Detecting Hacker Threats: Performance of Word and Sentence Embedding Models in Identifying Hacker Communications

Andrei Lima Queiroz, Susan Mckeever, and Brian Keegan

Applied Intelligence Research Centre, Technological University Dublin, Ireland
{andrei.queiroz,susan.mckeever,brian.x.keegan}@tudublin.ie

Abstract. Cyber security initiatives are finding new approaches to mitigating threats against the computational infrastructure of companies. One of these approaches is the use text mining techniques and classification models to detect potentially malicious messages or posts in hacker communications. This is a difficult task due the ambiguity and the strong use of technical vocabulary inherent in such posts. This paper aims to evaluate the use of robust language models for feature representation of input to downstream classification tasks of hacker communication posts. We perform the experiment against five hacker forum datasets using a variety of language models: two Word Embeddings (Word2vec and Glove), and three Sentence Embeddings (Sent2vec, InferSent and SentEncoder). We conclude that, for this task, only Sentence Embeddings enhance the performance of SVM classification models compared to traditional language models (Bag-of-words, word/char n-grams). Additionally, we found that models using CNN improves upon SVM models by achieving 93% of positive recall and 96% of average class accuracy.

Keywords: Cybersecurity, Threat intelligence, Text Mining, Word Embeddings, Sentence Embeddings, SVM, CNN

1 Introduction

In the age of information, hackers are taking advantage of communication channels on social media for sharing security information about computational assets. They participate on these channels by either offering tools that might be used to promote attacks against computer systems or by simply exchanging information related to current hacking techniques and software vulnerabilities [1].

According to [8], hacker forums are operating a parallel economy for selling and buying malicious tools, generating an estimated revenue of at least $600 billion per year. As a result, cyber security researchers have been focusing recently on the creation of classification models that will detect whether such forum posts present a potential threat or not [18][7]. With this collected information, a Chief Security Officer (CSO) can then strategically mitigate risks and potential threats against computational infrastructure of companies.

In the creation of these models, their accuracy is tied in with finding an appropriate combination of algorithms and input features. Determining which combination brings the best performance for each different task is a matter of experimentation and evaluation [17].

For this reason, this paper aims to analyse different combinations of classification algorithms and pre-trained language models. For classification algorithms, we use Support Vector Machine (SVM) and Convolutional Neural Network (CNN) due their good performance in text classification tasks [18][20]; For language models, we use pre-trained Word Embeddings (WEMB), including Word2vec and Glove, as well as Sentence Embeddings (SEMB), with Sent2vec, InferSent and SentEncoder as our chosen models.

The embedding models are considered an evolution over the classical language models (such as Bag-of-Words (BoW) and Word/Char N-grams). A principal characteristic of embedding models is the ability to capture semantic word (or sentence) information based on capturing contextual word usage as part of the embedding training task. As a result, they achieve better detection performance in a range of downstream text analysis tasks, such as spam detection [12], abusive content detection [4] and news categorisation [20].

Our goal in this paper is to investigate the performance of embedding models in detecting hacker threats in online forums. We compare the resulting models with our previous experimental work which was performed on classic language models. We also publish the five hacker forum datasets that we have labelled and used in our work. This paper is organised as follows: In Section 2, we review works that performed downstream tasks with WEMB and SEMB in a variety of domains. In Section 3 we describe our approach, including the datasets used, methodology, algorithm and features representation used in this experiment. in Section 4, we present the results of the baseline models. In Section 5, we present the results for all configuration of models. In Section 6, we discuss the experiment results comparing them with the baseline results. Finally, in Section 7, we summarise the contribution of this work.

2 Related Work

The WEMB and SEMB models have been widely used in many different downstream task. As an example, in [11] the authors of Sent2vec have shown that their model can outperform traditional feature representation for the tasks of sentiment analysis.

