A Comparative Study of Word Embedding Techniques for SMS Spam Detection

Prashob Joseph
School Computer Science and Informatics
Faculty of Computing, Engineering and Media
De Montfort University
Leicester, United Kingdom
P2676917@my365.dmu.ac.uk

Suleiman Y. Yerima
Cyber Technology Institute
Faculty of Computing, Engineering, and Media
De Montfort University
Leicester, United Kingdom
syerima@dmu.ac.uk

***Abstract*—E-mail and SMS are the most popular communication tools used by businesses, organizations and educational institutions. Every day, people receive hundreds of messages which could be either spam or ham. Spam is any form of unsolicited, unwanted digital communication, usually sent out in bulk. Spam emails and SMS waste resources by unnecessarily flooding network lines and consuming storage space. Therefore, it is important to develop high accuracy spam detection models to effectively block spam messages, so as to optimize resources and protect users. Various word-embedding techniques such as Bag of Words (BOW), N-grams, TF-IDF, Word2Vec and Doc2Vec have been widely applied to NLP problems, however a comparative study of these techniques for SMS spam detection is currently lacking. Hence, in this paper, we provide a comparative analysis of these popular word embedding techniques for SMS spam detection by evaluating their performance on a publicly available ham and spam dataset. We investigate the performance of the word embedding techniques using 5 different machine learning classifiers i.e. Multinomial Naive Bayes (MNB), KNN, SVM, Random Forest and Extra Trees. Based on the dataset employed in the study, N-gram, BOW and TF-IDF with oversampling recorded the highest F1 scores of 0.99 for ham and 0.94 for spam.**

***Index Terms*—Spam detection; machine learning; word embedding; bag-of-words; term frequency-inverse document frequency; n-grams; word2vec; doc2vec**

I. INTRODUCTION

SMS and email are the most commonly used means of communication by businesses, and other institutions, hence it is important to filter out unwanted messages called spam which is a major concern for 52% of all internet users based on a poll conducted in a survey [1]. Advertising emails are the most typical spam emails, making up 36% of all spam material globally [2]. Advertising spam can occasionally be seen as unpleasant and inconvenient. The second-largest spam category, including around 31.7% of all spam messages, is adult-related content. The third-largest spam email category, comprising 26.5% of all unsolicited emails, is financial-related matter. The most harmful spams are scams and fraud accounts, which make up roughly 2.5% of all spam emails. Phishing statistics show that 73% of those spammers are after identity theft [3].

Researchers have been developing spam filtering models for both emails and SMS messages using machine learning, by training these models with datasets consisting of ham and spam samples. Several important steps are involved in the development of the machine learning based spam filtering solutions. These include: data cleansing, text pre-processing, word embedding, model training, fine-tuning and testing. While the pre-processing step involves applying standard NLP techniques to produce tokenized input, the word-embedding step is a crucial aspect that may significantly impact the performance of the model. Hence, it is important to choose the right embedding technique for optimal performance of the spam filtering solution.

Previous works that presented SMS spam detection solutions have utilized different types of embedding techniques (such as N-grams, TF-IDF, Bag of Words, etc.), and have applied these with various machine learning algorithms. However, a comprehensive comparative study of how these techniques would perform when used to develop filtering models with different machine learning classifiers, is currently lacking. Hence, in this paper, we focus on efficiency of different word-embedding techniques used to represent the information in a message, and their impact on accuracy of spam detection models using various machine learning classifiers. The dataset used is an unbalanced, publicly available SMS spam collection dataset consisting of 747 spam messages and 4825 ham messages. We investigate the performance of three statistical word-embedding techniques and two techniques based on Neural Networks. The techniques include Bag-of-words, N-grams, TF-IDF, Word2vec and Doc2vec. The motivating research questions that underpin this study include:

- RQ1: what is the impact of the word-embedding technique on the performance of selected ML based models used for building spam detection filters?
- RQ2: which combination of word-embedding and machine learning classifiers achieves the optimal spam detection performance?
- RQ3: what effect does augmenting the data using oversampling to balance the training set, have on the performance of the different machine learning models?

The rest of our paper is organized as follows. In section 2, we review related work. The methodology is presented in section 3.

