Information Extraction, Information Retrieval in Cybersecurity

Sagar Patni
School of Computing, Informatics &
Decision Systems Engineering,
Arizona State University, Tempe, AZ, USA
shpatni@asu.edu

Chandni Shrivastava
School of Computing, Informatics &
Decision Systems Engineering,
Arizona State University, Tempe, AZ, USA
cshrivas@asu.edu

1. INTRODUCTION

Cybersecurity is an upcoming field in today's ever increasing technology era. It is concerned with protection of computer systems and internet services from damage and disruption caused by unauthorized third-party players. This field is growing in importance due to increasing reliance on computer systems, the internet, wireless networks like bluetooth and wifi, and due to the growth of smart devices, including smart phones and televisions and various small devices that constitute the Internet of things. With the rise of technology, all these systems are posed with an unbounded array of threats and attacks these days, that is becoming a challenge for security experts to build a system for timely risk management and prevention. Social media and magazines are platforms where we are at least notified of the various security incidents happening on a day-to-day basis all around the world. These reports are the first step in accumulating information about the various upcoming threats, their sources and target systems, the technologies they incorporate etc. that can be useful in countering the incidents in future.

1.1 Motivation

Today, there are multiple security firms working to capture information about a multitude of incidents. However, there is not currently a standard reporting format or naming scheme. As a result, two reports may appear to be very different but, in actuality, are referencing the same attacker. To better understand the attackers and discovered vulnerabilities, there needs to be a way to automatically collate the information within these reports and provide a more complete profile. In this project, we will propose a system to do this and generate a report in a standardized format.

1.2 Proposal

Our idea and approach is to collect security incident reports from different companies, merge reports for possibly similar threats and generate a standardized report that gives details about that threat.

Thee tables give an example of the type of information we are looking for.

Company	Threat Group Name
Fire Eye	APT 29
SECURELIST	Cozy Duke
CROWD STRIKE	Cozy Bear

⇓

Group Names	Cozy Bear, APT 29, Cozy Duke, ….
Target Systems	….
Incident Dates	….
Region/Country	….

2. SYSTEM ARCHITECTURE

In this section, we give a blueprint of our proposed system architecture. Figure 1 depicts the application of the trained model to a test data file. Figure 2 shows the sequence of the steps involved to train the model.

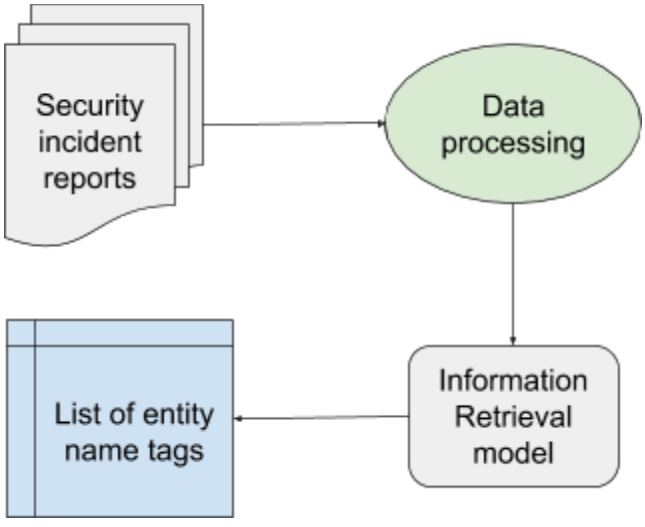

Figure 1. System working on a test report.

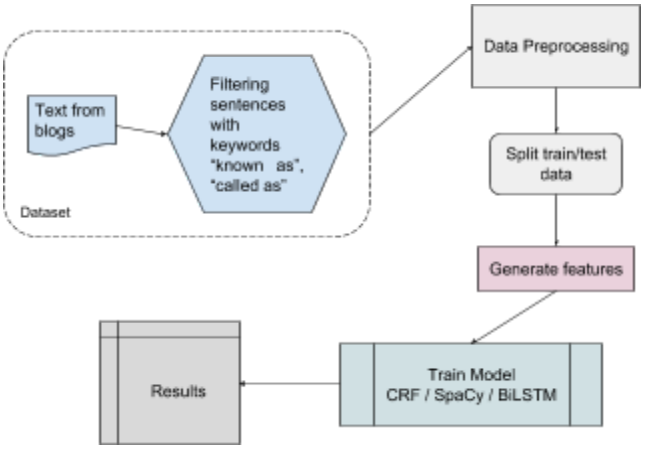

Figure 2. System diagram for training different models.

