TeamDL at SemEval-2018 Task 8: Cybersecurity Text Analysis using Convolutional Neural Network and Conditional Random Fields

Manikandan R [1]*, Krishna Madgula[2], Snehanshu Saha[1,2]
[1]CAMMS, Dept of CSE ,PESIT-Bangalore South Campus
[2]PESIT-Bangalore South Campus
manikandan.ravikiran@gmail.com
krishnac.madgula@gmail.com
snehangshusaha@gmail.com

Abstract

In this paper we present our participation to SemEval-2018 Task 8 subtasks 1 & 2 respectively. We developed Convolution Neural Network system for malware sentence classification (subtask 1) and Conditional Random Fields system for malware token label prediction (subtask 2). We experimented with couple of word embedding strategies, feature sets and achieved competitive performance across the two subtasks. Code is made available at https://bitbucket.org/vishnumani2009/securenlp

1 Introduction

Cybersecurity risks and malware threats are becoming common and increasingly dangerous requiring analysis of large repositories of malware related information in realtime to understand its capabilities and mount an effective defense. The sheer volume of data and its potential applications alone have increased traction in recent times among NLP researchers. In this line, SemEval 2018 Task-8 offers 4 subtasks addressing text classification and token, relation and attribute label prediction in cybersecurity domain using MalwareTextDB (Lim et al., 2017). While subtask 1 focuses on predicting sentences relevance to malware , subtasks 2, 3 and 4 focus on predicting token, relation and attribute labels for malware text from subtask 1. More details about the each of the subtasks can be found in Phandi et al. (2018).

Concerning subtask 1, which was inherently formulated as a text classification problem very few works are done till date in cybersecurity domain (Lim et al., 2017; Zhang et al., 2016). However, in general domain the problem of text classification is well addressed with extensive usage of deep learning approaches (Zhou et al., 2016; Liang and Zhang, 2016; Kim, 2014; Kalchbrenner et al., 2014; Zhang et al., 2015), Support vector machines, logistic regression (Genkin et al., 2007; Jiang et al., 2016) and Tree based approaches (Bouaziz et al., 2014). On the other hand, subtask 2 was formulated as sequence tagging problem which is addressed till date by CRF (Finkel et al., 2005; R. et al., 2016, 2017), deep learning approaches (Chiu and Nichols, 2016; Ma and Hovy, 2016; Lample et al., 2016) and SVM (Ekbal and Bandyopadhyay, 2012).

In this paper, we describe our system that addresses subtasks 1 and 2 involving malware sentence classification and malware token label prediction. We designed these systems by adapting various insights from previous works on text classification and sequence tagging. We submitted a Convolutional Neural Network(CNN) based system based system for subtask 1 and Conditional Random Field (CRF) based system for subtask 2.

The rest of the paper is organized as follows. In section section 2, we discuss datasets and preprocessing. In section 3, we describe the algorithms and features used in the process of model development. In section 4, we describe our results and some of our findings. Finally in section 5, we conclude with summary and possible implications on future work.

2 Dataset and Preprocessing

The MalwareTextDB corpus used for this work consists of APT reports describing malware reported information taken from APTnotes[1]. We designed an end-to-end pipeline consisting on three module which process input text across multiple stages. In stage 1, the input sentence is fed to a preprocessing module which pre-processes the

*Work performed during weekend part time assistantship at CAMMS

[1]https://github.com/aptnotes/

Token	Placeholder
C:/ProgramData/Mail/	__PATH__
www.ducklink.com/	__URL__
securityblog@gdata.de	__EMAILID__
profapi.dll	__EXE__
"epsilon"	__SPECIAL__

Table 1: Tokens and placeholders used in stage 1

text for stage 2 where the sentence are subject to classification and finally stage 3 sequence tags the tokens of the input sentence. We used following preprocessing steps in stage 1.

1. All the words are lower-cased.

2. All the words that can be grouped under common category were replaced by a category placeholder as shown in table 1.

We used following opensource tools 1) Stanford Core-NLP (Manning et al., 2014) 2) Keras (Chollet et al., 2015) 3) CNTK (Seide and Agarwal, 2016) 4) Gensim (Řehůřek and Sojka, 2010) 5) NLTK for preprocessing (Loper and Bird, 2002) 6) Scikit-learn (Pedregosa et al., 2011) for grid search 7) Glove (Pennington et al., 2014).

