Graphs are Documents: Predicting TCP Connection Attack via Weighted Jaccard Similarity

Minseok Choi
Graduate School of AI
KAIST
Daejeon, South Korea
minseok.choi@kaist.ac.kr

Sanghyeon Lee
Graduate School of AI
KAIST
Daejeon, South Korea
shlee6825@kaist.ac.kr

ABSTRACT

Were there any malicious attacks? TCP connection attack prediction is a task of detecting attacks that may have occurred in the given connection histories. This is rather a difficult task because we do not know the number of attacks contained in each set of histories. While this may look like a graph problem, we convert these graph data to documents and compute similarities between each pair to identify patterns of a certain type of attack. For a similarity measure, we introduce a new, modified version of Jaccard similarity that is capable of handling skewed, imbalanced data. Our novel approach performs far better than random guesses, as well as vanilla Jaccard similarity, indicating that graphs can be processed as documents.

1. INTRODUCTION

TCP (Transmission Control Protocol) is a connection-oriented protocol that establishes and maintains network conversations through which computers and application programs send packets of data to each other. With the progression of the Information Age, the size of networks connecting the world is exponentially growing, which calls for the protection of these virtual networks.

In our given task, we attempt to predict the type of attacks that may have occurred in the network based on past TCP connection histories. While there may be various ways to approach this problem, we view it in a slightly different perspective and consider this task as finding similar documents. Specifically, given training documents, which could be represented as graphs, we compute similarities with each training document. If the training file contains some kind of attack, we hypothesize that a similar pattern of connection histories will also be appearing in different files that had the same type of attack. We believe that this approach is not far-fetched, as many areas in various fields employ the principle of document similarity to check plagiarism.

There exist numerous methods for finding similarities between documents in the literature; however, the most prominent similarity measure is perhaps the cosine similarity. Documents are first transformed into vectors in some way (e.g. TF-IDF [3], Doc2Vec [5], BERT [1]), and then cosine similarities between these vectors demonstrate how close they are in the vector space. Nevertheless, this way of approach assumes the documents are in *text*, and such method may not be suitable for our given task, which is mostly in *numbers*. Therefore, we incorporate Jaccard similarity [cite] as our main measure, which does not require vectorization. By simply finding the intersection of the two sets of documents, we implemented an algorithm that detects a similar pattern as a certain type of attack.

In addition to the vanilla Jaccard similarity, we modify the similarity measure by giving weights to the detection of each type of attack. This way, the model adjusts the class imbalance in the dataset, classifying the labels more accurately.

The contributions of this paper are the following:

- We introduce a new perspective by approaching graph data as documents for solving TCP connection attack prediction task.
- We propose a novel way of predicting TCP connection attacks using weighted Jaccard similarity.
- Our approach performs far better than random guesses and has potential to expand and be improved upon.

2. PROPOSED METHOD

This section explicitly describes the process of how we predict TCP connection attacks given only the past TCP connection histories. At the end of the section, we present the details of applying weighted Jaccard similarity to the given dataset.

2.1 Shingling

Shingling is a process of converting documents to sets. Because documents can be large and hard to fit in main memory, a sequence of k tokens, called k-shingle, that appears in the document is used to represent the document. For instance, if a document contains letters *abcab*, the set of 2-shingles is $\{ab, bc, ca\}$. In the given dataset, rows of the documents are in the form of *source ID*, *destination ID*, *port number*, *timestamp*, and *type of connection* (if it is a training file). Then, when $k = 2$, a pair of source ID and destination ID is converted to a shingle but ignoring the rest of the data.

2.2 Jaccard Similarity

After converting each file to shingles, we apply Jaccard similarity to find the relationships between each document. Formally, Jaccard similarity J between two documents D_1 and D_2 is computed as

$$J = \frac{|S(D_1) \cap S(D_2)|}{|S(D_1) \cup S(D_2)|} \tag{1}$$

where S is the set of shingles for the document. Each validation document D_v is compared with each training document D_t, and

.

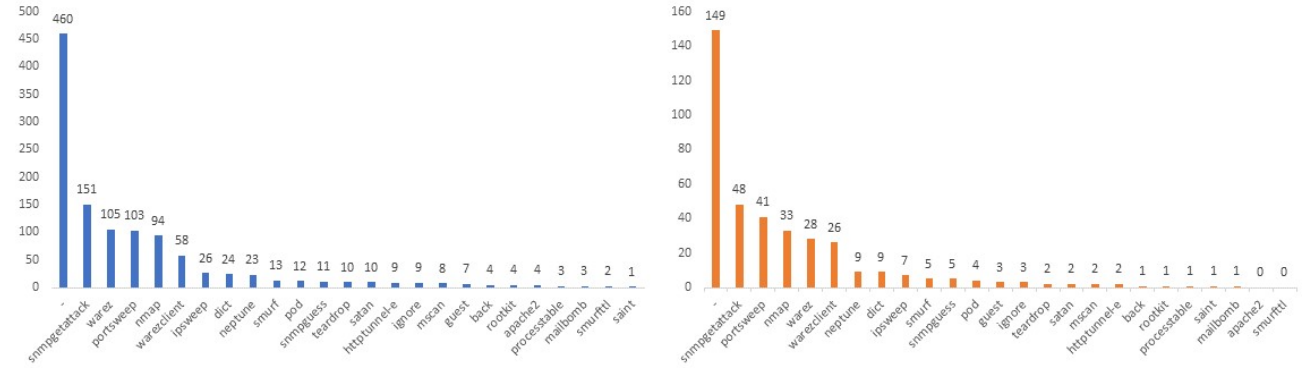

Figure 1: Distribution of the number of documents per each type of attack for training set (left) and validation set (right).