Also, in [13] and [12] the models achieved better results by using Word2vec WEMB models than traditional language model, in sentiment analysis and spam detection tasks respectively. The authors in [12] also highlight that that SVM + Word2vec have achieved slightly better results compared to the CNN models. Similarly, in [20], the authors provided results of a model for news categorisation which has also used SVM + Word2vec, with this combination achieving better performance compared to Neural-networks when trained with small datasets.

However, embeddings have not improved the performance of models in all domains. As seen in [14], the authors have used Sent2vec in their model to classify adverse drugs reaction mentions. However it did not overcome their baseline model using SVM with BOW.

Within the security domain, the authors in [7] compared traditional SVM models with CNN for a similar task as set out by this paper. They conclude that traditional classifiers and features representations have highly comparable results with Neural Network based classifiers and WEMB models. In their work, they apply dataset labels using key-word matching. In our work, we present five hacker forum datasets, with a robust expert multi-labeller approach. We view the dataset labelling as a critical component for producing robust real-life models.

3 Approach

In this section we describe the datasets, methodology, learning algorithms and feature representations used in our analysis. The experiments were performed using Python 3.6.0v programming language, and Sci-kit 0.20.2v, Keras API for TensorFlow 1.15.0v as libraries.

3.1 Datasets

We refer to the five datasets as D1 to D5 respectively. They are publicly available in `http://tiny.cc/8ws67y`. They contain social media posts from surface web, deep web and dark web, including forum, micro blogs, and hacker marketplace. These posts are related to technical and personal references to computing, security, internet services, and technology. A minority proportion refer to malicious activities in software products or have mentioned security problems in software (flaws, vulnerabilities).

In the preparation of our data, we performed a labelling task in which computer science domain experts determine whether posts relate to software-vulnerability-related communication or not. Analysing the dataset, we noted that posts can be ambiguous and difficult to assign as a binary task. To address this, each message was labelled by three domains experts. The labelling tasks results in the following classes:

- **Yes**, for posts that appear as malicious posts of vulnerabilities in software assets.
- **No**, for posts not related to hacker activity or are out of the scope of our research (Data breach, copyrighted software cracked, stolen accounts and credit card accounts).
- **Undecided**, for posts that the labeller does not have enough information or confidence to mark as Yes or No.

The final label was determined by the majority-of-votes rule, reducing the risk of individual human subjectivity. In Table 1 we present examples of these posts and their final labels.

Table 1. Message examples

ID	*Message*	*Label*
MSG-1	Multiple remote memory corruption vulns in all Symantec/ Norton antivirus products, including stack buffer overflows	Yes
MSG-2	PoC for dirtycow vuln [URL]	Yes
MSG-3	Reading about lawyers argue about our Jeep hack is endless fun	No
MSG-4	it is amazing a hacker can put up with a sociologist ;)	No
MSG-5	Just released ssh_scan v0.0.10. Release notes can be found here	Undecided
MSG-6	I like sneaker's error 0xC0000156	Undecided

The MSG-1, marked as Yes, is related to a type of vulnerability (Stack Buffer Overflow) affecting a software product. Message MSG-2, also marked as Yes, is related to a release of a Proof Of concept (PoC) of a vulnerability called dirtycow. The posts MSG-3 and MSG-4 are related to personal opinion and have no direct relation to real vulnerabilities in software. Despite MSG-3 and MSG-4 having hack and hacker keywords, they are not considered malicious communication, and are thus marked as No. In MSG-5, there is not enough information to decide whether either the ssh scan tool is vulnerable or can be used against a vulnerable software. Likewise in MSG-6, we cannot confirm that the error mentioned leads to a vulnerability into the sneaker software product, thus they are marked as Undecided. We acknowledge that the model will only be as good at detecting hacker posts as the knowledge of the labellers. For this reason, labellers who understand the ambiguity and subtlety of the posts are a critical component in our work.

Finally, for this experiment, we have included the Undecided posts as positive instances since they represent a risk category of posts needing further inspection in a real-life application of the resultant model. The details and description of each dataset can be seen in Table 2 as following:

Table 2. Dataset description.