The 14th IEEE International Conference on Computational Intelligence and Communication Networks, CICN 2022, Al-Khobar, Saudi Arabia, 4-6 Dec. 2022
(Accepted version)

Section 4 presents and discusses the results of our comparative study. Section 5 outlines the conclusions and future work.

II. Related work

Sahmoud and Mikki [3] developed a Spam detection model using BERT and trained the model on 4 different datasets namely SMS spam collection v.1, SpamAssassin, Ling-Spam and Enron. They utilized 90% of each of the datasets for training and the remaining 10% for testing the BERT model. The F1-scores for Enron, SpamAssassin, Ling-Spam, and SMS spam collection v.1 datasets were 98.62%, 97.83%, 99.13% and 99.28% respectively. Janez-Martino et al. [4] proposed spam email categorization of already detected spam email. They used a hierarchical clustering algorithm to create a multi-class dataset, which contains three types of emails. This dataset is used to evaluate the combination of TF-IDF and BOW encodings with Naive Bayes, Logistic Regression and SVM. TF-IDF with SVM had the best micro F1 score performance of 95.39% for the task of multi-class spam classification.

Tong et al. [5] proposed a capsule network model combining the long-short attention mechanisms to achieve efficient Chinese spam detection. Their experimental results show that the model performed better than TextCNN, LSTM and BERT achieving 98.72% accuracy on an an unbalanced dataset and 99.30% on a balanced dataset. Roy et al. [6], utilized deep learning for Spam detection and compared the performance of CNN and LSTM models to traditional machine learning classifiers. The deep learning model used Glove representation, while the ML models used several statistical features to represent the messages. CNN and LSTM were shown to perform better than the ML models. However, their paper did not consider investigating different word embedding techniques.

Liu et al. [7], proposed a modified Transformer model for SMS Spam detection and evaluated it on the SMS Spam collection v.1 dataset. They applied TF-IDF word embedding for the ML classifiers and used Glove representation for the LSTM and their proposed spam Transformer model. The models were also evaluated on the UtkM1 Twitter spam dataset. The proposed model achieved 98.92% accuracy, 97.81% precision, 94.51% recall and 96.13% F1-score on the SMS spam dataset. Harisinghaney et al. [8], studied spam email classification based on text and images using KNN, Naive Bayes and Reverse DBSCAN algorithm. Their work was based on a subset of emails from the Enron corpus dataset. The best accuracy of 87% was obtained with Naive Bayes.

Tida et al. [9], developed a universal spam detection model using BERT base uncased models. The combined model was trained with four datasets. An overall accuracy of 97% was obtained with and F1-score of 0.96. Rahman et al. [10], proposed a new spam detection using Bidirectional LSTM and CNN and employed keras embedding layer for word embedding. The models were trained on lingspam dataset and spam text classification dataset. The proposed CNN-bi-LSTM model achieved 98.25% F-measure.

Laorden et al. [11], applied Word Sense Disambiguation (WSD) to spam filtering, which is a pre-processing procedure capable of disambiguating confusing terms in messages to improve filtering mechanism in spam filters. The proposed approach was applied to LingSpam dataset and TREC 2007 Public corpus separately. Several ML classifiers were evaluated and Random Forest showed the best results. Raj et al. [12], proposed a robust method for spam classification using LSTM. Word2vec was used to convert the SMS spam collection dataset to vectors. Experimental results proved the LSTM model outperformed Machine Learning techniques like RF, SVM, KNN and decision tree with an accuracy of 97.5%.

Almeida et al. [13], discuss the best classification model identified for SMS spam collection dataset, which had a total of 81,175 tokens extracted from the corpus. In their analysis, linear SVM outperformed all other evaluated models. It was able to capture 83.10% of all spam with the cost of blocking only 0.18% of legitimate messages, and an accuracy rate higher than 97.5%. Note that they did not perform stop word removal or word stemming as they claimed that other researchers found them detrimental to spam filtering accuracy. Paper [14] proposed a semi-supervised novelty detection approach for SMS spam detection. They applied one-class SVM by training the model as an anomaly detector using only ham messages. Their technique achieved an overall accuracy of 98%, with 100% detection rate (recall) for spam messages and 3% false positive rate for ham.

