UMBC at SemEval-2018 Task 8: Understanding Text about Malware

Ankur Padia[1], Arpita Roy[1], Taneeya Satyapanich[1], Francis Ferraro[1], Shimei Pan[1], Youngja Park[2], Anupam Joshi[1], Tim Finin[1]

[1] University of Maryland, Baltimore County
Baltimore, MD, 21250 USA
{pankur1,arpita2,taneeya1,ferraro,shimei,joshi,finin}@umbc.edu
[2] IBM T. J. Watson Research Center
Yorktown Heights, New York, USA
young_park@us.ibm.com

Abstract

We describe the systems developed by the UMBC team for 2018 SemEval Task 8, SecureNLP (Semantic Extraction from CybersecUrity REports using Natural Language Processing). We participated in three of the subtasks: (1) classifying sentences as being relevant or irrelevant to malware, (2) predicting token labels for sentences, and (4) predicting attribute labels from the Malware Attribute Enumeration and Characterization vocabulary for defining malware characteristics. We achieved F1 scores of 50.34/18.0 (dev/test), 22.23 (test-data), and 31.98 (test-data) for Task1, Task2 and Task2 respectively. We also make our cybersecurity embeddings publicly available at https://bit.ly/cybr2vec.

1 Introduction

Task 8 for SemEval 2018 asked participants to work on a set of related sub-tasks involving analyzing information from text about malware drawn from the Advanced Persistent Threats Notes collection (Blanda and Westcott, 2018) using the semantic framework found in the Malware Attribute Enumeration and Characterization language (Kirillov et al., 2011; Beck et al., 2014). The task was composed of four related sub-tasks that could be part of a processing pipeline for an information extraction system for cybersecurity related text (Phandi et al., 2018).

Subtask 1 required classifying a sentence as being relevant or irrelevant for inferring malware actions and capabilities. Subtask 2 involved predicting token labels for entities, actions and modifiers in sentences. Subtask 3, which we did not undertake, expanded on subtask 2 by asking participants to label relevant relations between the entities. Subtask 4 required predicting more detailed attribute labels, including ActionName, Capability, StrategicObjectives and TacticalObjectives, drawn from the MAEC vocabulary.

One of our aims is to better understand the differences between cybersecurity text and general, non-cybersecurity text; another is to also better understand differences and variation within cybersecurity texts. To that end, we focus on learning and extracting better representations of the input reports. Specifically, for our approaches, we focus on approaches that incorporate additional, domain-specific knowledge into our system, and we use these enhanced representations and features in well-studied classification, representation, and sequence prediction models.

2 Subtask 1

In this section we describe our approaches to classify a given sentence as relevant or irrelevant to malware. We used logistic regression (LR), multi-layer perceptron (MLP), and Long-Short Term Memory (Hochreiter and Schmidhuber, 1997, LSTM) as classifiers, and we used multiple encoding schemes to represent features for the classification task.

2.1 Models

We experimented with and evaluated three different techniques for implementing the Subtask 1 relevance classifier. Each approach used similar features; we opted for bag-of-words or bag-of-embeddings due to their simplicity and competitive performance (Wang and Manning, 2012).

- **Logistic Regression**: We used logistic regression, as a baseline classifier with an L2 penalty of 10.
- **Multi-Layer Perceptron**: We used two architectures for MLP for two different kinds of input; one was bag-of-words, the other

	LR			MLP		
	ACC	**AUC**	**F1**	**ACC**	**AUC**	**F1**
Avg. Binary BOW	**89.78**	**81.58**	**47.9**	**90.85**	**77.44**	**46.89**
Binary BOW	78.9	79.29	32.98	65.05	77.77	25.61
Avg. Count BOW	89.94	79.31	46.49	91.01	75.76	45.77
Count BOW	78.4	81.38	33.84	83.35	75.2	33.99
Wiki embeddings	93.49	50	0	93.49	50	0
Cyber embeddings	88.38	79.65	43.82	83.51	82.35	39.02

Table 1: Performance on Task 1 (dev-data) for each of the model we implemented. We used the best MLP model for SemEval submission (test-data) and had F1 score of 18. We found LR and MLP to always label majority class resulting in zero F1 score using Wiki embeddings.