ID	*Source*	*Type*	*No. of instances*	*Distrib. (pos/neg)*	*Avg. words per msg.*
D1	Hacker Forum	Deep web	1,682	10/90%	50
D2	Twitter	Surface web	1,927	15/85%	13
D3	Marketplace	Dark Web	1,921	16/84%	169
D4	Hacker Forum	Deep web	1,966	13/87%	78
D5	Hacker Forum	Deep web	1,974	5/95%	68

[*] Available on `http://tiny.cc/8ws67y`

D1 - Cracking Arena Forum - This was one of the largest hacker forums existing in 2018 with 11,977 active users. It contains communication related to security issues in computing, which makes the data suitable to cyber security research on the interaction patterns among cyber criminals. The posts range from April 2013 to February 2018.

D2 - Twitter Security Experts - The data contains posts from 12 security-expert users on Twitter. 6 of whom are well-known-security experts with an average number of followers of 18,800, and, the other 6 are of the lesser-known security experts, with an average number of followers of 1,100. Their Tweets are mostly related to security aspects of technology, including software vulnerabilities and hacking. They have one year range from March 2016 to March 2017.

D3 - Dream Market - One of the largest marketplaces with 91,463 posted products from 2,092 sellers in 2016, this is a well-known place for selling illegal products such as illicit drugs, fake IDs, stolen credit card numbers and copyrighted software. It also advertises hacker products used in malicious hacker activities. This marketplace can be accessed only via the ToR network. The posts range from April 2013 to April 2017.

D4 - Garage4Hackers Forum - This is a medium-sized forum in terms of number of content and users. This forum contains material related to exploitation, botnets and reverse engineering, it also provides information regarding specialised hacking tools. The posts range from July 2017 to September 2017.

D5 - Cracking Fire Forum - This forum has approximately 14,511 active users. Some of the posts contains pieces of source code of a variety of languages, which is aimed to perform malicious operations, such as compromise online social media accounts. The posts range from April 2011 to February 2018.

3.2 Methodology

In the following, we describe the evaluation, metrics and sampling techniques applied to all models in this work.

Evaluation: We use 10-fold cross validation to divide our training and test data in each dataset as done in [18]. In addition, for the random partitioning of the datasets into the 10 folds, we use stratified folds, such that the ratios of positive instances to negative instances per fold matches the ratio of the full dataset.

Metrics: Our approach is to produce classification models that will assess user posts as potential threat communication or not. In this context, the impact of a false negative (FN), or non-detection of threat communication, is higher than the impact of a false positive (FP), or threat communication being detected as normal communication. Under these circumstances, our model is prioritising the classification of the positive classes (threat communication) rather than negative class (regular communication). As a result, we define the *Recall* (1) of positive classes as the principal metric.

We acknowledge that a model with high rate of FP (also known as false alarm) is not desirable either, as it implies that a model is wrongly detecting a threat where there is none. When this situation occurs, either a time-consuming expert investigation will be needed or unnecessary security actions will be taken. For this reason we also need to evaluate the *average class accuracy* (2). Additionally, this metric is suitable for imbalanced datasets as it prevents the majority class from dominating the results.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{1}$$

$$Avg.ClassAcc = \frac{Recall(pos.Class) + Recall(neg.Class)}{No.Classes} \tag{2}$$

Random Oversampling the Positive Instances: As seen in Table 2, all datasets have imbalanced class representations, with the positive class under-represented relative to the negative class. In order to solve this problem, we artificially increase the number of positive instances (training data only) as done in [18] and [3]. This is a proven method for improving recall measure. Using the *random oversampling* technique, we increase the number of positive instances using optimal proportions identified for these datasets previously [18]. After re-sampling, the new ratio of instances is shown in Table 3.