3 Model

In this section, we explain the algorithms and hyperparameters used for system development. More specifically, in section 3.1 we explain our CNN architecture for subtask 1 and in section 3.2 we show our CRF architecture for subtask 2.

3.1 Algorithm - Subtask 1

For subtask 1, we focused more towards deep learning. Previous works (Yin et al., 2017) suggests that both Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) architectures has been successfully applied for various instances of text classification analysis at various level. With most of recent works (Zhang and Wallace, 2017) showing success of CNN, we developed a CNN architecture based on work of Kim (2014). The architecture developed in this work is as shown in figure 1.

3.1.1 Convolutional Neural Network

Our CNN architecture was derived from original works of Kim (2014) by using grid search over input channel size, number of convolution layers and number of filters. We use a multichannel model architecture with five input channels for processing

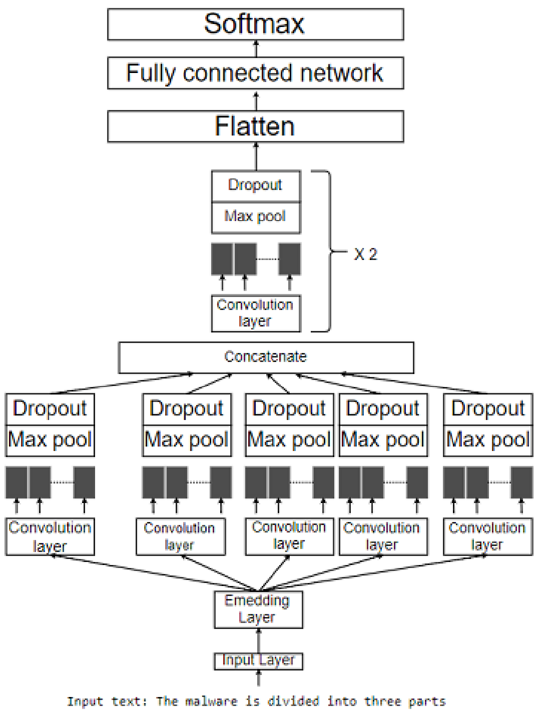

Figure 1: CNN Architecture

2-6 grams of input malware text. Each channel is comprised of the following elements:

1. Input layer that defines the length of input sequences.

2. Embedding layer set to the size of the vocabulary and 100-dimensional real-valued representations.

3. One-dimensional convolutional layer with 128 filters and a kernel size set to the number of words to read at once.

4. Channel wise Pooling layer with pool size of 5 to consolidate the output from the convolutional layer.

Following CNN, we use a Fully Connected Neural Network (FCNN) to transfer the the concatenated feature map (600 dimension) to a probability distribution over the two class labels. The number of layers in FCNN is set to be 2. The first layer uses 128 units with a tanh activation function. The second layer produces the classification probability distribution over 2 units combined with a softmax activation function.

Further to handle overfitting we use regularization via dropout (Srivastava et al., 2014) with

Feature	Value
Sentence pad length	1000
Dimensions of wordvectors	100
Number of CNN layers	8
Dimension of CNN layers	1
Number of CNN filters	128
Activation function	relu
Initialization function	Xavier
Number of FC layers	2
Dimension of 1st FC layers	128
Dimension of 2nd FC layers	2
Activation of Final layer	Softmax
Optimizer	Adam
Batch size	32
Max Epoch	10
Loss function	Cross Entropy

Table 2: Hyper parameters of CNN

threshold of 0.25. Additionally, we also apply cost sensitive learning (Zhou and Liu, 2006) in order to balance the effect of the larger negative samples present in the training dataset. For each class, we assigned weight proportional to class frequency. We implemented the neural network model using Keras. We trained our networks using Adam optimizer (Kingma and Ba, 2014). All the hyper parameters are listed in table 2.

3.1.2 Input Embeddings

We experimented with two category of word embeddings namely native embeddings and task specific embedding using Word2vec (Le and Mikolov, 2014) and Glove (Pennington et al., 2014) algorithms. Characteristics of each of the embedding is as explained below.

1. **Native Embeddings**: All words including the unknown ones that are randomly initialized use embeddings from original Word2vec/Glove models.