was dense embeddings. When the input representation was bag-of-words, we used one hidden layer of dimension 32 followed by a classification layer. When embeddings were used we used a network of three hidden layers followed by a classification layer; we found adding up to 3 layers to the network helped improving training accuracy. We used L2 regularization (Tibshirani, 1996) of 0.1 and dropout (Srivastava et al., 2014) of 0.1 to avoid overfitting. We fixed the value of dropout and experimented with multiple values of L2 and chose one giving the best performance on the development dataset. Performance decreased gradually when L2 penalty was either increased to {0.2, 0.25, 0.5} or decreased to {0.01, 0.001, 0.0001}. We set the size of the hidden layer to 100 when embeddings were used.

- **LSTM (for embeddings)**: We applied an LSTM network with one hidden layer of size 128. We used dense, pre-trained embeddings (§2.2) for each word in the input sentence.

2.2 Features for the models

We experimented with the following four feature sets in order to determine the best performing representation for Task 1.

- **(Average) Count Bag-Of-Words**: We created standard bag-of-words features from the training dataset. We experimented with normalizing each vector via averaging.
- **(Average) Binary Bag-Of-Words**: We also considered binary bag-of-words features, by replacing each term frequency count with binary value where positive value is set to one and a negative value to zero. We also experimented with normalizing each feature vector by averaging.
- **Cybersecurity embeddings**: The cybersecurity embeddings were generated using word2vec Skipgram model with negative samplings of 100 dimension and a window size of five (Mikolov et al., 2013b) on one million cybersecurity related documents.[1]
- **Wikipedia embeddings**: We generated 400 dimensional word2vec skip-gram embeddings from a recent Wikipedia dump. We used a window size of 5.[2]

2.3 Datasets, embeddings & hyperparameters

For Subtask 1, we used all 65 files available as part of SemEval Task 8. We tuned and tested our model on development data available as part of SemEval. For logistic regression we swept the L2 regularization coefficient ($\{100, 10, 1, 0.1, 0.01, 0.001\}$) and chose the value that gave best performance on the development dataset. For neural approaches we used stochastic gradient descent with momentum of 0.4 for LSTM and 0.9 for MLP. We tried multiple learning rates and chose one which gave best performance on the development dataset. We chose starting learning rate of 0.2 for LSTM and 0.1 for MLP. We also tried using Adam optimizer (Kingma and Ba, 2014) with the same learning rate as MLP but found the resulting model labeled all test instances with the majority class.

For our implementation we used Keras (Chollet,

[1]We used an embedding model produced by IBM Research trained on a collection of 1 million cybersecurity-related documents with a vocabulary size of 6.4 millions words and 100 dimensions.

[2]Reported models use the 20180220 English Wikipedia dump; we did not notice large differences in performance when using this vs. an earlier version for the competition.

	LSTM		
	ACC	AUC	F1
Wiki embeddings	80.21	82.35	35.83
Cyber embeddings	**93.98**	**72.05**	**50.34**

Table 2: LSTM Performance (dev data). We offer these as supplementary evaluations.

2015) with a Tensorflow backend to train neural network based models and Gensim (Řehůřek and Sojka, 2010) to train word embeddings. We used Scikit-learn (Pedregosa et al., 2011) for Logistic Regression. For the LSTM, we let the size of the input sequence be the maximum length of all sentences in the batch and padded shorter sentences with zero vectors.

The input was a matrix of dimension $l \times d$ where d is the size of embedding vector and l is the length of the longest sentence.

2.4 Discussion

As evident from Tables 1 and 2, the neural network based classifiers perform better compared to other methods depending on the features.

However, we find a considerable gap between the score from Table 1 and 2. As explained later, we believe that the models' low scores are related to the scope of the task. Overall, the LSTM performs better compared to the MLP due its ability to capture subtle nuances.

