Flytxt_NTNU at SemEval-2018 Task 8: Identifying and Classifying Malware Text Using Conditional Random Fields and Naïve Bayes Classifiers

Utpal Kumar Sikdar[1], **Biswanath Barik**[2] **and Björn Gambäck**[2]
[1]Flytxt, Thiruvananthapuram, India
[2]Department of Computer Science, NTNU, Norway
utpal.sikdar@gmail.com
{biswanath.barik,gamback}@ntnu.no

Abstract

Cybersecurity risks such as malware threaten the personal safety of users, but to identify malware text is a major challenge. The paper proposes a supervised learning approach to identifying malware sentences given a document (subTask1 of SemEval 2018, Task 8), as well as to classifying malware tokens in the sentences (subTask2). The approach achieved good results, ranking second of twelve participants for both subtasks, with F-scores of 57% for subTask1 and 28% for subTask2.

1 Introduction

Malware is a major problem in the digital world. Recently, Lim et al. (2017) addressed the malware threat by creating a new database of malware texts. In addition, they constructed different models for identifying and classifying malware sentences, and discussed the outstanding challenges. Sutskever et al. (2016) also pointed to cybersecurity defense as a key area because of its long-term impact on society. Still, there have been very few efforts addressing the problem. Many cybersecurity agencies such as Cylance (Gross, 2016) and Symantec (DiMaggio, 2015) have collected large repositories of malware-related online texts. The diversity of those texts shows that identifying malware is quite challenging.

The organizers of SemEval 2018, Task 8 defined four subtasks for Semantic Extraction from CybersecUrity REports using NLP, SecureNLP (Phandi et al., 2018). The paper outlines a supervised approach to the first two subtasks, on malware sentence and token identification, respectively. In subTask1, given a sentence, the systems need to predict whether the sentence is relevant for inferring the malware's actions and capabilities. For this subtask, two machine learning classifiers were used, Naïve Bayes (Rish, 2001) and Conditional Random Fields (CRF, Lafferty et al., 2001), as well as a combination of the two models.

	Total		Malware	
Data	**Sents**	**Tokens**	**Sents**	**Tokens**
Train	9,435	231,180	2,204	12,165
Dev	1,213	32,029	79	459
Test	618	13,080	90	453

Table 1: Malware dataset statistics

In subTask2, the systems should predict and classify malware tokens in the sentences into three different categories, namely Action, Entity, and Modifier. A CRF-based classifier was used also for the second subtask.

The paper is organized as follows: The datasets are presented in Section 2. The malware sentence identification is described in Section 3, while the token label malware identification is described in Section 4. Results are presented in Section 5, with system comparison and error analysis reported in Sections 6 and 7, respectively. Section 8 addresses future work and concludes.

2 Datasets

The SecureNLP shared task organizers provided three different datasets: training, development and test sets. The statistics of the datasets are reported in Table 1, with the total number of sentences and tokens in each set as well as the number of those sentences and tokens containing malware.

3 Malware Sentence Identification

Two classifiers, CRF and Naïve Bayes were used for malware sentence identification. When both the classifiers identified a sentence as malware, the outputs of the classifiers were combined. The system architecture is shown in Figure 1.

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 890–893
New Orleans, Louisiana, June 5–6, 2018.

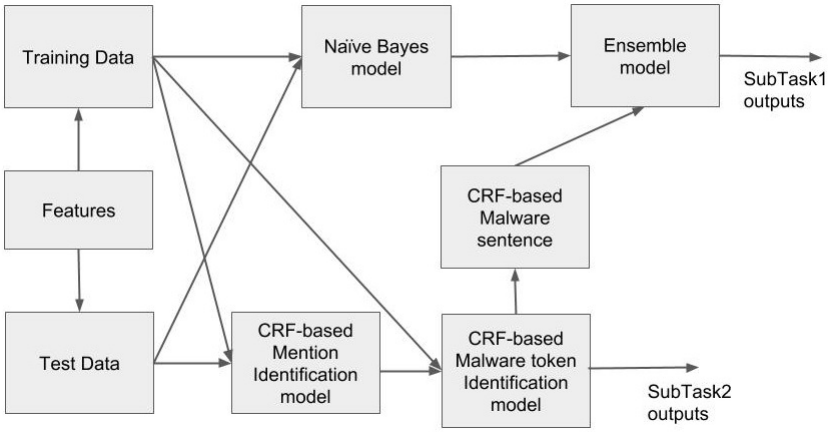

Figure 1: Overall system architecture

3.1 Conditional Random Fields

Token level malware words were identified in the texts described in Section 2. If a sentence contains malware token(s) as identified by the CRF classifier, the sentence is considered as a potential malware sentence. A range of features (further described in Section 4 below) were utilized to train the CRF classifier to predict malware tokens.