Ghourabi et al. [15], proposed a hybrid deep learning model for detecting SMS spam messages based on combining CNN and LSTM. It aims to deal with mixed text messages written in Arabic and English. For comparative purpose, they implemented several models and showed that CNN-LSTM outperformed all other models by achieving highest accuracy of 98.37%. Mishra et al. [16], proposed a model called “Smishing detector” with reduced false positive rate to detect smishing messages. The proposed model contains 4 modules. Using Na¨ıve Bayes classification algorithm, first module analyzes the content of text messages and identify malicious content. URL contained in the message is inspected by second module. The source code of the website linked in the messages is identified by the third module. The last module is an APK download detector which identifies if a malicious file is downloaded when URL is called. The experimental test showed the model achieved an accuracy of 96.29%. Other works that have applied ML to the problem of SMS spam detection include [17]–[22].

Unlike previous works, this paper aims to comparatively evaluate the performance of several word embedding techniques using 5 machine learning classifiers and a benchmark spam dataset that is publicly available. The objective is to answer the research questions (RQ1-RQ3) outlined in section 1, through an extensive empirical study. In the next section, we detail the methodology adopted in our study.

III. Methodology

A. Dataset

The dataset used in our work is the SMS spam collection dataset, available from [23], which consists of 5572 messages

with 4825 being ham, and 747 being spam. The dataset distribution is shown in Figure 1. The word embedding techniques will be applied to the dataset after the initial pre-processing steps have been performed.

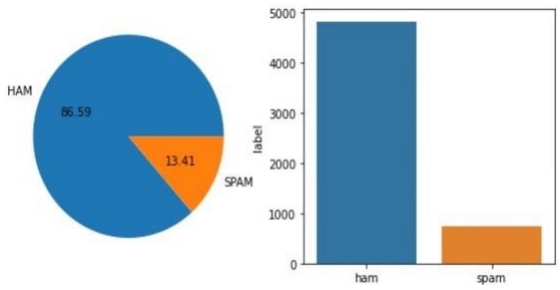

Fig. 1. Distribution of ham and spam in the original dataset

B. Text Preprocessing

The data pre-processing workflow is shown in figure 2. In the first step, all the characters in the text messages other than 'a-z' or 'A-Z' were removed using regular expression. This removes characters such as symbols and numbers which do not contribute any meaning to the sentences. Next, all the alphabets in the text were converted to lower case, to prevent similar words with different cases from being interpreted as unique words. The sentences were tokenized to a list of words using Tweet Tokenizer. The NLTK library was then used to remove 'stop words' from the sentences i.e. words of the English language which do not add any semantic meaning to the sentence. Stemming of the words in the list was done using the Snowball stemmer. Finally, the cleaned list of words were joined to form a sentence and added to a list called 'corpus'.

C. Word embedding techniques

Since machine learning algorithms can only process numerical data, words and text need to be made meaningful by expressing them numerically. Word embedding or word vectorization techniques are used to convert words into numerical representation. Words or phrases from a vocabulary are mapped to a corresponding vector of real numbers, which can then be utilized by the ML algorithms for training and prediction. In this study, we will focus on evaluating five commonly used approaches which are: Bag-of-words, N-grams, TF-IDF, Doc2Vec and Word2vec.

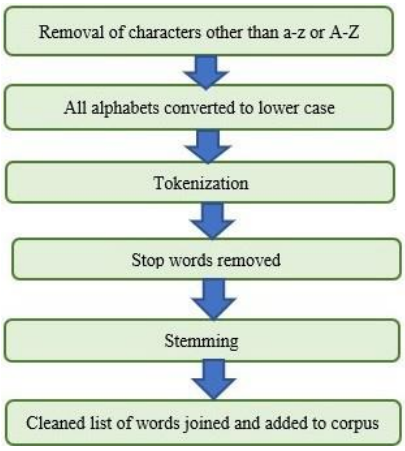

Fig. 2. Overview of text preprocessing workflow

1) Bag of Words: This is a vectorization technique which uses a vocabulary of known words and count of occurrence of each word in the sentence to represent the text without considering the order or structure of the word in the document. We implemented this by using the Count Vectorizer function imported from SkLearn library. We have selected only maximum occurring features of 2500, as including more features generally showed a decline in performance of the ML model.