2. **Task specific** : The embeddings are generated by training Word2vec/Glove algorithms on sentences from MalwareTextDB.

3.2 Algorithm - Subtask 2

For subtask 2, we developed a Conditional Random Field (CRF) system (Finkel et al., 2005) based on previous works of Lim et al.(2017).

3.2.1 Conditional Random Fields

We used Conditional Random Fields with following features that is available as part of Stanford CoreNLP ToolKit.

Common Features: N-grams of size 6, previous, next tokens and labels, features giving disjunctions of words anywhere in the left or right, word shape features, word lemma of current, previous and next words, word-tag pair features, POS tags, prefix and suffixes. The description of the features are given in CoreNLP(2014).

Additional features: Based on analysis of corpus, to tackle unknown malware entities we used a gazette with token that describes malware entity. These tokens were taken from training corpus and internet[2].

	Word2vec			Glove		
	P	**R**	**F**	**P**	**R**	**F**
test17	0.47	0.77	0.58	0.48	0.78	0.47
dev18	0.18	0.32	0.23	0.35	0.80	0.18
test18	0.24	0.34	0.28	0.38	0.72	0.50

Table 3: Results of subtask 1 on native Embeddings

	Word2vec-Task			Glove-Task		
	P	**R**	**F**	**P**	**R**	**F**
test17	0.28	0.50	0.36	0.43	0.71	0.54
dev18	0.18	0.30	0.22	0.33	0.72	0.45
test18	0.20	0.30	0.24	0.38	0.67	0.48

Table 4: Results of subtask 1 on task specific embeddings

4 Experiments and Results

In this section, we present results for each of the developed systems. The original dataset was split into **train17, test-17**[3] released at the start of competition and **dev-18, test-18** released during the competition pre-evaluation period for tuning of parameters and final evaluation respectively . We submitted CNN system for subtask 1 and CRF system for subtask 2. Tables 3-5 show the results of subtasks 1 and 2 respectively across the datasets. Our systems achieve F-score of 0.5 for subtask 1 and 0.25, 0.36 for subtask 2 over strict, relaxed runs of subtask 2.

4.1 Discussion

In previous sections we described the system developed for malware text analysis using which we achieved competitive performance for subtask 1 and subtask 2.

For subtask 1, we developed a CNN system and experimented the same with different embedding strategies as explained in section 3.1.2. Across all

[2]https://www.mcafee.com/threat-intelligence/malware/

[3](train/dev/test)-(17/18) is not an official naming convention , instead used here for ease of understanding

	CRF-Strict			CRF-Relaxed		
	P	R	F	P	R	F
test17	0.51	0.26	0.34	0.45	0.36	0.40
dev18	0.18	0.25	0.21	0.38	0.22	0.29
test18	0.29	0.23	0.25	0.42	0.30	0.36

Table 5: Results of subtask 2 on Conditional Random fields

the subset of datasets, glove embeddings consistently outperformed Word2Vec embeddings. This is in line with works of Kim (2014). We initially hypothesized that since ”the context of the malware texts are different from normal English texts”, task-specific embeddings would improve the results of subtask 1. However, we observe that task specific embeddings produced lower results compared to native embeddings. Observations of results revealed high false negative predictions of non-malware texts, we believe that this may attributed to limited dataset used for developing embeddings, unlike native embeddings which was created using very large corpus. This results also agrees the general observation, that the size of the training corpus has often a greater impact on results than its strict matching with the target domain(Tourille et al., 2017).

For subtask 1, we achieved an accuracy of 0.50 and were 7% behind the top performing systems. We identified three different sources of errors across the sentences in line with previous works(Lim et al., 2017) namely misclassification of i) Sentences consisting of malware related keywords without implication on actions; ii) Sentences describing attacker actions and additionally we also found iii) misclassification of sentences containing specific patterns like presence of __PATH__ and __EXE__. Further, we had initially hoped that the multichannel architecture would prevent overfitting(Kim, 2014) and thus work better than the single channel model, especially on small datasets like MalwareTextDB. The results, however, are vice versa and hence further work on regularizing the training process and simpler single channel architecture is warranted.

