Villani at SemEval-2018 Task 8: Semantic Extraction from Cybersecurity Reports using Representation Learning

Pablo Loyola, Kugamoorthy Gajananan, Yuji Watanabe and Fumiko Satoh
IBM Research Tokyo
Tokyo, Japan
{e57095,gajan,muew,sfumiko}@jp.ibm.com

Abstract

In this paper, we describe our proposal for the task of Semantic Extraction from Cybersecurity Reports. The goal is to explore if natural language processing methods can provide relevant and actionable knowledge to contribute to better understand malicious behavior. Our method consists of an attention-based Bi-LSTM which achieved competitive performance of 0.57 for the Subtask 1. In the due process we also present ablation studies across multiple embeddings and their level of representation and also report the strategies we used to mitigate the extreme imbalance between classes.

1 Introduction

Cybersecurity represents one of the most comprehensive and challenging tasks to tackle from a data-driven perspective. It is inherently technical, covering field such as networking and programming languages, but at the same time, it considers human aspects, such as intent, trust and strategy among benign and malicious agents (Buczak and Guven, 2016).

This rich mixture makes it an ideal playground for machine learning, to extract patterns and characterize the interaction between the different set of actors involved. Moreover, as we can collect large amounts of data from security related sources, such as trace logs and reports, the level of generalization that machine learning methods could achieve could increase. Nevertheless, several challenges also emerge such as noise, lack structure, unavailability of annotated sources and a characteristic class imbalance when data is labeled. Therefore, for machine learning to be considered useful in a cybersecurity context, it must provide robust and reliable results, overcoming the aforementioned issues.

In that sense, how we represent the data plays a key role, as it is known that in any machine learning setting, different feature representations yield to different results, entangling different explanatory factors of variation on the data (Bengio et al., 2013). We are interested in study the tradeoff between the use of hand crafted features the process of automatically learning feature representations.

In that sense, the present SemEval Task 8 (Phandi et al., 2018) represents a relevant scenario to test several hypotheses in the context of a controlled semantic extraction competition. The dataset, provided by Lim et al. (2017), contains over 6,800 labeled sentences from 39 malware reports. From the subtasks, we focused on the first one, as it provides compact goal to assess a proof of concept.

Our approach consists on the use of an attention based LSTM-based recurrent architecture (Hochreiter and Schmidhuber, 1997; Luong et al., 2015) which is capable of learning sentence level feature representations at both character and token level. Additionally, we prepend an embedding layer from which pretrained feature vectors can be associated to the tokens. Given the natural class imbalance of the data, we tried several techniques to alleviate generalization issues.

Our results show that out approach can outperform the baselines, reaching up to 0.57 in the competition leaderboard for the Subtask 1. Nevertheless, the actual scores show that the task is far from being solved, illustrating the difficulty of the problem and the need for more powerful methods that allow us to obtain more expressive feature representations.

2 Data Pre-processing

We tried to keep the data as natural as possible in an attempt to retain most of the characteristics and

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 885–889
New Orleans, Louisiana, June 5–6, 2018.

nuances from the original reports. Therefore, we did not perform any transformation or cleaning on the tokens extracted from each sentence.

Given the configuration of the contest, where the leaderboard on the test set was kept hidden to the public, the only way to obtain a reliable estimation of the performance of the experiments were to look at the validation set. While this is not usually a problem, in our exploratory study, we found that the set of sentences on the provided validation set differed in distributional terms from the sentences on the provided testing set.

To alleviate this issue we implemented an adjustment procedure that consisted of merging all the sources of data (training, validation and test) into one set while re-labeling each sentence as *part to the test set* or not.

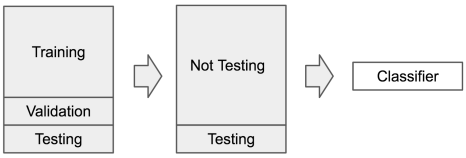

Figure 1: Data Re-classification process.

With this relabeled set, we trained a classifier to predict if a given sentence belonged to the test set or not. In this case, for feature extraction, we computed trigram based vectors for each sentence using Scikit-learn package[1], and a standard feed forward neural network was used. Figure 1 illustrates the approach.