3.2 Naïve Bayes

A Naïve Bayes classifier is a probabilistic classifier based on Bayes' theorem an independence assumption between the features. As an initial step, a dictionary was created using the vocabulary found in all the sentences. In the next step, a term-document matrix was built for each sentence. Then Bayes' Theorem was applied to calculate the malware ($y = 1$) and non-malware ($y = 0$) probabilities for each sentence. Equation 1 represents the malware probability, P for each sentence.

$$P(y = 1|S) = \frac{P(S|y = 1) \times P(y = 1)}{P(S)} \quad (1)$$

Here S denotes the set of words in a particular sentence and $P(S) = P(S|y = 1) \times P(y = 1) + P(S|y = 0) \times P(y = 0)$. The non-malware probability for each sentence can be calculated in the same way. If $P(\text{malware}) > P(\text{non-malware})$, the sentence is considered to be a malware sentence, otherwise it is assumed to be non-malware.

3.3 Classifier Ensemble Prediction

An ensemble classifier was created by merging the outputs of the two classifiers described above. If both classifiers identify a sentence as malware, it is considered to be malware, otherwise non-malware. Combining the two classifiers gave better accuracy than using each classifier individually.

4 CRF-based Malware Token Identification and Classification

To identify and classify each token from unstructured text into the three categories Action, Entity and Modifier, a supervised CRF-based approach was used. The task was divided into two steps. In the first step, each token (called a mention) was identified as belonging to one of the three categories or not. In the next step, the identified tokens were classified into one of the three categories.

The CRF token label malware identification model was implemented using the C++ CRF++ package[1], which allows for fast training by utilizing L-BFGS (Liu and Nocedal, 1989), a limited memory quasi-Newton algorithm for large scale numerical optimization. The classifier was trained with L2 regularization and the following features:

- local context (with a -1 to +2 window, i.e., from one preceding to two following tokens),
- part-of-speech information (-1 to +3 tokens),
- suffix (last two or three characters)
- prefix characters (three initial characters)
- starts-with-upper-case,
- stem (-3 to +2 tokens),
- is-a-stop-word,
- is-alphanumeric,
- is-sentence-initial,
- identified mention (-3 to +3 tokens),
- bi-gram: a combination of the current token output and previous token output.

[1] https://taku910.github.io/crfpp/

System	Precision	Recall	F-score
CRF	0.30	0.80	0.43
Naïve Bayes	0.30	0.89	0.45
Ensemble	0.43	0.75	0.55

Table 2: SubTask1 Development Results

System	Precision	Recall	F-score
CRF	0.40	0.71	0.51
Naïve Bayes	0.32	0.88	0.47
Ensemble	0.49	0.67	0.57

Table 3: SubTask1 Test Results

To identifying the mentions, the above features (except the mention feature) were used together with the current word and a context consisting of the previous two and the next two words.

5 Results

The supervised learning approaches were applied to subTask1 and subTask2. The systems were learned from the training data and tested on the development data. Table 2 reports the precision, recall and F_1-score on the subTask1 development data for the CRF approach, the Naïve Bayes, and the combined ensemble approach. The ensemble achieved 10% better F-score than the Naïve Bayes approach, which in turned slightly out-performed the CRF classifier. Before evaluating on the unseen test data, the development set was merged with the training set to build the classifiers. The combined approach also produced the best results on unseen test data, as reported in Table 3. Note that the enlarged training set helped to increase precision on the test set for all classifiers, while recall went down in all cases.

For subTask2, token label malware identification, we applied the Conditional Random Fields classifier using the features given in Section 4. The results are shown in Table 4. When tested on the development data, the classifier and achieved an F-score of 24.90%, with slightly higher recall than precision. Again, the unseen test data results were somewhat better, with an F-score of 28%. Tentatively since also here the development data was merged with the training data when learning the classifier used for the unseen test data.

