Unveiling the Type of Relationship Between Autonomous Systems Using Deep Learning

Tal Shapira  
School of Electrical Engineering  
Tel-Aviv University  
talshapira1@mail.tau.ac.il

Yuval Shavitt  
School of Electrical Engineering  
Tel-Aviv University  
shavitt@eng.tau.ac.il

Abstract—The ToR inference problem had been widely investigated in the last two decades, mostly using heuristic algorithms. In this problem, we attempt to reveal the economic relationships between ASes, data with applications in network routing management and routing security.

In this paper, we introduce a novel approach for ToR classification, which is based on embedding the AS numbers (ASN) in high dimensional space using neural networks. Similar to natural language processing (NLP) models, the embedding represents latent characteristics of the ASN and its interactions on the Internet. The embedding coordinates of each AS are represented by a vector; thus, we call our method BGP2VEC. In order to solve the supervised learning problem presented, we use these vectors as an input to an artificial neural network and achieve a state of the art accuracy of 95.2% for ToR classification.

Index Terms—Deep Learning, Internet, BGP, AS relationships, AS embedding

I. INTRODUCTION

The Internet consists of thousands of Autonomous Systems (ASes), each AS operated by an administrative domain such as an Internet Service Provider (ISP), a business enterprise, or a University. Each autonomous system is assigned a globally unique number, the Autonomous System Number (ASN), and advertises (announce) one or more IP address prefixes (APs) using the Border Gateway Protocol (BGP). BGP routing’s update messages list the entire AS path to reach an AP. For each AP, BGP allows each AS to choose which routes to accept (import policy), how to select the best route, and whether to announce it (export policy).

The commercial agreements between two connected ASes are broadly classified into three types of relationship (ToR) [1]: 1) Provider-to-customer (P2C) - the customer AS pays the provider AS for transit traffic from and to the rest of the Internet, 2) Peer-to-peer (P2P) - two ASes freely exchange traffic between themselves and their customers, but do not exchange traffic from or to their providers or other peers, and 3) Siblings (S2S) - two ASes that belong to the same administrative domain. Gao [1] defined concatenation rules for AS links in a route that model the way ASes usually configure their BGP, it is called Valley Free (VF) since once a route descend from a provider to a customer it cannot ascend again. An interesting observation from the VF model is that connectivity does not imply reachability, and the shortest path in the (undirected) AS graph may not be usable due to the BGP VF constraints.

ToR information allows us to infer the possible routes selected by BGP, e.g., in case of a link failure [2]. It can also be used to identify malicious fiddling with the routing system, known as IP hijack attack [3], [4]. However, ToR information is mostly not public, and thus there is a long line of research to infer it [1], [5], [6], [7], [8]. Most of these solutions are heuristic algorithms based on publicly available BGP announcement databases [9], [10]. An inherent problem in these algorithms is their use of heuristics, causing unbounded errors that are spread over all inferred relationships.

Over the past few years, advances in deep learning [11] have driven tremendous progress in many fields; one of them is Natural Language Processing (NLP). We build on the excellent results achieved for NLP tasks (Word2Vec [12]), were word adjacency in sentences is used to map words to a large dimensional space. Instead, we use adjacency of ASNs in BGP announcements to embed ASes in a large (we selected 32) dimensional space and attach to each AS a vector of its coordinates in this space. Based on the ASN embedding, we apply artificial neural networks for the ToR classification problems.

Our approach achieved excellent results: we classify AS ToRs with an accuracy of 95.2%. As far as we know, we are the first to solve this problem using deep learning methods. We should also mention that the embedding also allows us to classify ASNs, which is outside the scope of this paper.

The rest of the paper continues as follows. After describing related work in Sec. II, we describe the datasets in Sec. III. In Sec. IV we describe our two-stage method, which is based on ASN embedding, i.e., BGP2VEC, and applying artificial neural network for the classification task. Sec. V presents our experiments and their results with comparison to previous results. Finally, the last section concludes the paper.