2) N-Grams: This is a variant of the Bag of Words technique, where combination of words in the sentence are considered during vectorization. For example, consider the sentence:["I love this movie"]. When N=1, i.e. unigram model, the features will be: ["I","love","this","movie"]. When N=2, i.e. bi-gram model, the features will be: ["I love","love this","this movie"]. When N=3, i.e. tri-gram model, features will be: ["I love this","love this movie"]. We implemented N-grams vectorization by combining unigram and bi-gram features, with the maximum selected features set at 2,000. This was found to be the optimum configuration, as other parameters showed decline in models' performance.

3) TF-IDF: Term Frequency-Inverse Document Frequency is a technique that identifies the importance of a word in a sentence by making use of statistical measures. Term Frequency (TF) is used to calculate the frequency of words in sentence by dividing total number of repetitions of the word in a sentence by total number of words in the sentence. The IDF score is used to calculate the rarity of words, as words that are less frequently used in corpus can contain more significant information. IDF is calculated by dividing total number of sentences by number of sentences containing the words and computing the log of the result. The final TF-IDF score is calculated by multiplying TF and IDF scores.

4) Word2Vec: Word2Vec [24] is a word embedding technique published by Tomas Mikolov and his colleagues at Google [25]. The algorithm uses a neural network model to learn word associations from a dataset, representing each distinct word with a vector. The vector is a list of numbers chosen to capture semantic and syntactic meaning of words in a sentence. Word2vec has two variants: Continuous Bag of Words (CBOW) and Skip-gram. In CBOW, based on various input words, the neural network predicts the target word closely related to context of the input words; while Skip-gram takes one word as input and predicts closely related context words. In our implementation, we have selected CBOW, the default model for Word2Vec. To reduce computational time, we implemented an 'average word2vec' technique whereby the average of vectors of all words in a sentence are taken, to generate a new set of vectors of length 100.

5) Doc2Vec: Doc2Vec [26] is a two layered network which accepts a sentence, paragraph or document as input and converts

it into vectors instead of converting individual word to vectors as done in Word2Vec. In Doc2Vec, instead of using just words to predict the next word, another feature (vector D) which is document unique is added which represents the concept of the document. During the training process, word vector W and document vector D is trained as well, which holds the numeric representation of the document. This model is called Distributed Memory version of Paragraph Vector (PV-DM) which remembers the topic of the paragraph. We used the Gensim Python library [27] to implement both word2vec and doc2vec word embeddings in our study.

IV. Experiments and results

In this section we describe the experiments undertaken in our study and present the results of the comparative analysis of the word embedding techniques, using different machine learning classifiers. The ML classifiers used include: Multinomial Naive Bayes (MNB), K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Random Forest (RF), and Extra Trees (ET). The performance metrics used include: Precision, Recall, Accuracy and F1-score. In the experiments, 70% of the dataset is used for training and the remaining 30% for testing.

A. Performance of BOW for SMS spam detection

In Table I, the results of BOW vectorization for the 5 ML classifiers are shown. The precision for KNN is quite low for spam (38%), which means that it has a high false positive rate for ham prediction. Thus, KNN has the lowest F1-score of 0.55 despite having 100% recall for spam. (The dataset is highly unbalanced with only 13% as spam). The highest F1-scores are obtained with MNB (0.99 for ham and 0.94 for spam). MNB has 93% recall for spam which is lower than recall for RF, ET and KNN; however it has the best spam precision of 95% meaning that it blocks much less ham (less ham false positives) compared to the other classifiers.

TABLE I
Performance of different ML models with BOW

	Class	Precision	Recall	F1-score	Accuracy
MNB	ham	0.99	0.99	0.99	0.98
	spam	0.95	0.93	0.94	
ET	ham	1.00	0.98	0.99	0.98
	spam	0.87	0.98	0.92	
RF	ham	1.00	0.97	0.99	0.97
	spam	0.83	0.99	0.90	
KNN	ham	1.00	0.91	0.95	0.91
	spam	0.38	1.00	0.55	
SVM	ham	0.96	0.94	0.95	0.92
	spam	0.66	0.72	0.69	

B. Performance of N-grams for SMS spam detection

In Table II, N-grams results are presented. With this technique, MNB also showed the highest F1-scores of 0.99 and 0.94, but tied with ET. However, we consider MNB as the better option due to its higher ham recall (99%) compared to ET ham recall (98%), which implies that MNB blocks less ham messages than ET model.