Data	Precision	Recall	F-score
Dev	22.22	28.32	24.90
Test	26.00	29.40	28.00

Table 4: SubTask2 Results (%)

Team Name	T1	T2	T2-rel
Villani	0.57	0.23	0.31
Flytxt_NTNU	0.57	0.28	0.36
DM_NLP	0.52	0.29	0.39
HCCL	0.52	0.22	0.38
TeamDL	0.50	0.25	0.36
NLP_Found	0.49	0.28	0.39
ACL benchmark	0.51	0.23	0.31

Table 5: Top-7 results (F-score) for SubTask-1 (T1), SubTask-2 (T2), and SubTask2-relaxed (T2-rel)

6 Comparison with Other Systems

Comparing our system ('Flytxt_NTNU') with the other systems participating in the shared task, Table 5 reports the top 7 results and shows that in subTask1 (malware sentence identification) we secured second position, while achieving the same F-score (57%) as the top-rated system (Villani). Also for token label malware identification (subTask2), our system got second position with a 28% F-score. For both subtasks, we achieved clearly better scores than the baseline system ('ACL benchmark').

7 Error Analysis and Discussion

To analyze the outputs of the development data for subTask1 and subTask2, Tables 6 and 7 draws the confusion matrices for each subtask.

For subTask1, Table 6 shows that many non-malware sentences are identified as malware sentences by both classifiers, tentatively since many common words are shared by both malware and non-malware sentences. Both classifiers generate higher recall than precision values because the classifiers try to identify as many sentences as possible as malware. Once the outputs of the two classifiers were combined (when both classifiers agreed on a sentence being potential malware), about half of the non-malware sentence classification errors were removed, and the ensemble thus produced better F-scores than the Naïve Bayes and CRF models in isolation.

	CRF		Naïve Bayes		Ensemble	
	Non-Malware	Malware	Non-Malware	Malware	Non-Malware	Malware
Non-Malware	985	149	969	165	1057	77
Malware	16	63	9	70	20	59

Table 6: Confusion Matrix for SubTask1 on the development data

	B-Entity	**I-Entity**	**B-Action**	**I-Action**	**B-Modifier**	**I-Modifier**	**O**
B-Entity	51	68	1	0	1	0	135
I-Entity	8	122	1	0	7	0	383
B-Action	2	1	57	7	0	0	57
I-Action	0	0	0	0	0	0	6
B-Modifier	0	6	0	0	28	1	44
I-Modifier	0	0	0	0	0	0	0
O	277	537	107	12	46	3	30061

Table 7: Confusion Matrix for SubTask2 on the development data

The confusion matrix for subTask2 is reported in Table 7, in the BIO (beginning, inside, outside) format for each of the three classes (entity, action, modifier). We observe that many non-malware tokens (**O**) are identified as malware and vice versa. Again, this might be due to same words occurring as both malware and non-malware tokens in the sentences, which is why the system achieved low precision and recall values. Furthermore, two of the inside mention classes (**I-Action** and **I-Modifier**) are tiny, indicating that training machine learners on them will be difficult.

8 Conclusion and Future Work

The paper has proposed Naïve Bayes and CRF based approaches to identify malware sentences (subTask1). In the future, we will incorporate other features such as tf-idf and information gain to improve system performance. Furthermore, we aim to apply deep learning-based approaches such as LSTM (Long Short-Term Memory) and CNN (Convolution Neural Network) to malware sentence classification.

For subTask2, many features were developed to identify malware tokens using Conditional Random Fields. Most of the features were extracted directly from training data, but the features could have been further optimized using grid search and evolutionary approaches. Also for this subtask, we will in the future experiment with applying other approaches, such as LSTM and CNN, to identify the types of malware tokens in the sentences.

References

Jon DiMaggio. 2015. The Black Vine cyberespionage group. Technical report, Symantec.

Jon Gross. 2016. Operation Dust Storm. Technical report, Cylance.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, Williamstown, MA, USA. IMIS.

Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu, and Chen Hui Ong. 2017. MalwareTextDB: A database for annotated malware articles. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1: Long Papers, pages 1557–1567, Vancouver, Canada. ACL.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528.

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, Louisiana.

Irina Rish. 2001. An empirical study of the Naïve Bayes classifier. In *Proceedings of the IJCAI-01 Workshop on Empirical Methods in Artificial Intelligence*, pages 41–46. AAAI.

Ilya Sutskever, Dario Amodei, and Sam Altman. 2016. Special projects. Technical report, OpenAI.