Following are the various modules that will construct our system workflow :

- Collect reports from security blogs and websites
- Extract frequently occurring keywords from each article related to threat groups (we chose to look for APT threat groups from 1-50)
- Find and combine articles that are talking about the same APT# with different nicknames
- Extract specific entities from these articles, like source organization, target systems, vulnerability IDs, dates, regions
- Create a list of this consolidated set of entities for each APT#
- Brief a short report/summary after combining these keywords

3. IMPLEMENTATION & RESULTS

Step 1 - Data Collection and Processing

For this we used Python's BeautifulSoup library to crawl few security websites and blogs. The websites that were selected were - CrowdStrike, FireEye, Symantec, Rapid7, SecureList, SecurityWeek.

Our crawler system iterated over threat keywords APT1-APT50 on all the websites to extract the content of articles that reported incident specific to a particular APT#.

Step 2 - Extracting frequently occurring terms or phrases

These are some of the Natural Language Processing tools and technologies we used to get the desired results :

1. Word2vec
2. doc2vec
3. Phrase-2-vec - finding common phrases in our dataset
4. Entity extraction - extracting noun phrases from the articles
5. Document Summarization
6. Conditional Random Field
7. Sequence Labelling with CRF and biLSTM

Following is our approach and the results we got. We have divided it into two sections based on their success.

Section 1

In this section, we list out the approaches we started with for extracting keywords related to APT groups. Our aim was to narrow down articles for same APT group based on similar keywords. These are some of the tools we used :

- Word2Vec
- Word2Vec with bigrams
- Doc2Vec
- Doc2Vec with each sentence as document
- Phrase generation
- Document summarization

➢ **Word2Vec**

The Word2Vec is an unsupervised algorithm to develop embedding for the text data. The model uses the context words to generate vector representation of words. The model uses a neural network with a hidden layer size equals to embedding dimensions required. There are two approaches to the Word2Vec training, with input to the neural network as a one hot encoding.

1. CBOW: In which the model uses context words to predict the target word.
2. Skip-gram: In which the model uses target word to predict the context words.

We applied the Word2Vec model to the entire dataset and we obtained below results,

Most similar words for apt17
('vietnamese', 0.9986779689788818)
('aware', 0.9984214901924133)
('continues', 0.9983824491500854)
('hellsing', 0.9982231259346008)
('nationstate', 0.9981635808944702)
('apt38', 0.9981043934822083)
('aligns', 0.9980053901672363)
('stone', 0.9979338645935059)
('focused', 0.9978951215744019)
('netherlands', 0.9978913068771362)

Most similar words for apt28 :
('known', 0.9992156028747559)

('bear', 0.9970523118972778)
('apt', 0.9964451789855957)
('sofacy', 0.9962831735610962)
('storm', 0.9947123527526855)
('fancy', 0.9939792156219482)
('actor', 0.993463397026062)
('tsar', 0.9934523105621338)
('pawn', 0.9927756786346436)
('ups', 0.9882858991622925)

The results are mixed and although it returned threat group names, we can clearly see that it failed to capture the phrases. We extended this model to use n-gram phrases to get better results.

➢ **Doc2Vec**

The Doc2Vec is an unsupervised algorithm used to generate the embeddings for the paragraph/document. The model extends the Word2Vec model by sending a document identifier to neural network along with the one hot representation of words. The model performs reasonably well for the document classification task.

We applied the Doc2Vec model to our dataset after consolidating reports for same threat group

➢ **Phrase2vec**

This is a variation to these two models. It uses a word2vec model to generate frequently occurring phrases in the document. Phrase discovery and embedding was done using frequency counts and pointwise mutual information, after which it was tagged and fed into the word2vec model.

➢ **Document Summarization**

In the document summarization model, we tried a different approach from what we had covered so far. Rather than extracting entities first and then documenting a report, we thought of summarizing the articles first and then extract entities from them. Document summary would contain most important sentences of an article and it would be descriptive enough to include various threat related entities. To do this we tried to implement the Luhn's algorithm to detect sentences based on frequency analysis of important keywords within those sentences. It creates a cluster of important words by computing a threshold distance between the words in a sentence. Each of these clusters were scored based on the count of significant words it contained, which eventually signified the score of the sentence. The sentences were further filtered by two approaches - (i) filter out sentences by using average score plus a fraction of the std dev as a filter, (ii) return only the top N ranked sentences. The results with both these approaches worked on some of the documents but not all of the documents we had in our dataset. We could find We got few of the good results representing the threat group names in the summarized document but it does not capture all the names.