Table 3. RE-SAMPLED DATASET

	D1 (+/-) %	*D2* (+/-) %	*D3* (+/-) %	*D4* (+/-) %	*D5* (+/-) %
Before	(10/90)	(15/85)	(16/84)	(13/85)	(05/95)
After	(34/66)	(45/55)	(47/53)	(41/59)	(19/81)

3.3 Classification Algorithms

We apply two different types of classification algorithms for this experiment, one traditional (SVM) and the other a Neural Network-based (CNN). Both algorithms are commonly used for text classification, having produced high accuracy models [18][12].

Support Vector Machine (SVM): This is a supervised learning algorithm used for classification tasks and is based on the maximal margin principle. It is also known for achieving favourable performance with high dimensional data and text classification [9] [6]. SVM is no longer considered first choice for text classification in several tasks such as spam detection [12], sentiment analysis [13], online hate speech detection [3] and in our case, the detection of software vulnerability communications in hacker forums [18]. In this paper, we are using the SVM with linear kernel.

Convolution Neural Network (CNN): The architecture used for CNN in this experiment is based on [10] and optimal settings provided by [22]. For the input representation, the choice was Glove and Word2vec. As this model requires a fixed input size, we have used the length of the longest posts (in each dataset) excluding the outliers maximum size. As seen in Figure 1, this strategy presented a slight improvement over maintaining the outliers. For CNN + Glove model, excluding the outliers has improved in 3 of 5 datasets (D2, D3 and D5), and for CNN + Word2vec, it has improved in 2 of 5 datasets (D2 and D5). For D3 and D4, no visible improvement is noted.

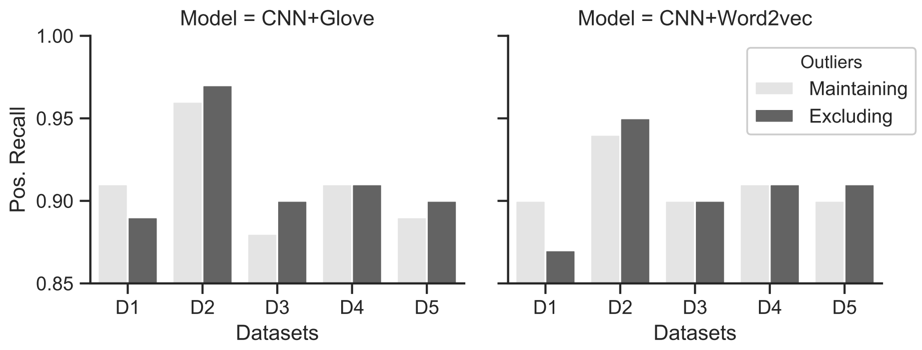

Fig. 1. Comparison of strategies for finding the fixed input length for CNN

Furthermore, we apply zero-padding to ensure that the same input length for short posts is achieved. In addition, we used *Adam* optimiser, categorical cross entropy loss function and *softmax* as an output layer. The rest of the parameters can be seen in Table 4.

Table 4. CNN parameters.

Activation. Func.	*Filter size*	*Feat. map*	*Dropout Rate*	*Regul.*	*Mini Batch*	*Epoch*
ReLu	3,4,5	100	0.5	L2	50	50

3.4 Feature Representation

Classical Language models: Bag-of-words (BoW), Word n-grams and Char n-grams are commonly used as text representation for several text mining and classification tasks. With BoW, the sentence (in our case the post) is split into a set of tokens (words), then each token is counted to produce a vector that represents the entire sentence. Words n-grams and Char n-grams are also token-based, however, the number of word/characters representing a tokens is defined by n, where $n \geq 1$. For Word and Char n-grams, we are using a range of $n = (1, 4)$ tokens. The main difference between BoW and N-grams models is that the latter encodes a degree of word of sequence information when $n > 1$. While char n-grams is better for representing rare words and morphological variants.

Word Embeddings (WEMB) models: The WEMB models we are using in this experiment are Word2vec [15] and Glove [16]. Word2vec is a model based on a three-layer neural network, while Glove is provided by the co-occurrence of words in a corpus. Different from the classical bag of words and n-grams model, these are known for mapping words together according to their semantic or syntactic similarity. However, these models suffer from the same problem as BoW. When each word in a post is translated to a vector, word sequence in the post is lost.