II. RELATED WORK

There are many works that focused on solving the problem of inferring Autonomous Systems (ASes) type of relationships (ToR), most of them proposed heuristic algorithms based on extracting information from BGP announcements or based on generating AS level Routes from traceroutes. We will focus here on works which are based on BGP route information, and...
disregard works that leveraged other information to improve ToR inference, such as usage of BGP communities or IXP route servers [13].

Gao [1] was the first to study the AS relationships inference problem. She presented heuristic algorithms that infer AS ToRs from BGP routing announcements based on the fact that a provider’s AS graph degree is usually larger than its customers, and that peers have about the same degree. The algorithm locally identifies the top provider for each path and classifies edges (ToRs) following the valley-free nature of routing paths.

Subramanian et al. [5] introduced the ToR maximization problem, which is to label all the edges in an undirected AS graph, in order to maximize the number of valley-free paths in a set of BGP routes. Their algorithm exploits the structure of partial views of the AS graph, as seen from different locations. For each location, it calculates the rank of each AS using a reverse-pruning algorithm, and infer the ToR between two ASes by comparing their vectors of ranks; if the ranks are similar, the algorithm classifies the link as P2P, otherwise as C2P.

Xia and Gao [14] used the BGP Community Attribute, the AS-SET object, and the routing policies in the IRR Databases to infer AS relationships. However, their approach only obtains partial AS relationships (14% of total AS pairs on Oct. 2003). They showed that both GAO [1] and SARK [5] inferred poorly P2P relationships.

Battista et al. [15] proved that the ToR optimization problem [1], [5] is NP-complete, and reduced the problem to the ToR-D problem that allows a small number of invalid paths. They reduced the ToR-D problem to 2SAT and introduced a heuristic algorithm for determining the ToRs. Cohen and Raz [6] defined the Acyclic Type of Relationship (AToR) problem that attempts both to minimize the number of invalid paths and keep the directed graph acyclic. They introduced a heuristic algorithm to solve the K-AToR problem.

Dimitropoulos et al. [7] used the IRR [10] to infer S2S relationships and then introduced a more realistic problem formulation that accepts that AS paths do not always exhibit a hierarchical pattern to infer P2C and P2P relationships. Their algorithm introduced a metric called reachability, sorted all ASes by their reachability, and grouped ASes with the same value into levels. They correctly inferred 96.5% C2P, 82.8% P2P, and 90.3% S2S relationships.

Shavitt et al. [8] were motivated to reduce the usage of heuristics. They proposed a near-deterministic algorithm for solving the ToR inference problem (ND-ToR), that uses the Internet’s core (a sub-graph of top-level ASes), which was constructed in three different ways: the Greedy Max Clique (GMC) core [16], the k-Core which is based on the k-shell decomposition [17], and the CAIDA Peers Core (CP) which is the largest connected component of a P2P graph (provided by CAIDA [18])- that contains some of the largest tier-I ASes. They inferred the rest of the links using a three-phase algorithm based on the valley-free rule, and the k-shell index [17] of the adjacent ASes. Their algorithm succeeded to infer over 95% of approximately 58,000 ToRs based on AS-level paths collected from RouteViews [9] and DIMES [19].

Luckie et al. [20] introduced the AS-Rank algorithm for inferring C2P and P2P links using BGP data. Their work relies on three assumptions: 1) there is a clique of large transit providers at the top of the hierarchy, 2) most customers enter into a transit agreement to be globally reachable, and 3) cycles of C2P links should not exist for routing to converge. Based on these assumptions, they introduced a new algorithm for inferring the customer cone of an AS, which is the set of ASes that the AS can reach using P2C links, and achieved state of the art results.

In order to overcome the inference barriers for hard cases, such as non-valley-free routing, limited visibility, and non-conventional peering practices, Jin et al. [21] identified key interconnection features and developed a probabilistic algorithm (ProbLink), and showed that their algorithm achieved an error rate that is better than AS-Rank over their validation set. However, they use additional information, such as sibling relationships, BGP communities, and IXP information.