We note the positive impact that domain-centered cybersecurity embeddings have. Nevertheless, not all cybersecurity texts may accurately reflect other cybersecurity texts, especially ones with the task-specific annotations as considered here. We posit that the performance of all our models, in particular the LSTM, may be improved with embeddings that are learned from documents more representative of those evaluated.[3]

Comparing the results of Wikipedia embeddings and embeddings trained on cybersecurity text we found Wikipedia based embeddings to consistently perform poorly. We believe one of the reasons Wikipedia embeddings performed poorly for this task is due to less overlap between the technicality of the task and content.

Moreover the F1 score is zero sometimes as the features are rich enough to classify positive instance and predicts only negative (as evident from AUC). On the other hand, the cybersecurity embeddings performed better when compared with Wikipedia embeddings, due to the more focused corpus, but we believe there is scope to improve the quality of embeddings. Frequency based features tend to perform better than binary features; averaging the features improves the performance score across all classifiers.

[3]The actual collection of APT notes included about 400 documents, vs. the 1 million documents trained on broader cybersecurity texts.

False Positive: Attackers always use this minimal effort approach in order to bypass a victim s defenses.
False Negative: Trojan.Karagany first checks for a live Internet connection by visiting Microsoft or Adobe websites.
General Information: The group has used two main malware tools: Trojan.Karagany and Backdoor.Oldrea.

Table 3: Task 1 classification examples.

2.5 Error analysis

Among the classifiers, the MLP makes mistakes by getting caught into to domain specific words that occur frequently, like *attack* and *attackers*, and skips less frequent but indicative words like *Trojan.Karagany*. Additionally we found the MLP incorrectly classifies general sentences as relevant. We demonstrate examples in Table 3.

Looking at the example sentences from 3, we see that whether or not a sentence is "relevant" is task-dependent. For example, the general information sentence above could be useful for identifying relationships among different malware instances or families. However, the sentence would be irrelevant in the context of action and capabilities of a particular malware mention.

3 Subtask 2

In this section we describe our approach for Task 2, which required participants to predict token labels for malware-related documents. The Task 2 system served as an automatic labeling tool using one of four labels:

- `Action`, referring to an malware-related event;
- Entities, referring to either `Subjects` or `Objects` in the malware-related sentence; or
- `Modifiers`, referring to prepositions that link between action and phrases.

Each label is represented by a tag using the inside, outside, beginning (IOB) format (Ramshaw and Marcus, 1999). The performance was measured using F1 score and the relaxed measurement by omitting the IOB tags.

3.1 Our Approach

We extended the previous work Lim et al. (2017), who trained a conditional random field (CRF) on unigram and bigram features of the surface words, part-of-speech tags and Brown clustering signatures (Brown et al., 1992). Like Lim et al. (2017), we also trained a CRF. Our features include:

- unigrams and bigrams of words in the dependency parse tree,
- unigrams and bigrams of the word lemmas,
- wordshape equivalence class analysis (Christopher, 2016), and
- Brown clustering signatures from a larger APT collection .

The word's context, which are words in the window of size three, was included. These features were extracted using Stanford CoreNLP (Manning et al., 2014). We did not use the surface word as in development we found it yielded lower performance. The dependency function will help to recognize the similar sentence by comparing similar sentence's structure. The wordshape features represent the classes of upper case, lower case, digits, and punctuations, and also groups the sequence of the same class. The wordshape features help to recognize named entities.[4]

We trained our own Brown clustering features (Liang, 2005) with our own APT corpus of 456 APT files from 2008 to 2017. We experimented with the Brown clustering hyperparameters: the Brown cluster size (1000,10000) and its prefix length (6,8,10,12,16). The best result from the experiment is the prefix of size 8 and cluster size 1000. We built our own Brown clustering for two reasons. First, we will not be able to identify Brown clustering feature when we encounter out of vocabulary word; we found the larger corpus to partially alleviate this concern. Second, we believed that the bigger size of the corpus, with an appropriate clustering size and prefix length, would yield better clustering features.

[4] We use the 'dan2' wordshape classes from CoreNLP (Manning et al., 2014).