TABLE II
Performance of different ML models with N-grams

	Class	Precision	Recall	F1-score	Accuracy
MNB	ham	0.99	0.99	0.99	0.98
	spam	0.92	0.96	0.94	
ET	ham	1.00	0.98	0.99	0.98
	spam	0.89	0.99	0.94	
RF	ham	1.00	0.97	0.99	0.98
	spam	0.84	1.00	0.91	
KNN	ham	1.00	0.91	0.95	0.92
	spam	0.42	1.00	0.60	
SVM	ham	0.96	0.95	0.95	0.92
	spam	0.69	0.74	0.72	

C. Performance of TF-IDF for SMS spam detection

In Table III, TF-IDF results are presented, where ET obtained the highest F1-scores of 0.99 and 0.93 for ham and spam respectively. The recall rate for ham was 98% while that of spam was 99%. Compared to BOW and N-grams, the F1-scores for SVM and RF were better with TF-IDF. We can also conclude based on the results, that MNB with N-grams or MNB with BOW, outperform ET with TF-IDF.

TABLE III
Performance of different ML models with TF-IDF

	Class	Precision	Recall	F1-score	Accuracy
ET	ham	1.00	0.98	0.99	0.98
	spam	0.88	0.99	0.93	
MNB	ham	1.00	0.97	0.99	0.98
	spam	0.84	0.99	0.91	
RF	ham	1.00	0.98	0.99	0.98
	spam	0.85	0.99	0.92	
KNN	ham	1.00	0.91	0.95	0.91
	spam	0.39	0.99	0.56	
SVM	ham	1.00	0.98	0.99	0.98
	spam	0.87	0.97	0.92	

D. Performance of word2vec for SMS spam detection

The results for word2vec embedding are shown in Table IV. The parameters used for the model were: vector size= 100; window=45; min count =2. From the table, RF had the highest F1-score of 0.98 for ham and 0.85 for spam. All the classifiers had worse performance with word2vec compared to BOW, N-grams and TF-IDF, except for KNN. Thus we can conclude that the statistical word embedding methods were better than word2vec based on the experiments on this dataset.

E. Performance of doc2vec for SMS spam detection

The results for doc2vec embedding are shown in Table V. The parameters used for the model were: vector size = 750; window = 40; min count =2. The overall performance for the classifiers was better than word2vec. RF had the highest F1-score of 0.99 for ham and 0.93 for spam. The results for KNN were better than with the other word embedding techniques. Thus, KNN and RF were the strongest classifiers for the doc2vec embedding approach.

TABLE IV
PERFORMANCE OF DIFFERENT ML MODELS WITH AVERAGE WORD2VEC

	Class	Precision	Recall	F1-score	Accuracy
RF	ham	0.99	0.97	0.98	0.96
	spam	0.79	0.92	0.85	
KNN	ham	0.97	0.96	0.97	0.94
	spam	0.77	0.82	0.80	
SVM	ham	0.89	0.85	0.87	0.76
	spam	0.03	0.04	0.03	
ET	ham	0.99	0.94	0.97	0.94
	spam	0.62	0.93	0.74	
MNB	ham	0.69	0.88	0.78	0.66
	spam	0.43	0.19	0.26	

TABLE V
PERFORMANCE OF DIFFERENT ML MODELS WITH DOC2VEC

	Class	Precision	Recall	F1-score	Accuracy
RF	ham	1.00	0.98	0.99	0.98
	spam	0.89	0.97	0.93	
KNN	ham	0.99	0.98	0.99	0.96
	spam	0.86	0.96	0.91	
SVM	ham	1.00	0.95	0.98	0.96
	spam	0.71	0.99	0.83	
ET	ham	0.99	0.98	0.98	0.97
	spam	0.85	0.91	0.88	
MNB	ham	0.99	0.98	0.98	0.97
	spam	0.87	0.92	0.89	

To evaluate the effect of oversampling, due to having an unbalanced dataset, we applied SMOTE to the training sets prior to training the classifiers. The testing sets (30% of the total) were left untouched so as to enable direct comparison with the unbalanced dataset scenarios. It was observed that BOW and N-grams did not achieve better performance for the ML classifiers with the introduction of the over-sampled training set. However, there was noticeable improvement with TF-IDF for ET, RF, KNN and SVM. This can be seen by comparing results of Table VI with Table III.