In order to represent the entire message of the dataset with fixed-length WEMB vector, we use a simple technique called *averaging*, which has shown positive performance compared to more complex approaches [19]. This technique consists in averaging vectors of the pre-trained WEMB model for each word of the sentence (post messages). The description of the pre-trained models used in this experiment can be seen in Table 5.

Sentence Embeddings (SEMB) models: SEMB models are categorised regarding their creation method which falls into three categories: Unsupervised, Supervised and Multi-task, with the latter being a combination of the supervised and unsupervised. In this work, we are using one pre-trained SEMB of each category, they are: Sent2vec [11] (unsupervised), InferSent [5] (supervised) and SentEncoder [2] (multi-task).

Similar to WEMB models, SEMB are known for mapping the entire sentences together according with their similarity (not only the words). However, the main improvement over WEMB is that it considers the order in which the words occur in a sentence. The description of the pre-trained models used in this experiment can be seen in Table 5.

Table 5. Summary Configuration.

Model	*Algorithm*	*Feat. Representation.*	*Pre-train. model*	*Source*	*Dim. size*
MDL-1	SVM	WEMB	Word2vec	Google News	300
MDL-2	SVM	WEMB	Glove	Common Crawl	300
MDL-3	SVM	SEMB	Sent2vec	SNLI	600
MDL-4	SVM	SEMB	InferSent	Wikipedia	4096
MDL-5	SVM	SEMB	SentEncoder	Wiki, Web News, SNLI	512
MDL-6	CNN	WEMB	Glove	Common Crawl	300
MDL-7	CNN	WEMB	Word2vec	Google News	300

4 Baseline - Experiment using classical language models

The baseline results are reported in Table 6. These results build on previous experimental work carried out in [18], for which we have extended with two more datasets (D4 and D5). To create the model, the SVM algorithm was used with three different feature representations, BoW, Word n-grams (1,4) and Char n-gram (1,4).

We see that, on average, the same score is recorded for BoW and Char n-gram, 0.75 and 0.55 of average class accuracy and positive recall, respectively. Word N-grams feature representation achieved the poorest performance with 0.68 and 0.38 of average class accuracy and positive recall, respectively.

Table 6. Baseline results using classical features representations and Support Vector Machine algorithm.

	Metric	Bag of Words	Word N-gram(1,4)	Char N-gram(1,4)
D1	Avg. acc	0.70	0.64	**0.71**
	Pos. recall	0.45	0.32	**0.47**
D2	Avg. acc	**0.88**	0.78	0.86
	Pos. recall	**0.78**	0.57	0.76
D3	Avg. acc	0.75	0.71	**0.76**
	Pos. recall	0.57	0.47	**0.60**
D4	Avg. acc	**0.70**	0.64	0.68
	Pos. recall	**0.46**	0.32	0.44
D5	Avg. acc	0.74	0.64	**0.75**
	Pos. recall	0.50	0.30	**0.52**
Avg	Avg. acc	**0.75**	0.68	**0.75**
	Pos. recall	**0.55**	0.39	**0.55**

5 Experiment using WEMB and SEMB language models

The results for each configuration of the model is described in Table 7. They are grouped into the following three categories: SVM+Word Embeddings (MDL-1 and MDL-2); SVM+Sentence Embeddings (MDL-3, MDL-4 and MDL-5); and CNN+ WEMB (MDL-6 and MDL-7). A description for each configuration can be seen in Table 5.

It was found that the best model is MDL-7 with CNN and Word2vec configuration, which achieved 0.96 of average class accuracy, 0.93 of positive recall. This performance is close to the second best model, MDL-6 with CNN and Glove, 0.95 of average class accuracy and 0.92 of positive recall. If we consider only models from the SVM+Word Embeddings category, such as MDL-1 (Word2vec) and MDL-2 (Glove), we see that they have recorded comparable results, with 0.58

and 0.62 of average class accuracy, and 0.23 and 0.33 of positive recall, respectively. It is important to notice that both have performed poorly on positive recall, although MDL-2 with Glove achieved a marginally better performance compared to MDL-1 in both metrics.