As mentioned in [14], [7], [8], [20], [22], the existing heuristic algorithms rely on assumptions such as the presence of valley-free paths, the existence of a peering clique of ASes at the top of the hierarchy and more. This highly motivated our work for introducing a deep learning based approach that relies only on the characteristic of the data (BGP routes). As far as we know, this is the first time deep learning is used for this problem. As we will show, our deep learning method produced significantly better results than previous rule-based and heuristic algorithms.

### Table I: Number of labeled ToRs in the dataset.

<table>
<thead>
<tr>
<th>CAIDA AS ToRs serial-2</th>
<th>P2P</th>
<th>C2P</th>
<th>P2C</th>
</tr>
</thead>
<tbody>
<tr>
<td>608,486</td>
<td>113,400</td>
<td>118,405</td>
<td></td>
</tr>
<tr>
<td>118,405</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### III. The Datasets

In our experiments, we use data that was collected in March 2018. We use two types of datasets:

1) **RouteViews’s BGP announcements (RV)** [9] - contains BGP path announcements collected from 19 route collectors. The dataset consists of approximately 3,600,000 BGP paths, 62,525 AS vertices, and approximately 113,400 undirected links. We use this unlabeled dataset for the first stage of our approach (i.e., ASN embedding).

2) **CAIDA AS Relationships Dataset** [23] - provides two datasets: serial-1, which contains 343,952 P2P and 118,405 P2C/C2P pairs, that were inferred from BGP paths using AS-Rank [20], and serial-2, which contains additional 264,534 P2P pairs that were inferred from BGP communities attributes using the method described in [13]. The total number of samples is displayed in Table I. We use this labeled dataset as labels for training our neural network and various supervised learning algorithms based on ASN embedding, and as a benchmark for...
comparison with previous works. Note that the CAIDA
dataset does not contain siblings.

IV. METHOD

Our method works as follows: first, using a shallow neural
network, we map each ASN to an embedded vector. Then,
for the ToR classification task, we activate Artificial Neural
Network (ANN) that receives the vectors from the previous
stage. In this section, we will introduce in details both stages.

A. ASN Embedding

Applications of neural networks have expanded significantly
in recent years [11]. One of them is embeddings, a method that
is used to represent discrete variables as continuous vectors.
This technique is broadly used in the field of Natural Language
Processing (NLP), also known as word embeddings [12], [24],
which helps machine learning algorithms to achieve better
performance by grouping similar words. Embeddings are
important for input to a neural network, as it is trained to work
on vectors of real numbers. Moreover, the method produces
vectors such that similar words have close vectors, where
similarity is defined in terms of both syntax and semantics.

As mentioned in Sec. I, in the first stage, we produce a 32-
dimensional continuous vector representation for each ASN.
As in the training process of word embedding in NLP, i.e.,
Word2Vec [12], we train our network over a large corpus of
AS paths (described in Sec. III), which are equivalent to the
sentences in NLP tasks. We apply a similar skip-gram model
as introduced in [12], such that for each ASN in a certain AS-
path we predict ASNs within a certain range before and after
the current ASN, (as shown in Fig. 1). As a result, the model
learns to characterize an ASN by its context, i.e., neighboring
ASNs.

Our model contains an input layer of size 62,525 (the num-
ber of distinct ASes in our dataset, denoted by V), one fully
connected (FC) hidden layer whose size is the embedding size
N (using a grid search method, we found that an embedding size
of 32 achieves best results, see Sec. V-B), and an output
layer whose size is determined by the window size. The hidden
layer weight matrix is of a size VₓN, such that each ASN in
the corpus corresponds to an N-features vector; these are the
ASN-vectors that are learned by the model. The output layer
is the softmax layer [25], whose size is V for each desired
output.

We choose to apply a window of size 2 (see Figure 1),
which is the maximum distance between the input ASN and
a predicted ASN (the output) within an AS path, which
results with an output layer with a maximum size equals
to 4ₓ62,525. In order to improve the representation, we use
negative sampling [12], to distinguish the target ASN from
the noise distribution using 5 negative samples for each target
ASN. We build and run our network using the Gensim [26]
library.