	P	R	F1
Action	24.50	39.20	30.15
Entity	11.26	17.34	13.65
Modifier	29.37	46.84	36.10
Average	18.22	28.54	22.24

Table 4: Official Task 2 scores on Test set

	P	R	F1
Action	25.67	50.00	33.92
Entity	23.71	45.45	31.16
Modifier	29.92	48.10	36.89
Average	24.42	46.31	31.98

Table 5: Official Task 2 relaxed/token-level scores on Test set

3.2 Experimental Results

We used the CRF++ toolkit (Kudo, 2005) to develop our conditional random field (CRF) models. For the official evaluation, we ran our system on Test set provided by SemEval2018. The test set contains 13,080 tokens in total. The official scoring reported our F1 performance of 22 for strict scoring, and 32 for relaxed scoring. Our F1-score for subtask 2 are generally on par with the baselines (23 for the strict, and 31 for the relaxed, measures). Detailed performance analyses are shown in Tables 4 and 5.

3.3 Discussion

Table 4 demonstrates that our system performance of predicting Entity is lower than Action and Modifier. We believe this is because malware-related entities are different from other text; in particular, they can be quite long. For example, the following (gold test) entity is a long clause with complex syntactic structure: *'method of leaving the encoded file on disk and only decoding it in memories.'* This entire clause is labeled as an Entity. Despite the dependency features, our system cannot identify these long spans as an entity. Another example of this limitation is shown in Figure 1. This is a rich area for future improvement.

4 Subtask 4

In this section we describe our approach for task 4. The task is to predict attribute labels (ActionName, Capability, StrategicObjectives and TacticalObjectives) for a given malware-related text

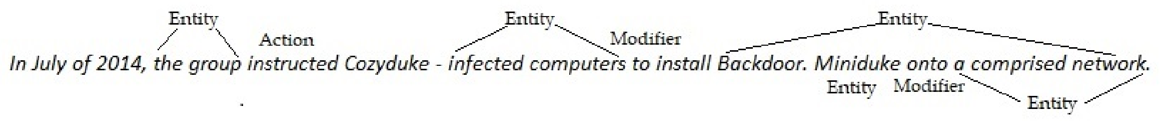
Entity
Action
Entity
Modifier
Entity
In July of 2014, the group instructed Cozyduke - infected computers to install Backdoor. Miniduke onto a comprised network.
Entity
Modifier
Entity

Figure 1: An example of wrong prediction of Task 2. Above the line is the gold standard annotation. Under the line is our predictions.

describing action of a malware.

4.1 Our Approach

For this task we focus on learning better quality word embedding features for a classifier, as classifier performance depends on the quality of its features. In addition to encoding semantics present in the text via embeddings, we want to encode domain specific knowledge in the embeddings. For this purpose, we developed an Annotation Word Embedding (AWE) model that is capable of incorporating diverse types of domain knowledge, such as metadata, keyword information, ontology knowledge, and manual annotation. The AWE model learns to predict this knowledge from the text, resulting in better quality embedding since domain knowledge can be incorporated in the embeddings. We then train a classifier with these high quality word embedding features to classify attribute labels.

4.1.1 Annotation Word Embedding

The AWE model's learning task is to predict annotations and context words given a target word. A sliding window on the input text generates the training samples. In each sliding window the task is to use target word to predict its own annotation as well as the context words. Formally, we maximize the log probability of context words and annotations given target word.

Given a sequence of T training words (W_1,W_2...W_{t-1}, W_t,W_{t+1}...W_T) and their annotations $((A_{1,1}, A_{1,2}...A_{1,M_1}), (A_{2,1}...A_{2,M_2})$ ... $(A_{T,1},...A_{T,M_T}))$, our objective is to maximize the average log probability shown in Equation 1:

$$\frac{1}{T}\sum_{t=1}^{T}\left(\sum_{-C\leq j\leq C, j\neq 0}\log P(W_{t+j}|W_t)+\sum_{0\leq k\leq M_t}\log P(A_{t,k}|W_t)\right) \quad (1)$$

where C is the size of the context window, W_t is the target word, W_{t+j} is a context word, $A_{t,k}$ is the kth annotation of target word W_t. In addition to using the target word to predict context words, like Mikolov et al. (2013a)'s skipgram model, the AWE embedding model predicts annotations of target word using target word itself.