Moreover, if we consider only SVM+Sentence Embeddings models, such as MDL-3 (Sent2vec), MDL-4 (InferSent) and MDL-5 (SentEncoder), we see that MDL-5 has achieved the best result, with 0.82 and 0.74 of average class accuracy and positive recall respectively. It should also be noted that the second best in this category is MDL-3, with 0.75 and 0.60 of average class accuracy and positive recall.

Table 7. Experimental results with best metric in bold per category and per dataset

		SVM+ Word. Emb.		**SVM+ Sent. Emb**			**CNN+ Word Emb.**	
		MDL-1	MDL-2	MDL-3	MDL-4	MDL-5	MDL-6	MDL-7
D1	Avg. acc	0.54	**0.55**	0.68	0.68	**0.81**	0.95	0.95
	Pos. recall	0.09	**0.10**	0.50	0.41	**0.72**	0.91	0.91
D2	Avg. acc	0.75	**0.80**	0.80	0.86	**0.88**	0.98	0.98
	Pos. recall	0.63	**0.73**	0.70	0.75	**0.87**	**0.97**	0.96
D3	Avg. acc	0.62	**0.65**	0.81	0.78	**0.83**	0.95	0.95
	Pos. recall	0.33	**0.45**	0.71	0.63	**0.78**	0.92	0.92
D4	Avg. acc	0.54	**0.62**	0.71	0.73	**0.77**	0.96	**0.97**
	Pos. recall	0.14	**0.38**	0.57	0.52	**0.70**	0.92	**0.95**
D5	Avg. acc	0.49	0.49	0.73	0.73	**0.81**	0.95	0.95
	Pos. recall	0.00	0.00	0.53	0.49	**0.66**	0.90	**0.91**
Avg	Avg. acc	0.58	**0.62**	0.75	0.75	**0.82**	0.95	**0.96**
	Pos. recall	0.23	**0.33**	0.60	0.55	**0.74**	0.92	**0.93**

6 Discussion

Comparing the results of the experiment (Section 5) with baseline (Section 4), we note that replacing classical language models with the semantically richer WEMB does not result in a better classification performance. However, when exchanged for SEMB models, the performance is improved, which is on par with results presented in [4].

Similar to this result, we see in [21] that WEMB have not improved the performance compared to their baseline model (traditional feature representation + linear algorithm). However for sentiment analysis [13] as well as spam detection in SMS messages [12], WEMB has been shown to outperform traditional language models.

WEMB model are known for considering the semantic- and syntactic-related words in closeness vector space but do not consider the order in which a word occurs in a sentence. Whereas SEMB models inherit both of these properties. We believe that it is this property that allows SEMB models to outperform WEMB in our classification task.

7 Conclusion

Detection of potential cyber threat communication in hacker forums and social media is a hard task due the technical vocabulary and ambiguity of certain posts. In this paper, we performed an experiment with different configurations of classification algorithms and language models for detecting malicious messages related to software vulnerabilities in forums and social media. We have evaluated these models through 5 different labelled datasets (D1 to D5). With this respect, it has been concluded:

(1) SEMB features representation is the best embedding for improving classification performance on linear SVM model compared to WEMB and other traditional representations, such as, bag of words, n-gram and char n-gram. In this work, 2 out of 3 models with SEMB were able to overcome the best baseline models. MDL-3 (Sent2vec) and MDL-5 (SentEconder), achieved 60% and 74% of recall, respectively, and MDL-4 (InferSent) recorded same performance as the baseline, 55% of recall.