Why these models failed ?

We observed that these models failed to capture the required results, the reason can be attributed to the foundation of these models word2vec and doc2vec models basically tries to create word embeddings by considering context words as a reference. We read the documents having threat group information such as threat names, countries and we observed that the threat group names occurs only a few times possibly single time. The document makes reference to the threat group and does not uses the reference further. Due to this, the model failed to capture the context for the threat group as most of the times, there was only a single reference.

When we ran the word2vec model on the entire dataset, we observed in the results that the model predicts the threat group Cozy_Bear being similar to the APT3. But this is not actually the right match. This happened because word2vec and doc2vec models found APT3 having a somewhat similar context as Cozy_Bear. Similar results were obtained from the doc2vec model for other APT#. When we applied word2vec model document wise, still the model failed to capture the threat names as the there are few mentions of them in the articles. The doc2vec model with each sentence as a document was somewhat useful but it was returning the results as a simple filter operation with apt names as query.

Phrase2vec could generate common phrases of varying lengths from our corpora. But it could not fetch the exact keywords that represent our entities like CVE numbers, or threat group names. Moreover, all the phrases were generated from consecutive words in the document. What we intended to get from this model was to extract predominant keywords and combine them to generate phrases. The phrases we got as a result were not informative enough to proceed with the next steps.

The document summaries were descriptive in some way, but this did not work for all the articles. Some of them had higher frequency of unrelated sentences that although were related to the context of topic but didn't include the keywords.

Possible extension to the model

Though the word2vec and doc2vec models failed to capture exact results, the embeddings created by them captures the context in the document and can be used in neural network models such as CNN and LSTM. In the

section-2 we extend these models with some neural network approaches.

Section 2

In this section we propose named entity recognition approach to the problem. We treat our problem as a sequence labelling problem which can be solved using machine learning techniques or neural network approaches. The section includes information about the data collection, data preprocessing and feature engineering performed in order to create these models.

➢ Main idea

The main idea of the section is to learn the model on our dataset to perform named entity recognition task. The pretrained named entity recognition model from spacy, nltk doesn't solve the purpose as they are trained on specific corpora such as CoNLL or google news and cannot capture the dependencies present in the our dataset and using transfer learning with them will required lot of sentences

➢ Data Filtering/ Data Collection

After manually going through the documents we figured that the reference to the threat group comes with the keywords such as 'known as', 'attributed to', 'called as', 'a.k.a.' etc. As in the test examples there can be attribution to the threat names using the phrases similar these. We extended our filter vocabulary with words similar to these from wordnet We filter our data set with such keywords and apt group names that resulted into total of 40 sentence which has the threat names along with other sentences.

The data set has 3 columns, First is origin whether its from security blog or normal sentence all the normal sentences have value 'test' for this column. The second is actual sentence. And the third contains actual entities, we manually retrieved these entities from the dataset.

To train any machine learning model such small dataset can lead to underfitting, in order to tackle that problem we retrieved sentence from 'http://sentencedict.com/wordQueryDo.php' which has similar sentence structure to our dataset. After retrieving such sentences we manually name tagged them with the entities. So in total our dataset has 200 sentences out of which 40 comes from the actual threat blogs and remaining comes from the web search.

➢ Conditional Random Field

The Conditional Random Field is a statistical machine learning technique which can be used for structured prediction and prediction sequence. The model is a discriminative classifier which can create decision boundary between different classes.

To train the CRF model we model the conditional distribution as follows,

$$\hat{y} = argmax_y P(y|x)$$

Conditional Distribution

And we define our feature function as,

f(X, L{i-1}, L{i+1})

Where the X is the current word and L{i-1} and L{i+1} represents the features of previous and next word.

The loss function for the CRF model can be defined as a negative log likelihood function

$$L(y, X, \lambda) = -log\{\prod_{k=1}^{m} P(y^k|x^k, \lambda)\}$$

The model can be useful for natural language processing tasks such as Part of Speech Tagging, Named Entity Recognition and predicting next words in the sentences. In Natural Language Processing Named Entity recognition tasks, The model uses context information from surrounding words and try to predict the label to the selected words.

