Digital Operatives at SemEval-2018 Task 8: Using dependency features for malware NLP

Chris Brew
Digital Operatives
chris.brew@digitaloperatives.com

Abstract

The four sub-tasks of SecureNLP build towards a capability for quickly highlighting critical information from malware reports, such as the specific actions taken by a malware sample. Digital Operatives (DO) submitted to sub-tasks 1 and 2, using standard text analysis technology (text classification for sub-task 1, and a CRF for sub-task 2). Performance is broadly competitive with other submitted systems on sub-task 1 and weak on subtask 2. The annotation guidelines for the intermediate sub-tasks create a linkage to the final task, which is both an annotation challenge and a potentially useful feature of the task. The methods DO chose do not attempt to make use of this linkage, which may be a missed opportunity. This motivates a post-hoc error analysis. It appears that the annotation task is very hard, and that in some cases both deep conceptual knowledge and substantial surrounding context are needed in order to correctly classify sentences.

1 Introduction

The SecureNLP challenge is motivated by (Lim et al., 2017) and further described in (Phandi et al., 2018), it aims to provide automation for malware analysts who might otherwise be overwhelmed by the task of finding key data in long threat reports. The annotation guidelines used to create the task ask analysts to include actions carried out by the malware, but exclude actions carried out by the human designers of the malware. These actions are related to the MAEC cybersecurity ontology (Kirillov et al., 2010). The guidelines include one substantial caveat:

> As a general guide [for a positive annotation], the sentence should imply a particular malware action or capability, with reference to the list of attribute labels. [i.e. the MAEC labels]

Sub-task 1 calls for a determination of relevance or irrelevance to malware activity on a per-sentence basis. However a number of issues make this difficult. See detail in (Lim et al., 2017). First, it is not obvious what to do when a sentence describes malware activity but does not fit in with any MAEC category. Second, the distinction between things done by the malware and things done (or intended) by its designers is not easy to maintain.

We describe the systems that we built for tasks 1 and 2 and use them to conduct ablation studies and error analyses.

2 Digital Operatives Systems

The Digital Operatives submission used spaCy (Honnibal and Johnson, 2015). to generate features for each token, then aggregated the features from the whole sentence. To estimate performance, we used 5-fold cross-validation on the combined training and development sets.

As an example, consider the word ”ago” in the sentence:

”A few days ago we detected a watering hole campaign in a website owned by one big industrial company .”

We extract:

- the word itself
- the lemmatized form provided by spaCy,
- the orthographic shape (all lower case, represented as ”xxx”)
- the part-of-speech (”ADV”)
- the detailed part-of-speech (”RB”)
- Brown cluster (6442) (Brown et al., 1992)
- the fact that it does or does not look like a URL

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 894–897
New Orleans, Louisiana, June 5–6, 2018.

- the bigrams in which it participates ("days ago" and "ago we")
- similar bigrams for lemma, Brown cluster and shape.
- extract features from dependency links. Each token has a head from which it depends, and the relation that it holds to the head has a name. The head of "ago" is the verb "detected", and the relation is "advmod". We package this up into the feature detected-ADVMOD-ago.
- features of the form X-advmod-Y where X and Y are either both cluster ids or both orthographic shapes.

The result is passed to a passive aggressive classifier (Crammer et al., 2006)[1]. This learner, which is similar in cost and performance to a linear kernel SVM and to a number of other linear classifiers, seems to be close to the best choice from a large number of experiments. Grid search was used to choose the `C` regularization parameter for the classifier. On our 5 validation splits this method had a mean f1 of 0.60 with standard deviation 0.045. Performance on the actual test set was lower, at 0.52. This was rank 5 among the 11 submitted systems, well behind the top 2 systems and slightly ahead of the organizers' benchmark.

Our system classifies each sentence in isolation. No attempt was made to establish the referents of pronouns.

For sub-task 2 we used CRFsuite (Okazaki, 2007) to implement a linear-chain conditional random field. Per-word features were: the lowercased word and its part-of-speech tag; a two-letter prefix of the word; two- and three-letter character prefixes of the word; shape indicators for whether the word is numeric, title-case or upper-case; indicators for whether the word is beginning or end of sentence; the nltk part of speech of previous and subsequent words, if present; the shape indicators of previous and subsequent words, if present. We would have preferred to use features from spaCy, as in task 1, but did not overcome tokenization differences in time. This system was not competitive, with an F-score of 0.16 and a ranking of 9th. The best system had an F1 score of 0.39.

[1] In `scikit-learn`'s implementation (Pedregosa et al., 2011)

	precision	recall	f1-score	sup
irrelevant	0.96	0.83	0.89	528
relevant	0.44	0.79	0.57	90
avg / total	0.88	0.83	0.84	618

Table 1: F1-scores using only lemmas and dependencies between lemmas.

2.1 Ablation study

The submitted system used all the features generated by spaCy. We augmented this with ablation studies in which only subsets of the features were used. The best performance came using only unigrams and bigrams derived from lemmas, along with dependency features derived from lemmas. Ablation actually improved performance over the submission, as shown in table 1. Presumably, these features give the sparse linear classifier an appropriate level of generality. There are 89 false positives in the complete test set and 19 false negatives. There are only 71 true positives.