(2) In terms of classification performance over all models, we found that MDL-7 (CNN + WEMB (Word2vec)) is able to achieve 93% of positive recall and 96% of average class accuracy. The result is compatible with experiment done in [7], where the authors applied a CNN model to detect security communication of hackers, similar to this work. This configuration shows promise as an optimal approach for detecting posts related to software vulnerabilities. In practice, these models can be used by companies for prioritising security updates (patching) of vulnerable systems in their assets.

Acknowledgement. Andrei Lima Queiroz would like to thank the scholarship granted by the Brazilian Federal Programme Science without Borders supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), No 201898/2015-2.

References

1. Algarni, A., Malaiya, Y.: Software vulnerability markets: Discoverers and buyers. International Journal of Computer, Information Science and Engineering 8(3), 71–81 (2014)
2. Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y., Strope, B., Kurzweil, R.: Universal sentence encoder. CoRR abs/1803.11175 (2018)
3. Chen, H., Mckeever, S., Delany, S.J.: Harnessing the power of text mining for the detection of abusive content in social media. In: Advances in Computational Intelligence Systems. pp. 187–205. Springer International Publishing, Cham (2017)
4. Chen, H., McKeever, S., Delany, S.J.: A comparison of classical versus deep learning techniques for abusive content detection on social media sites. In: Social Informatics. pp. 117–133. Springer International Publishing, Cham (2018)

5. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 670–680. Association for Computational Linguistics (2017)
6. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning Journal 20(3), 273–297 (Sep 1995)
7. Deliu, I., Leichter, C., Franke, K.: Extracting cyber threat intelligence from hacker forums: Support vector machines versus convolutional neural networks. In: 2017 IEEE International Conference on Big Data (Big Data). pp. 3648–3656 (Dec 2017)
8. InfoSecurity: Evolution of the Cybercrime-as-a-Service Epidemic, uRL: https://www.infosecurity-magazine.com/magazine-features/evolution-cybercrime-service/ [Accessed: Oct, 2019]
9. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) Machine Learning: ECML-98. pp. 137–142. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)
10. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751. Association for Computational Linguistics (Oct 2014)
11. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32. pp. 1188–1196. ICML'14, JMLR.org (2014)
12. Lee, H., Kang, S.: Word embedding method of sms messages for spam message filtering. In: 2019 IEEE International Conference on Big Data and Smart Computing (BigComp). pp. 1–4 (Feb 2019)
13. Liu, H.: Sentiment analysis of citations using word2vec. CoRR abs/1704.00177 (2017)
14. Miftahutdinov, Z., Alimova, I., Tutubalina, E.: KFU NLP team at SMM4H 2019 tasks: Want to extract adverse drugs reactions from tweets? BERT to the rescue. In: Proceedings of the Fourth Social Media Mining for Health Applications (#SMM4H) Workshop & Shared Task. pp. 52–57 (Aug 2019)
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: 1st International Conference on Learning Representations, ICLR (2013)
16. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: In EMNLP. pp. 1532–1543 (2014)
17. Perone, C.S., Silveira, R., Paula, T.S.: Evaluation of sentence embeddings in downstream and linguistic probing tasks. CoRR abs/1806.06259 (2018)
18. Queiroz, A.L., Mckeever, S., Keegan, B.: Eavesdropping hackers: Detecting software vulnerability communication on social media using text mining. In: The Fourth International Conference on Cyber-Technologies and Cyber-Systems. pp. 41–48 (2019)
19. Wieting, J., Bansal, M., Gimpel, K., Livescu, K.: Towards universal paraphrastic sentence embeddings. In: 4th International Conference on Learning Representations (ICLR) (2016)
20. Wu, Z., Zheng, X., Dahlmeier, D.: Character-based text classification using top down semantic model for sentence representation. CoRR abs/1705.10586 (2017)
21. Zhang, X., Zhao, J.J., LeCun, Y.: Character-level convolutional networks for text classification. CoRR abs/1509.01626 (2015), http://arxiv.org/abs/1509.01626
22. Zhang, Y., Wallace, B.C.: A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. CoRR abs/1510.03820 (2015)