Table VII depicts the best classifiers from each of the word embedding techniques. From the Table, we can deduce that RF is a good choice for doc2vec and word2vec, while MNB should be considered for use with BOW or N-grams. In the case of TF-IDF, ET was the best before oversampling, while SVM was the best classifier after applying the SMOTE oversampling.

In Figure 3, average F1-scores are shown for the embedding techniques using RF. Doc2Vec had the highest average F1-score (0.96), while word2vect had the lowest (0.915). In Figure 4, average F1-scores are shown for the embedding techniques using Extra Trees. N-grams had the highest average F1-score (0.965), while word2vect had the lowest (0.855). In Figure 5, average F1-scores are shown for the embedding techniques using MNB. N-grams and BOW had the highest average F1-score (0.965), while word2vect had the lowest (0.52).

V. CONCLUSION AND FUTURE WORK

In this paper we presented a comparative analysis of popular word embedding techniques using KNN, RF, MNB, ET and SVM classifiers. We considered three statistical embedding approaches i.e. TF-IDF, BOW and N-grams as well as two Neural Network-based techniques i.e. word2vec and doc2vec. In general, we observed the best overall performance in spam detection with the statistical word embedding methods paired with MNB, RF or ET. While doc2vec appeared promising, based on the dataset employed in the study, word2vec did not perform well compared to the other methods. Nevertheless, we envisage that the Neural Network based techniques could achieve improved performance with a larger dataset. This will be further explored in our future work. From our study, TF-

TABLE VI
PERFORMANCE OF DIFFERENT ML MODELS WITH TF-IDF (OVERSAMPLING)

	Class	Precision	Recall	F1-score	Accuracy
ET	ham	1.00	0.98	0.99	0.98
	spam	0.90	0.98	0.94	
MNB	ham	0.97	0.99	0.98	0.97
	spam	0.95	0.85	0.90	
RF	ham	1.00	0.98	0.99	0.98
	spam	0.86	1.00	0.92	
KNN	ham	0.99	0.96	0.97	0.95
	spam	0.72	0.94	0.81	
SVM	ham	0.99	0.99	0.99	0.98
	spam	0.95	0.93	0.94	

TABLE VII
PERFORMANCE OF BEST ML MODEL WITH VARIOUS WORD-EMBEDDINGS

	Best	Class	Precision	Recall	F1	Acc
BOW	MNB	ham	0.99	0.99	0.99	0.98
		spam	0.95	0.93	0.94	
N-gram	MNB	ham	0.99	0.99	0.99	0.98
		spam	0.92	0.96	0.94	
TF-IDF	ET	ham	1.00	0.98	0.99	0.98
		spam	0.88	0.99	0.93	
TF-IDF oversampled	SVM	ham	0.99	0.99	0.99	0.98
		spam	0.95	0.93	0.94	
AVg. W2V	RF	ham	0.99	0.97	0.98	0.96
		spam	0.79	0.92	0.85	
Doc2Vec	RF	ham	1.00	0.98	0.99	0.98
		spam	0.89	0.97	0.93	