➢ Feature Generation/ Model Training

Our feature vector contains the information regarding current word such as part of speech tag and information about the word such as if its lower word or its title case or a digit. And similar information for the previous and next word.

```
'bias',
'word.lower=' + word.lower(),
'word[-3:]=' + word[-3:],
'word[-2:]=' + word[-2:],
'word.isupper=%s' % word.isupper(),
'word.istitle=%s' % word.istitle(),
'word.isdigit=%s' % word.isdigit(),
'postag=' + postag
```

```
‘-1:word.lower=' + word1.lower(),
'-1:word.istitle=%s' % word1.istitle(),
'-1:word.isupper=%s' % word1.isupper(),
'-1:word.isdigit=%s' % word1.isdigit(),
'-1:postag=' + postag1
```

```
'+1:word.lower=' + word1.lower(),
'+1:word.istitle=%s' % word1.istitle(),
```

```
        '+1:word.isupper=%s' % word1.isupper(),
        '+1:word.isdigit=%s' % word1.isdigit(),
        '+1:postag=' + postag1
```

To train the model we splitted our dataset in to 80% train data and 20% test data. We used python-crfsuite module, The module provides simple interface to train and test our dataset.

```
trainer = pycrfsuite.Trainer(verbose=True)

for xseq, yseq in zip(X_train, y_train):
    trainer.append(xseq, yseq)
```

➢ **Results**

After running the model for named entity recognition for 200 iterations and , we were able to correctly identify the entities in our test database. Some of the results for our test sentences are as below,

although (O)
the (O)
groups (O)
exact (O)
motives (O)
remain (O)
unclear (O)
its (O)
initial (O)
tranche (O)
of (O)
information (O)
exposed (O)
individuals (O)
connected (O)
to (O)
longrunning (O)
gothic (GROUP_NAME)
panda (GROUP_NAME)
apt3 (GROUP_NAME)
operations (O)

apt3 THREAT_GROUP also known as ups THREAT_GROUP
gothic THREAT_GROUP panda THREAT_GROUP tg-011
THREAT_GROUP is a sophisticated threat group that has been active since at least 2010

Where 'GROUP_NAME' is the entity of our choice. Below table represents the precision recall scores for our model,

	precision	recall	f1-score	support
O	0.92	0.97	0.94	210
GROUP_NAME	0.83	0.67	0.74	52
avg/total	0.90	0.91	0.90	262

Deep Named Entity Recognition

In paper 'Named Entity Recognition with Bidirectional LSTM-CNNs' Jason P.C. Chiu, Eric Nichols explains an approach to Named Entity Recognition using LSTM and CNN, we extend this model to recognize threat group names in cyber security articles.

We used the Word2Vec embedding created in previous section to encode the sentences. We created architecture similar to that of presented in the original paper. The model contains character in a word, casing of the character, whether the character contains digit or not as a feature vector. The feature vector is somewhat similar to one that we used in CRF model.

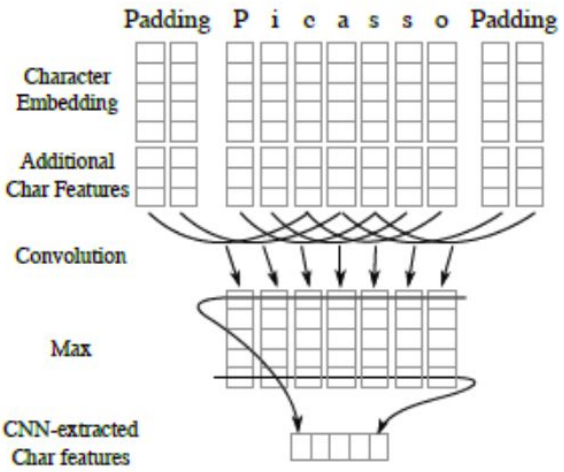

Figure 3. The Deep Learning model Named Entity Recognition with Bidirectional LSTM-CNNs

We used keras framework to develop the deep model, The model takes three types of input 1. Character level embedding which represents index of each character in a word 2. Word level embedding from Word2Vec 3. Numerical Information about the casing of the letters.

The model then passes all the character level embeddings through 1-D convolution layer with 30 filters followed by a 1-D max pooling layer. Finally the model flattens the character level vectors. It concatenates all the vectors char, word, case and pass it through a Bi-LSTM layer.

The output of the LSTM is a vector of length 400 finally we apply an Dense Layer which has number of neurons equal to the number of sequence labels.