We then considered the sentences that originally belonged to the training + validation set but that obtained the highest probability to be part of the test set by the trained model. This new set presented slightly more similar characteristics to the original test set than the original validation set, in both average sentence length and vocabulary overlap perspectives. For the empirical study, we experimented with both the original validation set and the proposed variation.

3 Proposed Approach

The problem we are trying to solve is, given a sentence from a report, estimates if it contains relevant information about malware characterization or not. Therefore, we can treat it as a binary classification.

[1]http://scikit-learn.org/

3.1 Use of pretrained feature vectors

The use of pretrained feature vectors has shown positive effects when initializing recurrent architectures (Le et al., 2015). In that sense, we provide three initialization sources.

The first one is the use of an external source represented by Glove vectors (Pennington et al., 2014). The decision of incorporating such source is related to the fact that given that Glove vectors are trained on a Wikipedia corpus, in theory they can provide a considerable level of coverage. On the other hand, we are aware that, as the corpus for this competition is highly focused on security topics, there is a chance that a portion of the token may not have a corresponding feature vector from Glove source. For this cases, we decided to initialize such tokens with a vector computed as the average of the top 10 % less frequent words appearing in Glove set.

In the second place, we wanted to test if pretraining with the same sources used for the competition could have a positive outcome. This is natural decision as we are building an ad-hoc feature representation that is narrowed to the security context. To achieve that, we computed a feature vector based on i) bigrams and trigrams combination and ii) learning a continuous vector representation by means of a word2vec (Mikolov et al., 2013) configuration. We experimented with these three sources isolated as well as combined.

Finally, in addition to a token-level feature representation, we implemented a character level feature learning pretraining that consists of a single LSTM that passes through each token and from which we capture the last hidden state to use as a token representation.

3.2 Model Architecture

The resulting sequence of vectors associated to a given sentence is then passed to a BiSTM module. In this step, as we need to obtain a learned representation of the entire sequence, we explored three configurations. The first one consists of just take the last hidden state and treat it as the sentence level representation. The second one consists of performing a simple averaging on all the hidden states obtained from the sequence. This average can be further complemented by means of element wise operations, such as direct sum or the dot product, following a schema similar to (Mou et al., 2015). The third configuration was the use

the kernel sends commands to each module using its module id .
the victim is then redirected to a url which in turn determines the best exploit to use based on the information collected .
here also it uses hashes to look up apis .
pipe server is a special mode of the injected dll .
it appears this method makes sending sms easy .

Table 1: Sample of false positives from the custom validation set.

the data is sent encrypted with rc4 , and base64-encoded .
the malicious content is stored inside the document in encoded form , and the shellcode decodes and writes this to disk .
it is designed as a survey tool .
its method of exfiltrating the logged keystrokes relied upon a hardcoded email address stored in the binary .
the communication between the attacker and the sockss is encoded using the rc4 key .

Table 2: Sample of false negatives from the custom validation set.

of an attention mechanism, as proposed by (Luong et al., 2015). In this case, a feedforward neural network is trained jointly with the recurrent module and it is in charge of learning the portions of the sentences that are more expressive for the classification.

The learned sentence representations are then passed to a binary classifier that estimates the likelihood of a sentence to be relevant or not by minimizing the categorical cross entropy loss. For all our experiments we use as optimizer Rmsprop (Tieleman and Hinton, 2012). For regularization, we added Dropout (Srivastava et al., 2014) both in the recurrent module as well as in the binary classifier.

4 Results and Discussion

The main results are shown on Table 3 and Table 4. From them, we can see that taking Glove as a baseline and incorporating n-grams vectors produce the best results. The addition of word2vec based vectors did not have a positive impact. We hypothesize this is due to the size of the training set used, the resulting vectors are based on a model that could not converge.

In the case of the configurations used to obtain a sentence representation, a clear improvement is obtained by using an attention mechanism, which outperforms both the selection of the last hidden state and all the averaging alternatives. These results are aligned with the current state of the art in language modeling, as attention is a robust way to prioritize the elements the conforms a sentence. The use of a character level representation learning module slightly improved the results, while at the same time increased the time consumption of the model in approximately 15%. Therefore, it is not clear if such addition is cost effective in all cases.

Name	Accuracy
Glove	0.612
Glove+Ngrams	0.683
Glove+Ngrams+ w2v	0.681

Table 3: Classification results for different initializations.