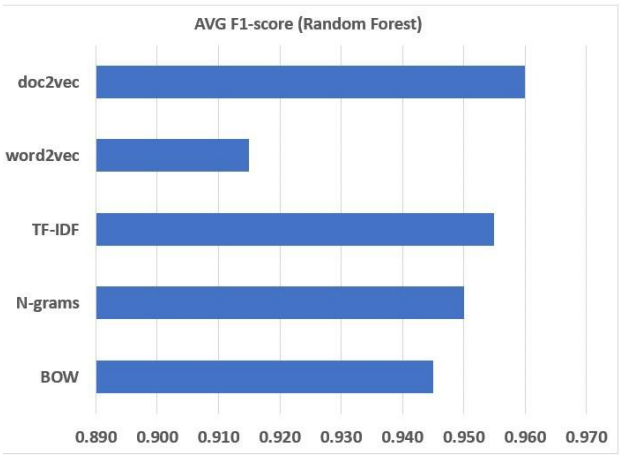

Fig. 3. Average F1-scores for different embedding techniques with RF classifier

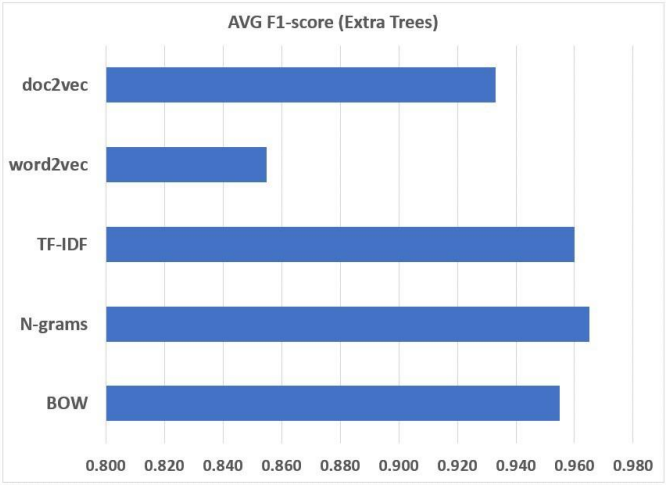

Fig. 4. Average F1-scores for different embedding techniques with Extra Trees

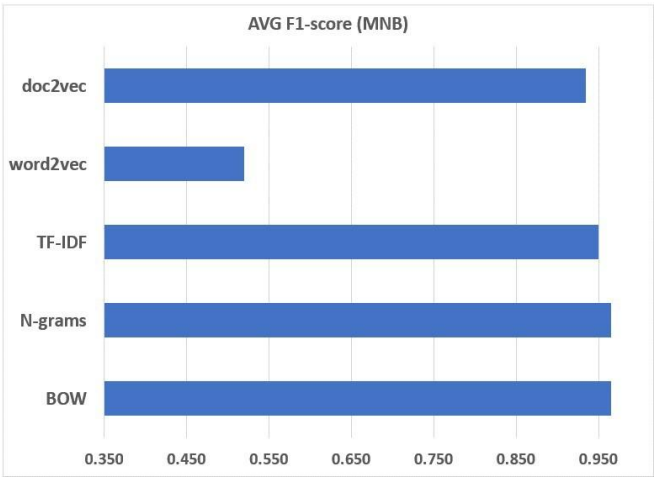

Fig. 5. Average F1-scores for different embedding techniques with MNB

IDF derived performance gain from employing oversampling to balance out the training sets using SMOTE, while N-grams and BOW did not. However, other forms of data augmentation techniques could be explored in future work.

REFERENCES

[1] G. Shanmugasundaram, S. Preethi, and I. Nivedha, "Investigation on social media spam detection," in *Proceedings of 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2017, pp. 1–8.

[2] SpamLaws. Spam Statistics and Facts. [Online]. Available: https://www.spamlaws.com/spam-stats.html/ [Last accessed: 25 Sept., 2022]

[3] T. Sahmoud and M. Mikki, "Spam detection using BERT," *Cryptography and Security (cs.CR); Machine Learning (cs.LG) arXiv:2206.02443 [cs.CR]*, 2022.

[4] F. Ja´n˜ez-Martino, E. Fidalgo, S. Gonza´lez-Mart´ınez, and J. Velasco-Mata, "Classification of spam emails through hierarchical clustering and supervised learning," *Computation and Language (cs.CL); Machine Learning (cs.LG) arXiv:2005.08773 [cs.CL]*, 2020.

[5] X. Tong, J. Wang, C. Zhang, R. Wang, Z. Ge, W. Liu, and Z. Zhao, "A Content-Based Chinese Spam Detection Method Using a Capsule Network With Long-Short Attention," *IEEE Sensors Journal*, vol. 21, no. 22, pp. 25 409–25 420, Nov. 2021.

[6] K. R. Pradeep, J. P. Singh, and S. Banerjee, "Deep learning to filter SMS Spam," *Future Generation Computer Systems*, vol. 102, pp. 524–533, Jan. 2020.