Layer (type)	Output Shape	Param #	Connected to
char_input (InputLayer)	(None, None, 52)	0	
char_embedding (TimeDistributed	(None, None, 52, 30)	2850	char_input[0][0]
dropout_7 (Dropout)	(None, None, 52, 30)	0	char_embedding[0][0]
time_distributed_13 (TimeDistri	(None, None, 52, 30)	2730	dropout_7[0][0]
time_distributed_14 (TimeDistri	(None, None, 1, 30)	0	time_distributed_13[0][0]
words_input (InputLayer)	(None, None)	0	
casing_input (InputLayer)	(None, None)	0	
time_distributed_15 (TimeDistri	(None, None, 30)	0	time_distributed_14[0][0]
embedding_10 (Embedding)	(None, None, 100)	22000	words_input[0][0]
embedding_11 (Embedding)	(None, None, 8)	64	casing_input[0][0]
dropout_8 (Dropout)	(None, None, 30)	0	time_distributed_15[0][0]
concatenate_4 (Concatenate)	(None, None, 138)	0	embedding_10[0][0] embedding_11[0][0] dropout_8[0][0]
bidirectional_4 (Bidirectional)	(None, None, 400)	542400	concatenate_4[0][0]
time_distributed_16 (TimeDistri	(None, None, 2)	802	bidirectional_4[0][0]

Total params: 570,846
Trainable params: 548,782
Non-trainable params: 22,064

Figure 4. Represents the Layers of the deep learning model.

We trained the neural network for 50 epoches and computed scores for the model, The result of our model are as follows,

Prec: 0.961
Rec: 0.961
F1: 0.961

We found that due to limited test data, The learning is limited and the model overfits and performs very poorly on test sentences. The accuracy of the model can be improved by providing more data during training. We improved the performance by reducing number of parameters which are prescribed in the original paper.

4. CONCLUSION

Named Entity Recognition can be effectively applied to retrieve required keywords from set of documents. We found that the model precisely apply the name entity tags to previously unseen keywords as well. The Word2Vec and Doc2Vec models are not effective when the dataset is smaller and the keywords appear only a few times in the document.

5. FUTURE WORK

The dataset for the project was limited. Even though we included data from search engine, it's still not enough to train the models.

- Approaches such as using pre-training/ semi-supervised learning can be employed.
- The models can be further extended to perform named entity tagging to countries, tools and techniques used by the groups.
- The results can be further improved by training the model on the larger datasets.
- The word embeddings created in 'section 1 implementation' can be effectively used to train models such as Bi-LSTM with CRF, CNN with CRF to further improve the accuracy.
- Additionally, these advanced models can be applied on top of the summarized documents we got earlier.

6. ACKNOWLEDGEMENT

We would like to express our gratitude to professor Chitta Baral for the valuable guidance throughout the semester. We would like to express special thanks to Mr. Kazuaki Kashihara for giving us the golden opportunity to do this project. His mentorship and motivation were essential to carry out the project.

7. REFERENCES

Samuel J. Perl, 2017, Improving Data Extraction from Cybersecurity Incident Reports [Blog Post], Network Situational Awareness, SEI Insights, CMU

Luis Rocha, 2017, Extract And Use Indicators Of Compromise From Security Reports [Archive], Digital Forensics And Incident Response, Security Monitoring, Threat Intelligence

Ravendar Lal, 2013, Information Extraction Of Cyber Security Related Terms And Concepts From Unstructured Text (Master of Science Thesis). Retrieved from https://ebiquity.umbc.edu/_file_directory_/papers/680.pdf

Arnav Joshi, Ravendar Lal, Tim Finin, Anupam Joshi, 2013, Extracting Cybersecurity Related Linked Data from Text, 2013 IEEE Seventh International Conference

Russell, Matthew A., 2013, Mining the Social Web, 2nd ed, O'Reilly Media, Chapter 5: Mining Web Pages: Using Natural Language Processing to Understand Human Language, Summarize Blog Posts, and More

Robert A. Bridges, Corinne L. Jones, Michael D. Iannacone, Kelly M. Testa, John R. Goodall, 2013, Automatic Labeling for Entity Extraction in Cyber Security, Cyber & Information Security Research Group

Arnav Joshi, Ravendar Lal, Tim Finin and Anupam Joshi, 2013, Extracting Cybersecurity Related Linked Data from Text, University of Maryland, Baltimore, MD, USA

Sonit Singh, 2018, Natural Language Processing for Information Extraction, Macquarie University, Australia

Christopher Manning, Information Extraction and Named Entity Recognition, Introducing the tasks: Getting simple structured information out of text, lecture slide from class CS124, Stanford University

Steven Bird, Ewan Klein, and Edward Loper, 2009, Natural Language Processing with Python --- Analyzing Text with the Natural Language Toolkit, 1st ed, O'Reilly Media, Chapter 7: Extracting Information from Text

Kavita Ganesan, 2018, How to incorporate phrases into Word2Vec – a text mining approach