4.2 Experiments

To train the AWE model we used all 456 APT reports as text corpus. In addition we used keywords for each attribute label described in MAEC vocabulary (Kirillov et al., 2011) and gold annotation given for 65 APT reports available as part of the SemEval task to create text annotation.

To create text annotation we collected keywords from attribute label descriptions and extracted the token groups from the gold annotations. Token groups consist of the subject, action and object linked to each other via relation labels. We used these token words and keywords to create text annotation; we deleted stop words.

For example, one token group extracted from gold annotation is "these configuration issued commands to attack following domain and IPs." After deleting stop words this token group we get "configuration," "issued," "commands," "attack," "domain," and "IPs." In the gold annotation, this token group has label Capability12 in attribute category of Capability. In MAEC vocabulary (Kirillov et al., 2011) keywords given for this capability label are "machine access," "control," "execute," "terminate," and "create." All these token words and keywords will have an annotation of Capability12 in our AWE model.

After creating the text annotation we train an AWE model with 100 dimension feature vectors, window size 5 and negative sampling. After training embeddings we use these embeddings to create features for classifier. We use average embeddings of all the words in each token group to create classifier instance. We use SVM as classifier. On the test dataset we get F-score of 0.19.

4.3 Discussion

This task is one of the most challenging tasks because of data sparsity and large number of attribute labels. In fact, out of the 444 attribute labels, 185 labels do not appear in dataset. For the remaining 259 attribute labels 169 labels occur less than five times. In addition, among 3348 instances there are 2298 instances without any ActionName attribute, 642 instances without a Capability attribute, 1244 instances without a StrategicObjective attribute and 1675 instances without a TacticalObjective attribute.

To improve classifier performance future work can try training a classifier that focuses on the common classes, with non frequent classes labeled as "other." Applying other techniques like similarity score to classify infrequent classes may also be beneficial. Additionally, we noticed that in the gold annotation there are often missing relation labels. This missing relation labels result in incomplete token group as token groups are tokens linked by relation labels.

5 Conclusion

We described the systems developed by the UMBC team for 2018 SemEval Task 8, SecureNLP (Semantic Extraction from CybersecUrity REports using Natural Language Processing). We participated in three of the subtasks: (1) classifying sentences as relevant or irrelevant for further malware analysis, (2) predicting token labels for sentences about malware, and (4) adding detained attribute labels to sentences from the MAEC vocabulary for defining malware characteristics. Our cybersecurity embeddings are available at `https://bit.ly/cybr2vec`.

We plan to continue development our systems by getting additional annotations for training, exploring the application of different machine learning algorithms, making use of the knowledge in our Unified Cybersecurity Ontology (Syed et al., 2016) and associated data, and through our ongoing collaboration with colleagues at IBM as part of the AI Horizons Network.

Acknowledgments

The research described in this paper was partially supported by gifts from IBM and Northrop Grumman. We thank Agniva Banerjee, Sudip Mittal, Sandeep Narayanan, Maithilee Prabodh, Vishal Rathod, and Arya Renjan for helping with annotations.

References

Desiree Beck, Ivan Kirillov, and Penny Chase. 2014. The MAEC language. Technical report, MITRE.

Kiran Blanda and David Westcott. 2018. APTnotes.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479.

Franois Chollet. 2015. keras. `https://github.com/fchollet/keras`.

M Bishop Christopher. 2016. *PATTERN RECOGNITION AND MACHINE LEARNING*. Springer-Verlag New York.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ivan Kirillov, Desiree Beck, Penny Chase, and Robert Martin. 2011. Malware attribute enumeration and characterization. Technical report, MITRE.

Taku Kudo. 2005. CRF++: Yet another CRF toolkit. `https://taku910.github.io/crfpp/`.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

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. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

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.

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.

Zareen Syed, Ankur Padia, Tim Finin, M Lisa Mathews, and Anupam Joshi. 2016. Uco: A unified cybersecurity ontology. In *AAAI Workshop: Artificial Intelligence for Cyber Security*.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.