For subtask 2, during analysis we found that there were multiple malware names which were previously unseen and felt only orthographic features would be insufficient. Hence in addition to commonly used features, we also included gazette features with words that quantify malware entity. However, during evaluation on development set we found high drop in precision when we used gazette features owing to its deterministic nature. Hence, we submitted CRF only with common features described in section 3.2.1 for final evaluation. With this system we achieved a result of 0.25 and 0.36 in strict and relaxed evaluation respectively. Our accuracy is 3.5% (avg) behind the top performing system across the evaluations. We identified following sources of errors i) Tagging of tokens in sentences containing only actions but not entities - these are sentences with only attackers actions in line with error from subtask 1 ii) Lack of sensitivity to context - some tokens in test document are given same label from train irrespective of context iii) Miss tagging of some of the tokens with common suffixes. For subtask 2, we experimented with simple CRF architecture with basic features, hence we believe further exploration of future engineering is needed to reduce context related errors. As far as addressing rest of the errors, we plan to explore combination of rule based and deep learning approaches.

5 Conclusion

In this work, we developed CNN and CRF systems for malware text classification and token label prediction, achieving competitive results. For subtask 1, we experimented with couple of word embedding strategies and found native glove embedding to be useful. For subtask 2, we used CRF with simple features achieving results closer to top performing system and above the official benchmark. Further, we described various sources of errors identified in the due process of analysis. In future, we plan to further improve our system to show higher performance based on the above observations.

Acknowledgments

We thank the task organizers for providing access to MalwareTextDB corpus and organizing the shared task. Further, we would like to thank various authors for open sourcing the codes of various algorithms used in this work.

References

Ameni Bouaziz, Christel Dartigues-Pallez, Célia da Costa Pereira, Frédéric Precioso, and Patrick Lloret. 2014. Short text classification using semantic random forest. In *DaWaK*.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*, 4:357–370.

François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

Stanford CoreNLP. 2014. Nerfeaturefactory. https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/NERFeatureFactory.html.

Asif Ekbal and Sivaji Bandyopadhyay. 2012. Named entity recognition using support vector machine: A language independent approach.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*.

Alexander Genkin, David D. Lewis, and David Madigan. 2007. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49:291–304.

Mingyang Jiang, Yanchun Liang, Xiaoyue Feng, Xiaojing Fan, Zhili Pei, Yu Xue, and Renchu Guan. 2016. Text classification based on deep belief network and softmax regression. *Neural Computing and Applications*, pages 1–10.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *HLT-NAACL*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Depeng Liang and Yongdong Zhang. 2016. Ac-blstm: Asymmetric convolutional bidirectional lstm networks for text classification. *CoRR*, abs/1611.01884.

Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu, and Ong Chen Hui. 2017. Malwaretextdb: A database for annotated malware articles. In *ACL*.

Edward Loper and Steven B Bird. 2002. Nltk: The natural language toolkit. *CoRR*, cs.CL/0205028.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jacob VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Peter Phandi, Amila Silva, and Wei Lu. 2018. Semeval-2018 Task 8: Semantic Extraction from CybersecUrity REports using Natural Language Processing (SecureNLP). In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Sarath P. R., Manikandan R, and Yoshiki Niwa. 2016. Hitachi at semeval-2016 task 12: A hybrid approach for temporal information extraction from clinical notes. In *SemEval@NAACL-HLT*.

Sarath P. R., Manikandan R, and Yoshiki Niwa. 2017. Hitachi at semeval-2017 task 12: System for temporal information extraction from clinical notes. In *SemEval@ACL*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Frank Seide and Amit Agarwal. 2016. Cntk: Microsoft's open-source deep-learning toolkit. In *KDD*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Julien Tourille, Olivier Ferret, Xavier Tannier, and Aurélie Névéol. 2017. Limsi-cot at semeval-2017 task 12: Neural architecture for temporal information extraction from clinical narratives. In *SemEval@ACL*.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *CoRR*, abs/1702.01923.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Ye Zhang and Byron C. Wallace. 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *IJCNLP*.

Yunan Zhang, Qingjia Huang, Xinjian Ma, Zeming Yang, and Jianguo Jiang. 2016. Using multi-features and ensemble learning method for imbalanced malware classification. *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 965–973.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *COLING*.

Zhi-Hua Zhou and Xu-Ying Liu. 2006. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18:63–77.