN-LSTM's memory cell did not improve the results as much as the CNN's pattern detection. We found that spam detection in email had success using a CLS token rather than the pooler token as well. We did not expect to arrive at this result due to the short nature of SMS messages and the longer nature of emails. The pooler token and CLS token have less of an impact when the body of the message is short. This is depicted by the F1 scores between the two being less than a percent different (0.9566 and 0.9478).

5 Conclusion

In this paper, we tested multiple natural language processing (NLP) models to detect SMS spam messages. To lay a strong foundation, we built various models by analyzing existing models in email spam detection and getting a grasp of why the model could add true value to telecom companies. We evaluated the results of the BERTweet model by comparing it with other SMS detection approaches. The results show that out of all different NLP models, the BERTweet-CNN ensemble model performs the best on both datasets.

Despite the promising results achieved with our proposed BERTweet-CNN model, we recognize the need for further enhancements. Our datasets comprise only thousands of messages dated in 2012, which might limit the model's ability to generalize. Also, SMS spam messages evolve over time and our results might not be applicable to today's standard. To further test its performance, we plan to expand our model to a larger and more current dataset containing a more extensive collection of messages. Another notable challenge we encountered during the pre-processing step was the significant imbalance in the dataset, with ham messages outweighing spam messages. Although we attempted different techniques to address this issue, we aim to introduce different datasets in the future with a more balanced breakdown of ham and spam.

Our research demonstrated the effectiveness of the BERTweet-CNN ensemble model for SMS spam detection tasks. While the performance was impressive, we recognize the potential for further improvement, as well as ethical considerations while collecting additional data points from various sources. By addressing these aspects, our model can make a significant contribution to the field of SMS spam detection and enhance both the security and user experience for telecom companies and their customers.

References

[1] P. K. Roy, J. P. Singh and S. Banerjee (2020). *Deep learning to filter SMS spam* <https://pure.york.ac.uk/portal/en/publications/deep-learning-to-filter-sms-spam>

[2] S. Kantra (2023) *How to Stop Spam Text Messages* <https://www.techlicious.com/tip/how-to-stop-spam-text-messages/#:~:text=Spam%20texts%20in%20the%20form.call%20and%20 text%20app%20RoboKiller>

[3] B. Fung (2023). *FCC cracks down on spammy text messages* <https://www.cnn.com/2023/03/17/tech/fcc-spam-text-crackdown/index.html>

[4] M. Salman, M. Ikram and M. A. Kaafar (2022). *An Empirical Analysis of SMS Scam Detection Systems* <https://arxiv.org/pdf/2210.10451.pdf#:~:text=The%20SMS%20 Sam%20 filtering%20has,the%20data%20is%20another%20 challenge>

[5] T. Almeida and J. Hidalgo (2012). *SMS Spam Collection*. UCI Machine Learning Repository <https://archive.ics.uci.edu/dataset/228/sms+spam+collection>

[6] Grumbletext <https://www.grumbletext.co.uk/> Accessed 28 July 2023.

[7] T. Chen and M. Kan (2013). *Creating a Live, Public Short Message Service Corpus: The NUS SMS Corpus*. <https://link.springer.com/article/10.1007%2Fs10579-012-9197-9>

[8] C. Tagg (2009). *A Corpus Linguistics Study of SMS Text Messaging* <http://etheses.bham.ac.uk/253/1/Tagg09PhD.pdf>

[9] J.M. Gómez Hidalgo, G. Cajigas Bringas, E. Puertas Sanz, and F. García (2006). *Content Based SMS Spam Filtering* <http://devres.zoomquiet.top/data/20150319104818/index.html>

[10] C. Brisson (2021) *56 SMS Marketing Statistics for 2021*. <https://www.salesmessage.com/blog/sms-marketing-statistics#:~:text=Spam%20rates%20for%20SMS%20are.app%20notifications%2C%20and%20 direct%20mail>

[11] S. J Delany, M. Buckley, and D. Greene (2012). *SMS Spam Filtering: Methods and Data.* <https://arrow.tudublin.ie/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1022&context=scschcomart>

[12] K. Matthew and B. Issac (2011). *Intelligent Spam Classification for Mobile Text Message* <https://ieeexplore.ieee.org/document/6181918>

[13] A. K. Uysal, S. Gunal, S. Ergin and E. S. Gunal (2012). *A Novel Framework for SMS Spam Filtering* <https://ieeexplore.ieee.org/document/6246947>

[14] M. R. BollaPragada (2019). *Spam Detection using NLP Techniques* <https://www.ijrte.org/wp-content/uploads/papers/v8i2S11/B12800982S1119.pdf>

[15] M. Popovac, M. Karanovic, S. Sladojevic, M. Arsenovic, and A. Andrela (2018) *Convolutional Neural Network Based SMS Spam Detection* <https://ieeexplore.ieee.org/document/8611916>

[16] G. Jain, M. Sharma and B. Agarwal (2019). *Optimizing Semantic LSTM for Spam Detection* <https://www.infona.pl/resource/bwmeta1.element.springer-doi-10_1007-S41870-018-0157-5>

[17] F. J. Martino, R. A. Rodriguez, V. G. Castro, E. Fidalgo and E. Alegre (2022) *A Review of Spam Email Detection: analysis of spammer strategies and the dataset shift problem* <https://link.springer.com/article/10.1007/s10462-022-10195-4>

[18] X. Liu, H. Lu and A. Nayak (2021). *A Spam Transformer Model for SMS Spam Detection* <https://ieeexplore.ieee.org/document/9433507/references#references>

[19] T. Jain, P. Garg, N. Chalil, A. Sinha, V. K. Verma and R. Gupta (2022) *SMS Spam Classification Using Machine Learning Techniques* <https://ieeexplore.ieee.org/document/9734128/metrics#metrics>