We tried regularization via Dropout, from 0.2. to 0.7 but it did not show relevant improvements. Another technique we implemented was to reweight the labels to mitigate the imbalance problem, but such addition decreased the overall performance of the model.

We repeated the experiments using the original train-validation schema, as well as with the variation proposed. On average using the proposed variation only increased the performance in around 2% (obtained once the test set was released) by the competition organizers.

From the results showed above, we can summarize the best configuration for the proposed task as follows:

- Initialize token vectors by means of a concatenation between Glove vectors and trigrams captured from the training set.
- The use of an BiLSTM module and an attention mechanism to efficiently weight the importance of the tokens in a sentence.

4.1 Misclassification Analysis

To obtain a better understanding of the performance of the model, we sampled a group of instances coming from both the false positive and false negative sets.

Table 1 shows a set of false positives, i.e., not relevant sentences misclassified as relevant by the

Name	Accuracy
Last hidden	0.692
Average hidden	0.694
Attention	0.822
Attention + char	0.828

Table 4: Classification results for different sentence level representation learning.

model. One of the main characteristics is the lack of specific details expressed on the sentences and the shallow descriptions of processes or tasks. On the other hand, on Table 2, the set of false negatives, i.e., relevant sentences classified as not relevant by the model, presents a denser population of specific terms, usually associated to tools or protocols used by the attackers. From this, we can hypothesize the model is not able to effectively understand the context of the token usage, especially when it is rare and probably has very low frequency. In that sense, this could represent the need for additional information the model requires treating this cases in a particular way (e.g. passing explicitly token dependencies). Further experiments are needed to find ways to operationalize such additions.

4.2 Parameter Sensibility Analysis

Given the considerable number of hyper-parameters that can be configured, we decided to study how such tuning impact model performance. During training, we found that the learning rate was key factor. While it is known that a small learning rate is beneficial when working with recurrent architectures, we found that in this case, give the extreme class imbalance, it was required to search for learning rates in a considerable lower spectrum. Figure 2 and Figure 3 show different learning dynamics sampled from several learning rates configurations. Values higher than 0.0001 produce uniform increments in the validation loss, which means the model was not able to generalize (as seen on Figure 2, training loss rapidly converged in such cases, not giving the model enough space to learn efficiently). Learning rates around 0.0001 or below started providing a decreasing validation loss. This phenomenon provides some insights on the difficulty of the task and the challenges associated when trying to achieve generalization.

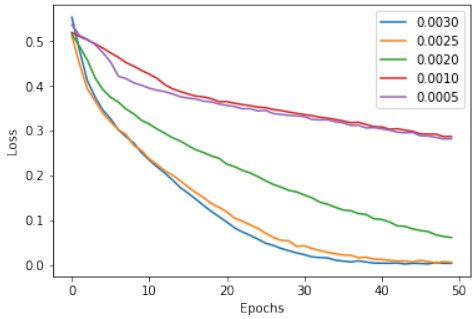

Figure 2: Impact of the learning rate on the training loss.

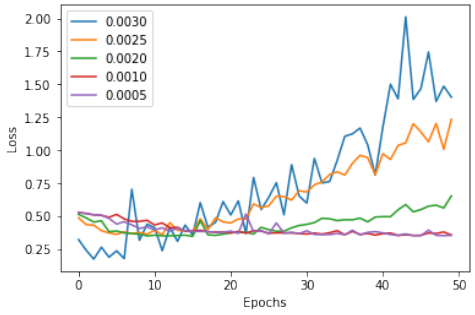

Figure 3: Impact of the learning rate on the validation loss.

5 Conclusion and Future work

In this work, we described our experience on the semantic extraction from cybersecurity reports. We were able to produce a model the generate feasible results for estimating the relevance of sentences in the context of security information. For future work, we consider that while keep improving the performance of the model is vital task, equally important is to explore ways to allow decision makers to have a clear understanding of the internals of the model to assess how much they can depend on it to support their actions. Therefore, we consider working towards incorporating an explanatory or interpretable layer in the design of a model.

References

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Anna L Buczak and Erhan Guven. 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

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)*, volume 1, pages 1557–1567.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *arXiv preprint arXiv:1504.01106*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.