The training procedure is done by feeding the network with
the ASN pairs; the input is a one-hot vector representing the
input ASN and the training outputs, which are also one-hot
vectors representing the output ASNs (the context ASNs).
Then applying gradient descent learning [27] (also known as
back-propagation) to adjust the weights of the network in order
to maximize the log probability of any context word given
the input word.

Table II: The architecture of our artificial neural network.

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Input/Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embeddings:</td>
<td>Input: 2, 1 Output: 2, 32</td>
</tr>
<tr>
<td>Conv1D:</td>
<td>Output: 2, 32</td>
</tr>
<tr>
<td>MaxPool:</td>
<td>Output: 2, 16</td>
</tr>
<tr>
<td>Conv1D:</td>
<td>Output: 32, 16</td>
</tr>
<tr>
<td>MaxPool:</td>
<td>Output: 16, 16</td>
</tr>
<tr>
<td>Fully Connected:</td>
<td>Output: 100</td>
</tr>
<tr>
<td>Softmax:</td>
<td>Output: 3</td>
</tr>
</tbody>
</table>

B. ANN Architecture

For the ToRs classification problem, we choose to use a
simple ANN (see Table II) comprised of seven layers, not
counting the input. A sequence of ASNs is fed into the first
layer of the network, which is an embedding layer. Each ASN
is embedded into a 32-dimensional vector based on the first
stage. The next layer is a 1-dimensional convolutional layer
[28] (labeled as ConV1D) followed by ReLU [29] activation
function with 32 filters of length 3 and a total number of 3,104
trainable parameters (3072 weights and 32 bias parameters).
The next layer is a max-pooling layer with 32 feature maps
of size 2, where each unit in each feature map outputs the
maximum value of 2 neurons in the corresponding feature
map in ConV1D. The next layer is a second ConV1 layer
(with 224 trainable parameters) followed by a second max-
pooling layer. The next layer is a fully-connected layer with
100 neurons and a ReLU activation function (with 25700
trainable parameters). Finally, our output layer is the softmax
layer with 3 outputs, one for each class. The last layer contains
303 trainable parameters.

The training of the neural network is done by optimizing the
categorical cross entropy [30] cost function, which is a
measure of the difference between the softmax layer output
and a one-hot encoding vector of the same size, representing
the correct label of the sample. For the optimization process,
we use the Adam [31] optimizer, which is an extension to
the stochastic gradient descent algorithm. We use the default
hyper-parameters as provided in Kingma et al. [31] and set
our batch size to 64.

We build and run our networks using the Keras [32] library
with Tensorflow [33] as its back-end. We use 80% of the
samples as a training set and 20% of the samples as a test set.
We split each dataset such that the ratio between the quantities
of the classes remains the same in both the training set and
the test set, while there is no ToR in the training set in which
its inverse appears in the test set. We run our network for 40

Authorized licensed use limited to: TEL AVIV UNIVERSITY. Downloaded on August 14,2020 at 17:16:03 UTC from IEEE Xplore. Restrictions apply.
epochs of the training set. During the test time, our network classifies a ToR in an average time of 0.1 milliseconds.

V. EXPERIMENTS AND RESULTS

In this section, we report our experimental results by a comparison of our ToR classification to best known previous results. Moreover, we will show that our method had found many mistakes in the CAIDA dataset, which emphasizes the strength of our results.

Due to a lack of space, we omit experimental results that demonstrate the strength of the ASN embedding. We can attest that the ASN embedding captures many latent characteristics of the ASNs. For example, the closest ASN to AS3356 is its sibling AS3549, and the next 3 nearest neighbors are other tier-1 providers: Telia, Verizon, and KPN. All the 5 nearest neighbors of IUCC (The Israeli universities' network, AS378) with high cosine similarity scores are also Educational/Research ASes from Europe and the Middle East with a small degree. None of these ASes are connected directly.