[7] X. Liu, H. Lu, and A. Nayak, "A Spam Transformer Model for SMS Spam Detection," *IEEE Access*, vol. 9, pp. 80 253 – 80 263, May 2020.

[8] A. Harisinghaney, A. Dixit, S. Gupta, and A. Arora, "Text and image based spam email classification using knn, na¨ıve bayes and reverse dbscan algorithm," in *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*, February 2014.

[9] V. S. Tida and S. Hsu, "Universal Spam Detection using Transfer Learning of BERT Model," *Computation and Language (cs.CL); Machine Learning (cs.LG) arXiv:2202.03480 [cs.CL]*, 2022.

[10] S. E. Rahman and S. Ullah, "Email spam detection using bidirectional long short term memory with convolutional neural network," in *2020 IEEE Region 10 Symposium (TENSYMP), 2020*, 2020, pp. 1307–1311.

[11] C. Laorden, I. Santos, B. Sanz, G. Alvarez, and P. G. Bringas, " Word sense disambiguation for spam filtering ," *Electronic Commerce Research and Applications*, vol. 11, no. 3, pp. 290–298, May–June 2012.

[12] H. Raj, Y. Weihong, S. K. Banbhrani, and S. P. Dino, "Lstm based short message service modelling for spam classification," in *ICMLT '18: Proceedings of the 2018 International Conference on Machine Learning Technologies*, 2018, pp. 76–80.

[13] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of sms spam filtering: New collection and results," in *Proceedings of the 11th ACM Symposium on Document Engineering in DocEng'11, New York, 2011*, 2011, pp. 259–262.

[14] S. Y. Yerima and A. Bashar, "Semi-supervised novelty detection with one class svm for sms spam detection," in *Proceedings of the 2022 29th International Conference on Systems, Signals and Image Processing (IWSSIP '22)*, 01-03 June 2022.

[15] A. Ghourabi, M. Mahmood, and Q. Alzubi, " A hybrid CNN-LSTM model for SMS spam detection in arabic and english messages ," *Future Internet*, vol. 12, no. 9, 2020.

[16] S. Mishra and D. Soni, "Smishing Detector: A security model to detect smishing through SMS content analysis and URL behavior analysis ," *Future Generation Computer Systems*, vol. 108, pp. 803–815, July 2020.

[17] N. Choudhary and A. K. Jain, "Towards Filtering of SMS Spam Messages Using Machine Learning Based Technique," *Advanced Informatics for Computing Research*, vol. 712, pp. 18–30, 2017.

[18] E. M. El-Alfy and A. A. AlHassan, "Spam filtering framework for multimodal mobile communication based on dendritic cell algorithm," *Future Gen. Comput. Syst*, vol. 64, pp. 98–107, 2016.

[19] P. Poomka, W. Pongsena, N. Kerdprasop, and K. Kerdprasop, "SMS Spam Detection Based on Long Short-Term Memory and Gated Recurrent Unit," *International Journal of Future Computer and Communication*, vol. 8, no. 1, pp. 11–15, 2019.

[20] L. GuangJun, S. Nazir, H. U. Khan, and A. Ul Haq, "Spam Detection Approach for Secure Mobile Message Communication Using Machine Learning Algorithms," *Security and Communication Networks*, 2020.

[21] T. Huang, "A cnn model for sms spam detection," in *4th International Conference on Mechanical Control and Computer Engineering (ICMCCE)*, 2019.

[22] S. Gadde, A. Lakshmanarao, and S. Satyanarayana, "Sms spam detection using machine learning and deep learning techniques," in *7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2021, pp. 358–362.

[23] Kaggle. SMS Spam collection dataset. [Online]. Available: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset [Last accessed: 29 Sept., 2022]

[24] Y. Goldberg and O. Levy, "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method," *Computation and Language (cs.CL); Machine Learning (cs.LG); Machine Learning (stat.ML) arXiv:1402.3722 [cs.CL]*, 2014.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *Computation and Language (cs.CL); arXiv:1301.3781 [cs.CL]*, 2013.

[26] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[27] Gensim Python Library. [Online]. Available: https://radimrehurek.com/gensim/index.html/ [Last accessed: 25 Sept., 2022]