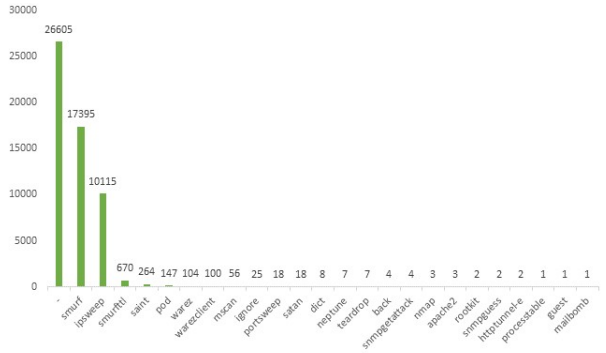

Figure 2: Number of unique 2-shingles per class.

all of the training labels in which the similarity with the validation document is above a certain threshold s are used to predict the label of D_v.

2.3 Weighted Jaccard Similarity

Although the vanilla Jaccard similarity works in most cases, we realized that the dataset is very skewed and has a class imbalance problem. Excluding the benign connection (symbol -), frequencies of the top 5 attacks, `snmpgetattack`, `warez`, `portsweep`, `nmap`, and `warezclient`, seem to be dominant over the rest of the classes in both training and validation sets, as shown in Figure 1. Despite the dominance, however, the number of unique shingles for each class does not reflect the appearances of each class in the documents, as shown in Figure 2. This causes an issue with Jaccard similarity because if two comparing sets has only a small intersection of `snmpgetattack` shingles, which has merely 4 unique shingles, in a relatively large union set, it will yield a low similarity, failing to detect the attack.

To mitigate such problem, we tweak the similarity measure by multiplying weights to each intersection of the type of attack. Formally, weighted Jaccard similarity of J_w between two documents D_1 and D_2 is computed as

$$J_w = J_1 + J_2 + \cdots + J_C$$
$$J_c = \frac{w_c \cdot |S(D_1(c)) \cap S(D_2(c))|}{\sum_{i \in C} w_i \cdot |S(D_1(i)) \cup S(D_2(i))|} \quad (2)$$

where J_c is the weighted Jaccard similarity of class c, w_c is the weight of the class c, and $S(D(c))$ is the set of shingles for class c in the document D. All J_c's are then summed up to yield the weighted Jaccard similarity J_w. Ultimately, we hope to adjust weights in a way that our model emphasizes heavily on classes with the less number of unique shingles.

Although there may be optimal parameters for the weights, we deemed that finding the analytical solution is difficult. Furthermore, because the similarity threshold s in which we employ to predict the labels is a hyperparameter, our model optimizing the weighted F1-score is indifferentiable and cannot be trained in a traditional machine learning fashion. Therefore, we created our custom pseudo F1-score F_1^p where we replace the prediction labels with the predicted scores for each label and the ground-truth labels with a one-hot vector of length C, the number of classes. The predicted scores for each label are max-pooled from each Jaccard similarity score J_c, and our predicted F_1^p is optimized with a mean-squared loss against the ground-truth F1-score of value 1.

Additionally, we believe that giving negative weights to Jaccard similarity makes no sense, so we clamped learned weights to 0.01 every time before we trained our model. We then fine-tune our hyperparameters with the Adam optimizer [4] to converge to locally optimal weights.

3. EXPERIMENTS

3.1 Experimental Setup

First, we standardize our dataset, as the training set and validation set are in a different format. Given training files in `train` directory, we convert each file to shingles, and the set of all labels in a file becomes one list of labels for the file. For validation files in `valid_query` directory, the same step is taken, but the labels are read from `valid_answer` directory. For hyperparameters, we tune the number of shingles k, similarity threshold s, and learning rate lr.

3.2 Results

Table 1 shows the weighted F1-scores against the validation set. Vanilla Jaccard similarity performed its highest at threshold $s = 0.3$ with 0.775, while the weighted Jaccard similarity reached its best at threshold $s = 0.6$ with 0.785. This illustrates that the weighted Jaccard similarity outperforms one without weights by effectively

Threshold	Vanilla	Ours
0.1	0.764	0.226
0.2	0.774	0.411
0.3	**0.775**	0.647
0.4	0.765	0.766
0.5	0.763	0.779
0.6	0.756	**0.785**
0.7	0.745	0.766
0.8	0.744	0.760
0.9	0.738	0.757

Table 1: Weighted F1 Scores for each similarity threshold.

learning the weights. By applying learned weights, we have empirically showed that Jaccard similarity can be trained to fit the data better than the original one.

4. CONCLUSIONS

Predicting TCP connection attacks can be viewed as a graph problem, where IP addresses are represented as nodes and connections are edges. However, we wanted to find out if we could detect these attacks without any graph attributes. Therefore, we treated the graph data as a type of document and computed similarities between each pair to discover a certain overlap of patterns in connection histories. Our model using weighted Jaccard similarity performed far better than random guesses, as well as the vanilla Jaccard, demonstrating that graphs can be represented as a certain kind of document. Moreover, because our model is capable of shingling, it can adapt to massive data by expanding it to various algorithms, such as min-hashing and locality-sensitive hashing [2]. Nevertheless, our approach did not consider the temporality of the data, as well as the port number, which could be important factors when determining the type of attack. Incorporating such metadata may be an interesting future work of representing graphs as documents.

5. REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[3] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[5] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

APPENDIX

A. APPENDIX

A.1 Labor Division

The team performed the following tasks:

- Vanilla Jaccard Experiment [Minseok]
- Weighted Jaccard Experiment [Sanghyeon]
- Progress Report [Minseok, Sanghyeon]
- Final Report [Minseok, Sanghyeon]
- Slides [Minseok, Sanghyeon]
- Presentation Video [Minseok]