Table III: A comparison of the ToR classification accuracy and recalls.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CAIDA AS Relationships Dataset (serial-1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GAO</td>
<td>38.4%</td>
<td>1.0%</td>
<td>99.1%</td>
<td>85.7%</td>
</tr>
<tr>
<td>SARK</td>
<td>81.9%</td>
<td>16.1%</td>
<td>95.4%</td>
<td>95.2%</td>
</tr>
<tr>
<td>NDToR CP-CORE</td>
<td>89.0%</td>
<td>99.9%</td>
<td>70.8%</td>
<td>71.2%</td>
</tr>
<tr>
<td>NDToR TS-CORE</td>
<td>56.2%</td>
<td>54.7%</td>
<td>33.3%</td>
<td>82.3%</td>
</tr>
<tr>
<td>NDToR Kmax-CORE</td>
<td>84.6%</td>
<td>12.3%</td>
<td>99.5%</td>
<td>99.4%</td>
</tr>
<tr>
<td>RUAN</td>
<td>78.6%</td>
<td>2.3%</td>
<td>97.8%</td>
<td>97.7%</td>
</tr>
<tr>
<td>BGP2VEC - NN</td>
<td>94.2%</td>
<td>89.0%</td>
<td>93.1%</td>
<td>98.5%</td>
</tr>
<tr>
<td>BGP2VEC - LR</td>
<td>81.6%</td>
<td>93.7%</td>
<td>51.1%</td>
<td>49.0%</td>
</tr>
<tr>
<td>BGP2VEC - SVM</td>
<td>76.5%</td>
<td>89.1%</td>
<td>58.6%</td>
<td>57.7%</td>
</tr>
<tr>
<td>BGP2VEC - KNN</td>
<td>92.6%</td>
<td>94.0%</td>
<td>90.0%</td>
<td>90.1%</td>
</tr>
<tr>
<td>BGP2VEC - KMeans</td>
<td>61.6%</td>
<td>79.8%</td>
<td>26.0%</td>
<td>43.6%</td>
</tr>
<tr>
<td>CAIDA AS Relationships Dataset (serial-2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BGP2VEC - NN</td>
<td>95.2%</td>
<td>98.0%</td>
<td>88.4%</td>
<td>87.8%</td>
</tr>
</tbody>
</table>

A. Evaluation Criteria

We use the accuracy criteria to evaluate our model performance, which is defined as the proportion of examples for which the model produces the correct output of all predictions made. A formal definition of the accuracy for multiclass classification is

\[ \text{Accuracy} = \frac{\sum_{i \in \text{classes}} TP_i}{\sum_{i \in \text{classes}} (TP_i + FP_i)}, \]

where \(TP_i\) and \(FP_i\) are the true positive and the false positive of the class \(i\), respectively. Moreover, for each class we also calculate the recall, defined by \(Rc = \frac{TP}{TP + FN}\) (where \(FN\) is the false negative).

B. Hyper-parameters Optimization

Figure 2 shows the results of a grid search over different ASN embedding sizes (\(V\)) and window sizes, in order to optimize these parameters to achieve the best accuracy for ToRs classification. We performed a grid search for window sizes of 1, 2, and 3; and for embedding sizes in powers of 2 ranging from 2 to 512. In each experiment, we used the same architecture as depicted in Sec. IV, with only one modification, which is the output of the embedding layer.

The results show that embedding sizes greater than 8 and different window sizes give similar results for the ToRs classification problem. Since increasing the embedding size increases dramatically the number of parameters in the neural network, we select an embedding size of 32 and a window of 2.

Figure 2: Grid search results for the ToRs classification’s accuracy as a function of the embedding size (\(V\)) and the window size of the BGP2Vec.

C. ToR Classification Results

Table III compares BGP2VEC results to other previously suggested algorithms based on our CAIDA AS Relationship serial-1 dataset, which contains AS relationships inferred from BGP using the method described in [20]. In our comparison, we focused only on methods that are based on AS-level paths, thus do not include the PTE [14] algorithm. Shavitt et al. [8] observed a problem when executing the heuristic phases for inferring P2P (and S2S) relationships of the AToR [6] and BPP [15] algorithms. Thus, we compare only the P2C and C2P assignments of both algorithms. In our experiments, the AToR algorithms succeeded to accurately infer 93.7% and 98.1% of the C2P ToRs and P2C ToRs, respectively, while BPP infer only 83.7% and 68.2% of these ToRs.
In this paper, we introduce a novel approach for numerical characterization of ASes using deep learning methods and characterizing AS communities using the method described in [13]. Despite the large percentage of sibling ToRs, this may hint that many of our misclassifications are attributed to siblings.