3 Task analysis

MAEC labels classify malware along several dimensions. We focus on a tractable subset that can be assigned to `Actions` and that describe capabilities of malware. Table 2 lists these labels.

3.1 Error analysis

We carried out our own annotation of the 112 examples from the test set for which our system did not agree with the organizers annotations. To these we added a further 111 randomly chosen examples from the test set, for which our system did agree. We assessed each of these examples against the 20 categories from table 2, There were differences of judgment between our annotations and those of the organizers. Potential reasons for this include simple errors, misunderstandings, and consequences of the linkage between sub-task 1 and sub-task 4.

Table 4 shows performance when using the new annotations. There are now 72 false positives in the complete test set and 13 false negatives. True positives now slightly outnumber false positives, at 88. It is of course possible that some of our annotations are unintentionally biased in our favor. See table 3. In future work, it may be beneficial to apply the crowdsourcing methods and careful evaluation of annotation found in, for example, (Snow et al., 2008).

Number	Capability	Example	Frequency
000	anti-behavioral_analysis	avoid detection and frustrate analysis,	36
001	anti-code_analysis	XOR 0xAA applied on top of it,	36
002	anti-detection	making its open ports invisible to scanners.	240
003	anti-removal	block access to where the rootkit keeps its file	5
004	availability_violation	DDoS attacks were launched	28
005	command_and_control	C & C proxies talk to [other] proxies	580
006	data_exfiltration	exfiltrate data back to the C&C server	189
007	data_theft	extract Internet Explorer passwords	186
008	destruction	collects file names and overwrites them	63
009	fraud	smart meters could be manipulated	7
010	infection/propagation	infected via a multi-stage attack	525
011	integrity_violation	attacker can hijack the network	85
012	machine_access/control	control the keyboard and mouse.	245
013	persistence	the malware creates a registry key	57
014	privilege_escalation	achieve admin privileges	23
015	probing	malware checks if an old versn is installed	77
016	remote_machine_manipulation	the malware will access network shares	13
017	secondary_operation	The dropper installs a second file	280
018	security_degradation	bypass User Account Control (UAC)	33
019	spying	Babar is able to sniff all keystrokes	128

Table 2: Malware capability labels. Note that the frequency distribution is highly skewed. Examples are edited to fit.

Example	DO	SN	MAEC
A screenshot of the desktop is saved into the `C:\\ProgramData\\Mail\\MailAg\\scs.jpg` file .	0	1	spying?
As you can see this powershell script simply extracts another VBScript and executes it .',	0	**1**	
Cozyduke was used throughout these attacks to harvest and exfiltrate sensitive information to the attackers .	**1**	0	exfiltration
Cozyduke will periodically contact these websites to retrieve task information to be executed on the local machine .	0	**1**	C&C
Execute contents in unlabeled textbox1 as a SQL query and return binary data to adversary.	0	1	exfiltration
The malware hides behind numerous layers of encryption and obfuscation and is capable of quietly stealing and exfiltrating sensitive information such as email from the victim's computer	**1**	0	anti-detection,data_theft
To communicate with the C&C - server , the Trojan makes use of asymmetric encryption with a hardcoded pair of private and public keys .	**1**	0	c&c

Table 3: Sample annotation disagreements. The column labeled DO reflects our classification, SN represents that given by the SecureNLP organizers. The column labeled MAEC gives detail on the capability that DO thinks is being described. When we feel confident that one of the annotations for a sentence is clearly right, it is shown in bold. If not, neither is bold.

	P	R	F1	sup
irrelevant	0.97	0.86	0.91	517
relevant	0.55	0.87	0.67	101
avg / total	0.90	0.86	0.87	618

Table 4: Performance against DO's (possibly unintentionally biased) annotations.

4 Discussion and conclusions

We suspected that the secure NLP task is difficult (Lim et al., 2017). Results bear this out:

- Our post-hoc annotation study suggests that it is indeed difficult to distinguish between things done by attackers and things done by malware.
- Often, the system described is distributed, consisting of downloaded malware, websites and C&C servers. The MAEC classification and the SecureNLP annotation guidelines emphasize measurable properties of malware samples. This puts tension into the annotation scheme and may well be a contributor to annotation errors.
- A more extensive effort using multiple annotators and reformulated guidelines could be beneficial.
- With the technology that we used, analysts relying on the classifier's judgment as a filter will still need to read approximately double the number of sentences that actually contain relevant information, and will miss 10% to 20% of the relevant material, which the classifier regards as irrelevant.

Acknowledgments

Thanks to Nathan Landon and colleagues at Digital Operatives for resources, feedback and encouragement. Particular thanks to Jordan Bryant for detailed discussions. This work was funded, in part, by IARPA's Cyber-attack Automated Unconventional Sensor Environment (CAUSE) program. Judgments and opinions are our own. Thanks to two anonymous reviewers for thoughtful suggestions on how to improve the paper.

References

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.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Ivan Kirillov, Desiree Beck, Penny Chase, and Robert Martin. 2010. Malware attribute enumeration and characterization. Technical report, The MITRE Corporation.

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. Association for Computational Linguistics.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (CRFs).

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.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, Stroudsburg, PA, USA. Association for Computational Linguistics.