D. Testing BGP2VEC Embedding Performances Using Different Machine Learning Algorithms

In order to emphasize the strength of the BGP2VEC embedding, we apply ‘classic’ supervised and unsupervised machine learning algorithms for the ToRs classification problem. Table III summarizes our results using a 64-dimensions embedding-vector for each Tor, by concatenating together two 64-dimensions embedding-vectors, one for each AS. We tested two basic supervised learning algorithms, Logistic Regression (denoted as LR) and Support Vector Machine (SVM), which achieve accuracies of 81.6% and 76.5%, respectively. As can be seen in Table III, both methods struggle to classify C2P/P2C ToRs, which is mainly due to the imbalanced dataset (for example, by applying weighted loss to balance the imbalances in the data, we achieve similar recalls for both P2P, C2P, and P2C ToRs).

In order to understand the strength of the AS-pairs similarity, for each AS-pair we generate a distance-embedding, i.e., we subtract the embedding of the second AS from the embedding of the first AS and get a 32-dimensions distance-vector. Then we apply the K-Nearest Neighbours (KNN) algorithm over the distance-vectors (denoted as D-KNN in Table III), which choose the ToR which achieves a majority vote among the 5 nearest neighbors. The D-KNN achieves an accuracy of 92.1%, which is the best accuracy achieved over the symmetric dataset. We also apply a KNN algorithm using the concatenated 64-vectors and achieve an accuracy of 92.6%.

Figure 3 displays the number of neighbors with the same ToR, based on the KNN algorithm with K=5 over the symmetric test set for; P2P, C2P, and P2C combined. As can be seen, 83.0% of the 5-neighbors ToRs are identical, which means that the distance-vectors that achieved by the BGP2VEC embedding characterize well the ToRs.

Last, we apply the K-Means unsupervised-learning algorithm. Then we determine the class of each cluster by applying a majority vote. We achieve a best accuracy of 61.6% with K=10. This result shows that although similar ToRs are located close to each other (as can be concluded by the KNN results), they are not spatially arranged in clusters.

In summary, our deep learning method achieves state of the art results in a relatively short evaluation time for ToRs classification. Moreover, we show that ToRs can be inferred using simple machine learning algorithms based on BGP2VEC embeddings.

VI. CONCLUDING REMARKS

In this paper, we introduce a novel approach for numerical characterization of ASes using deep learning methods and characterizing AS communities using the method described in [13]. Despite the large percentage of sibling ToRs, this may hint that many of our misclassifications are attributed to siblings.
apply it to achieve state of the art results for AS Type of Relationships (ToRs) classification. Our solution consists of two stages: learning dense representations of ASNs from BGP routes (BGP2VEC), and applying ANN using the ASN embeddings as inputs for the classification. As far as we know, we are the first to employ deep learning for this problem.

We had tested our algorithm on the CAIDA AS Relationship dataset and found it to perform very well, with 95.2% accuracy. Manual inspection showed that of the 4.8% of the misclassifications, almost half were correct and are due to errors in the CAIDA dataset. By comparing our method with previous works, our method achieves the best performance, 5.2% higher than the second-best algorithm. Moreover, our method found mistakes in the labeled dataset, which demonstrates the strength of our results.

This work is the first in this direction, and we plan to use it as a building block for other problems, such as detecting hijacked routes from BGP announcements. We also plan to explore the ASN embeddings further to reveal the latent characteristics of ASes.

ACKNOWLEDGMENT

This research was funded in part by a grant on cyber research from the Israeli PMO, and by the Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University.

REFERENCES


Figure 3: Number of neighbours with the same ToR based on KNN with K=5 over the symmetric test set for; P2P, C2P, P2C and combined.


NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium