7-2015

A Machine Learning Approach to Edge Type Prediction in Internet AS Graphs

Jinu Susan Varghese
Iowa State University

Lu Ruan
Iowa State University, ruanlu@iastate.edu

A Machine Learning Approach to Edge Type Prediction in Internet AS Graphs

Jinu Susan Varghese and Lu Ruan
Department of Computer Science
Iowa State University
Ames, Iowa, USA
jsusan@iastate.edu, ruanlu@iastate.edu

ABSTRACT

The Internet consists of a large number of interconnected autonomous systems (ASes). ASes engage in two types of business relationships to exchange traffic: provider-to-customer (p2c) relationship and peer-to-peer (p2p) relationship. Internet AS-level topology can be represented by AS graphs where nodes represent autonomous systems (ASes) and edges represent connectivity between ASes. While researchers have derived AS graphs using various data sources, inferring the types of edges (p2c or p2p) in AS graphs remains an open problem. In this paper we present a new machine learning approach to edge type inference in AS graphs. Our method uses the AdaBoost machine learning algorithm to train a model that predicts the edge types in a given AS graph using two node attributes–degree and minimum distance to a Tier-1 node. We train a model for a BGP graph and validate the model using ground truth AS relationships and CAIDA's inferred AS relationship dataset. Our results show that the model achieves over 92% accuracy on a number of BGP graphs.

Keywords

Internet, AS graphs, edge type prediction, machine learning

1. INTRODUCTION

The Internet connects tens of thousands of autonomous systems (ASes) operated by different administrative entities. Routing between ASes is determined by the Border Gateway Protocol (BGP). BGP is a policy-driven interdomain routing protocol that allows each AS to choose its own policy in determining which routes to import from and export to its neighbors. The routing policies of ASes are largely determined by the business relationships between neighboring ASes, which can be broadly classified into two types: provider-to-customer (p2c) and peer-to-peer (p2p). In the p2c relationship, the customer pays the provider for transiting traffic to the rest of the Internet. In the p2p relationship, two ASes exchange traffic between themselves and their customers on a settlement-free basis.

The Internet AS-level topology can be represented by an AS graph in which nodes represent ASes and two nodes are connected by an edge if they engage in a business relationship to exchange traffic. Researchers have derived AS graphs from three commonly-used data sources [20]: BGP data [9, 7], traceroute measurements [1, 3], and the Internet Routing Registry (IRR) [5]. These data sources result in three types of AS graphs: BGP graphs, traceroute graphs, and IRR graphs. Each type of AS graphs contains unique edges that do not exist in other types of graphs. Thus, the three types of AS graphs complement each other to provide a more complete view of the Internet AS topology.

Given an AS graph, it is important to determine the type of each edge (p2c or p2p) in the graph. Without knowing the edge types, we cannot fully characterize the Internet AS-level topology because connectivity does not imply reachability. Specifically, the AS paths selected by an AS depend on its routing policies, which in turn depend on its business relationships with neighbor ASes. Since network operators are usually not willing to publicly disclose their AS relationship information, researchers have developed various heuristic algorithms to infer AS relationships using BGP data [16, 25, 23, 11, 21, 12, 18]. These algorithms infer AS relationships by analyzing a set of AS paths extracted from BGP routing tables. Thus, they can be used to infer edge types in a BGP graph by analyzing the set of AS paths used to derive the graph. However, these algorithms cannot be used to infer edge types in traceroute graphs or IRR graphs. Thus, a new method is needed to infer edge types in AS graphs derived from any data source.

In this paper we present a novel method of inferring edge types in AS graphs. Our method uses the AdaBoost machine learning algorithm to train a model for a given AS graph that predicts edge types (p2c or p2p) in the graph using two node attributes–degree and minimum distance to a Tier-1 node. We train a model for a BGP graph and validate the model using a set of ground truth AS relationships and CAIDA's AS relationship inference dataset [2]. The results show that our model achieves over 92% accuracy on a number of

BGP graphs.

Machine learning techniques have been used to solve various problems such as detecting anomalous routing dynamics in BGP updates [26], classifying ASes in the Internet [13], identifying network applications based on features of IP traffic flows [24], and winnowing the prefixes with unintended limited visibility from those with intended limited visibility [19]. To the best of our knowledge, this work is the first to use machine learning techniques to infer edge types in AS graphs.

The rest of the paper is organized as follows. In Section 2 we describe how to build a model for an input BGP graph to illustrate our method of building machine learning models for edge type prediction in AS graphs. In Section 3 we provide a detailed analysis of the prediction results of our model. In Section 4 we study the consistency of our model by applying it to ten different BGP graphs. We conclude our work and discuss future directions in Section 5.

2. A METHOD FOR BUILDING EDGE TYPE PREDICTION MODELS

In this section, we present our method of building models for edge type prediction in AS graphs. We use an input BGP graph G to illustrate our method. Given G, we describe how to build a model that predicts the type of each edge (p2c or p2p) in G.

2.1 Creating the Input BGP Graph

We create a BGP graph G using the BGP data archived by Route Views [9] and RIPE RIS [7]. We download a BGP table snapshot from all Route Views and RIPE collectors on each day for the first five days of November 2014. We create G using this BGP dataset as follows. We first extract AS paths from the BGP table snapshots. We then sanitize the AS paths by discarding the AS paths that contain a loop or an invalid ASN[1]. We also remove any AS set and duplicate ASNs arising from AS prepending from the AS paths. Finally, we process each sanitized AS path p to create G as follows. If AS X appears in p, then X becomes a node in G. If AS X and AS Y are adjacent in p, then an edge is added between X and Y in G. The resulting AS graph G contains 48,867 nodes and 183,606 edges.

2.2 Preparing the Training Data

We predict edge types using two node attributes computed from G–*degree* and *distance*. The *degree* of a node is the number of its adjacent nodes in G. The *distance* of a node is its minimum distance to a Tier-1 node, which is calculated as follows. All Tier-1 nodes have distance 0[2]. To find the distance of a non-Tier-1 node v, we compute the shortest path from v to any Tier-1 node. The number of hops in this shortest path is the distance of v.

In order to create a set of training examples, we extract a set of ground truth AS relationships using the same BGP dataset we used to construct G. ASes sometimes add BGP communities to a route to tag the type of neighbor from which the route is received. For example, the community 2914:420, defined by AS2914, indicates that the route was received from a peer. We use the list of BGP communities for tagging type of neighbor assembled by CAIDA [17] to extract a set T of 42,486 ground truth AS relationships from our BGP dataset. T contains 15,070 (35.47%) p2p relationships and 27,416 (64.53%) p2c relationships[3]. Every entry in T is of the form $< (i, j), r_{ij} >$ where (i, j) is a pair of ASes and $r_{ij} \in$ {p2c, p2p} is the AS relationship of i and j. Since we extract the ground truth and construct G from the same BGP dataset, (i, j) is an edge in G. Thus, T gives the ground truth AS relationships for a subset of the edges in G. Specifically, T covers 23.14% of the edges in G.

Given T, we create a set of training examples, denoted by TE, as follows. For every entry $< (i, j), r_{ij} >$ in T, we create an entry in TE with the form $<[degree_i, distance_i, degree_j, distance_j], r_{ij}>$ where $[degree_i, distance_i, degree_j, distance_j]$ is the feature vector that lists the degree of node i, the distance of node i, the degree of node j, and the distance of node j.

2.3 Creating the Model

Given G and the set of training examples TE, we use the Gentle AdaBoost machine learning algorithm [15] to build a model that predicts edge types in G. Boosting is an approach to machine learning in which weak learners are combined to form a strong learner. The AdaBoost algorithm of Freund and Schapire [14] was the first practical boosting algorithm and remains one of the most widely used and studied [22]. Gentle AdaBoost is a variant of AdaBoost that is more resistant to outliers.

2.3.1 The Gentle AdaBoost Algorithm

The Gentle AdaBoost algorithm [15] is shown in Figure 1. The algorithm takes as input a set of training examples $(x_1, y_1), \cdots, (x_m, y_m)$ where each x_i belongs to some domain $\mathcal{X}$ and each label y_i belongs to the set $\{-1, +1\}$. In our problem, each x_i is a feature vector that contains the degree and distance of two ASes for which the ground truth AS relationship is known, and the labels -1 and +1 correspond to p2c and p2p, respectively.

[1] An ASN is invalid if it is unallocated or reserved.

[2] We manually identify 16 Tier-1 ASes according to Wikipedia [8].

[3] We do not include in T hybrid AS relationships where two ASes have different AS relationships at diffferent interconnection locations.

Input: (x_1, y_1),...,(x_m, y_m) where $x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

1. Initialize $w_i = 1/m$ for $i = 1, ..., m$
   $F(x) = 0$
2. For $t = 1, ..., T$
   (a) Fit the regression function $f_t(x)$ by weighted least-squares of y_i to x_i with weights w_i
   (b) Update $F(x) \leftarrow F(x) + f_t(x)$
   (c) Update $w_i \leftarrow w_i exp(-y_i f_t(x_i))$ and renormalize
3. Output the classifier $sign[F(x)]$

Figure 1: Gentle AdaBoost Algorithm

The algorithm maintains a distribution of weights over the training set. Initially, the weight w_i of each training example is set to $1/m$ where m is the number of training examples, and the final classifier $F(x)$ is set to 0. The algorithm is performed in T iterations. In each iteration, a weak classifier $f_t(x) : \mathcal{X} \mapsto \mathbb{R}$ is computed, where the sign of $f_t(x)$ gives the classification. Here, $f_t(x)$ is updated using Newton stepping. Next, the final classifier $F(x)$ is updated with $f_t(x)$, and the weights to be applied in the next iteration on the training examples are updated as $w_i \leftarrow w_i exp(-y_i f_t(x_i))$. In this way, the weak classifiers are combined to form the final classifier $F(x)$ such that $F(x) = \sum_{i=1}^{T} f_t(x)$ and $sign[F(x)] \in \{+1, -1\}$.

We use an R package named ada [10] that implements the Gentle AdaBoost algorithm. The base classifier used by *ada* is a classification tree. So *ada* uses another package called *rpart* [6] that implements recursive partitioning of classification trees.

2.3.2 Training the Model

We split TE into a training set and a test set where the training set is used to train a model and the test set is used to validate the model. To study the effect of training set size on the accuracy of the model, we create 9 different training sets, each containing p% of the entries in TE where $p \in \{10, 20, \cdots, 90\}$. For every training set we create a test set, denoted by $Test_1$, that contains the edges in T that are not contained in the training set. We can use $Test_1$ to validate the model as we know the ground truth AS relationships for the edges in $Test_1$.

Since T only covers 23.14% of the edges in G, we need another test set to validate the prediction results for the edges in G that are not covered by T. Since we do not know the ground truth AS relationships for these edges, we need an alternative way to validate the prediction results. CAIDA has developed a highly accurate AS relationship inference algorithm [18] that infers AS relationships from a set of AS paths. They publish inferred AS relationship data for the first day of every month since January 1998 at [2]. We have compared CAIDA's inferred AS relationship data for 11/01/2014 with T and found that they have a 98.88% match. Thus we use CAIDA's AS relationship inference dataset to validate our prediction results for the edges in $G - T$. Let C denote the set of edges that exist in both G and CAIDA's 11/01/2014 AS relationship inference data. We create a test set $Test_2 = C - T$, which contains the edges in G that cannot be validated by T but can be validated by CAIDA's inferred AS relationship dataset. $Test_2$ contains 126,300 edges, which covers 68.79% of the edges in G. Note that the models trained with different training sets share the same $Test_2$.

We train 9 models using 9 different training sets. For each model we compute the accuracy of the model on $Test_1$ and $Test_2$ where accuracy is defined as the percentage of correctly classified instances over the total number of instances in the test set. Figure 2 plots the accuracy on the two test sets for the 9 models. We see that the accuracy on $Test_1$ is between 93.41% and 94.42%, and the accuracy on $Test_2$ is between 91.50% and 92.31%. The results show that the models can predict edge types in G with high accuracy.

Among the 9 models, the one trained with 70% of TE has the best overall accuracy on $Test_1$ and $Test_2$. The accuracy of this model on the training set is 94.10% and the accuracy on $Test_1$ is 93.95%. The fact that the test accuacy is very close to the training accuracy means that there is little, if any, overfitting. In the next section, we present a detailed analysis of the prediction results of this model.

3. ANALYSIS OF THE PREDICTION RESULTS

In this section, we study the prediction results of the model trained with 70% of TE. The training set, denoted by $Train$, contains 29,740 instances. $Test_1$ contains 12,746 instances and $Test_2$ contains 126,300 instances.

We use three metrics–*accuracy*, *precision*, and *recall*– to evaluate the model. *Accuracy* is defined as the percentage of correctly classified instances over the total number of instances. *Precision* is defined as the percentage of class members classified correctly over the total number of instances classified as class members. *Recall* is defined as the percentage of class members classified correctly over the total number of class mem-

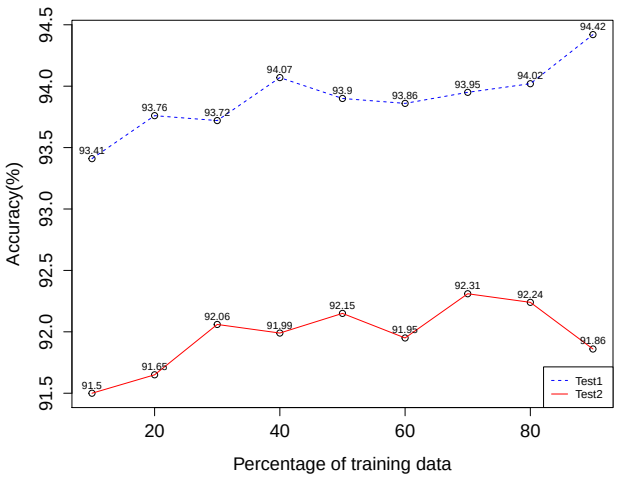

Figure 2: Accuracy vs. training set size

bers.

3.1 Results for $Test_1$

The performance metrics of the model on $Test_1$ are shown in the first column of Table 1. We see that $Test_1$ covers 6.94% of the edges in G and the model has an accuracy of 93.95% on $Test_1$. The p2c class has high precision (96.32%) and recall (94.28%). For the p2p class, recall is high (93.34%) but precision is below 90%.

	$Test_1$	$Test_2$	$Test_{21}$	$Test_{22}$
Size	12746	126300	125647	653
Coverage in G	6.94%	68.79%	68.43%	0.36%
Accuracy	93.95%	92.31%	92.40%	74.58%
Precision-P2C	96.32%	91.87%	92.04%	74.58%
Precision-P2P	89.83%	92.80%	92.80%	NA
Recall-P2C	94.28%	93.45%	93.38%	100%
Recall-P2P	93.34%	91.10%	91.35%	0%

Table 1: Performance metrics of the model on various test sets.

	00	01	11	12	22	23
$Test_1$ Accuracy	100%	97.94%	91.12%	88.71%	93.96%	99%
#Edges in $Test_1$	27	5862	4145	2347	265	100
%P2P edges in $Test_1$	100.00%	2.37%	83.67%	28.33%	65.66%	1.00%
#Edges in $Train$	75	13372	9776	5561	728	228
%P2P edges in $Train$	98.67%	2.29%	83.81%	28.14%	62.36%	1.75%

Table 2: $Test_1$ accuracy and characteristics of $Test_1$ and $Train$ for different distance patterns.

We next examine the accuracy for the edges in $Test_1$ with different distance patterns. We say an edge in G has distance pattern ij if one of the endnodes has distance i and the other endnode has distance j. Fig. 3 shows the percentage of edges with different distance patterns for T, $Test_1$, and $Train$. We see that T,

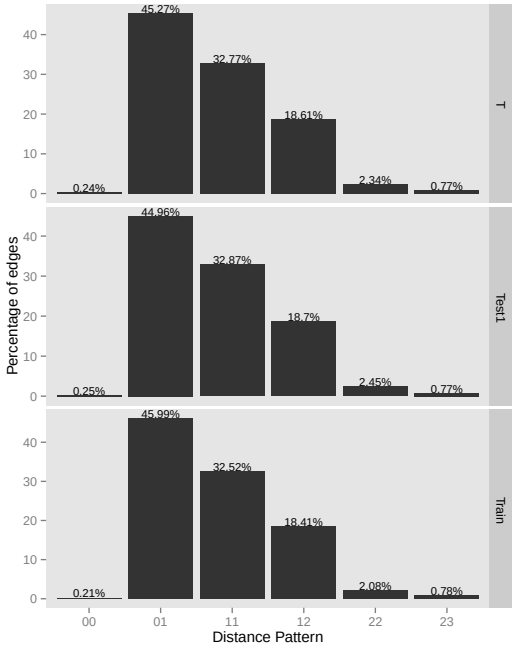

Figure 3: Distribution of distance patterns in T, $Test_1$, and $Train$.

$Test_1$, and $Train$ have the same set of distance patterns $\{00, 01, 11, 12, 22, 23\}$ and they have similar distribution of the distance patterns. This is expected because $Test_1$ and $Train$ are subsets of T. Most edges fall under distance patterns 01 and 11, indicating that most of T are extracted from BGP communities tagged by Tier-1 and Tier-2 ASes.

Table 2 shows $Test_1$ accuracy for each distance pattern and the characteristics of $Test_1$ and $Train$ in terms of the total number of edges and the percentage of p2p edges in each distance pattern. It can be seen that $Test_1$ and $Train$ have similar percentage of p2p edges in each distance pattern. This explains why the model trained with $Train$ has high accuracy on $Test_1$. For distance patterns 00, 11, and 22, $Test_1$ and $Train$ have high percentage of p2p edges while the percentage of p2p edges is low for distance patterns 01, 12, and 23. This means that most of the p2p relationships are established between nodes in the same tier of the routing hierarchy and most of the p2c relationships are established between nodes in different tiers of the routing hierarchy. We see that distance patterns 00, 01, and 23 have very high accuracy (between 97.94% and 100%). This is because for these distance patterns almost all edges in $Train$ and $Test_1$ are of the same type (i.e., the percentage of p2p edges is either very low or very high).

3.2 Results for $Test_2$

The performance metrics of the model on $Test_2$ are shown in the second column of Table 1. We see that

$Test_2$ covers 68.79% of the edges in G and the model has an accuracy of 92.31% on $Test_2$. Both p2p and p2c classes have similar precision and recall, which are between 91.10% and 93.45%. $Test_1$ and $Test_2$ together cover 75.73% of the edges in G. Thus, we have validated 75.73% of the prediction results using either the ground truth or CAIDA's inferred AS relationship dataset.

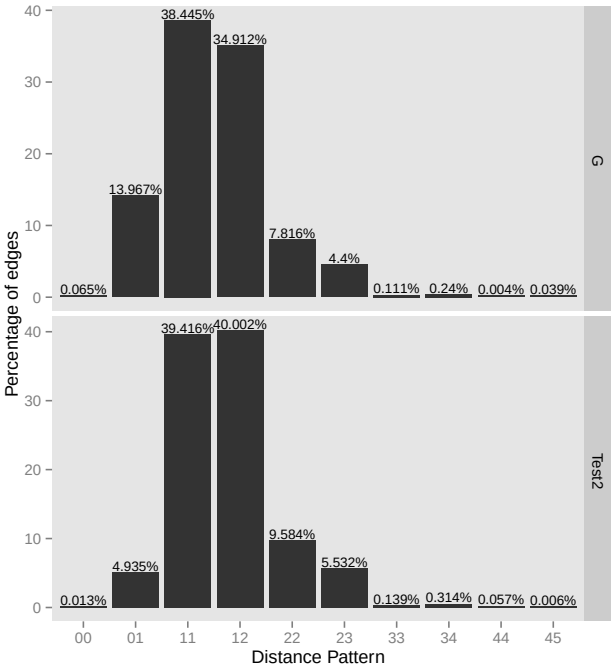

Figure 4: Distribution of distance patterns in G and $Test_2$

We plot in Fig. 4 the percentage of edges with different distance pattens for G and $Test_2$. By comparing Fig. 3 and Fig. 4 we see that G and $Test_2$ contain distance patterns 33, 34, 44, and 45 but T, $Train$, and $Test_1$ do not contain these distance patterns. Also, the top two most represented distance patterns in $G/Test_2$ are 11 and 12 while the top two most represented distance patterns in $T/Train/Test_1$ are 01 and 11. This means that T is biased towards edges in higher tiers of the routing hierarchy in comparison to G. This is due to the fact that T is extracted from BGP communities, which mostly come from Tier-1 and Tier-2 ASes. The fact that the distance pattern distribution of $Train$ matches that of $Test_1$ but does not match that of $Test_2$ explains why the accuracy on $Test_1$ (93.95%) is higher than the accuracy on $Test_2$ (92.31%).

To study how the absence of distance patterns 33, 34, 44, and 45 in $Train$ affects the prediction accuracy for these distance patterns, we split $Test_2$ into two subsets: $Test_{21}$ and $Test_{22}$. $Test_{21}$ contains the edges with distance patterns 00, 01, 11, 12, 22, 23 and $Test_{22}$ contains the edges with distance patterns 33, 34, 44, 45. The performance metrics of the model on $Test_{21}$ and $Test_{22}$ are shown in the third and the fourth columns of Table 1. We see that the accuracy on $Test_{21}$ is 92.40% while the accuracy on $Test_{22}$ is only 74.58%. This shows that our model can make accurate predictions for the edges whose distance patterns are represented in $Train$ but have low prediction accuracy for the edges whose distance patterns are not represented in $Train$. We note that while the accuracy on $Test_{22}$ is significantly lower than the accuracy on $Test_{21}$, the overall accuracy on $Test_2$ (92.31%) is only slightly lower than the accuracy on $Test_{21}$ (92.40%) because $Test_{22}$ contains only 0.52% of the edges in $Test_2$. We also note that our model predicts all the edges in $Test_{22}$ as p2c. As a result, Precision-P2P is NA, Recall-P2P is 0%, and Recall-P2C is 100% in Table 1.

Table 3 shows $Test_2$ accuracy and the characteristics of $Test_2$ and $Train$ for each distance pattern in G. We see that the accuracy for distance patterns 33, 34, 44, and 45 are low as they are are not represented in $Train$. Since all edges in distance patterns 33, 34, 44, and 45 are predicted to be p2c, the accuracy for each of these distance patterns is equal to the percentage of p2c edges in that distance pattern. For example, the accuracy for distance pattern 33 is 81.82% because 81.82% of the edges with distance pattern 33 in $Test_2$ are p2c. All distance patterns that are represented in $Train$ have high accuracy (between 92.07% and 98.67%) except for 01, which has an accuracy of 85.61%. This is because while 15.66% of the 01 edges in $Test_2$ are p2p, only 2.29% of the 01 edges in $Train$ are p2p. Thus, our model has not seen enough p2p edges with distance pattern 01 in $Train$ to make accurate predictions.

We observe in Table 3 that for $Test_2$ the percentage of p2p edges in distance pattern ii decreases as i increases from 0 to 3. For example, 94.21% of the edges in distance pattern 00 are p2p, and only 18.18% of the edges in distance pattern 33 are p2p. This means that higher tier ASes are more likely to peer with another AS in the same tier.

4. VALIDATING CONSISTENCY OF THE MODEL

We apply our model to other BGP graphs to evaluate the consistency of our model. We use the Internet AS-level topology data archived by the Internet Research Lab (IRL) at UCLA. IRL publishes daily snapshots of AS topologies [4] derived from BGP data collected by Route Views, RIPE RIS, PCH, and Internet2. We create 10 AS graphs using the IRL AS topology data, one for each month between April 2014 and February 2015 except May 2014[4]. The AS graph for a specific month

[4]We do not create an AS graph for May 2014 because the AS relationship inference data for May 2014 is not available at CAIDA's website. As a result, we cannot validate our

	00	01	11	12	22	23	33	34	44	45
Accuracy	94.12%	85.61%	92.07%	92.18%	94.55%	98.67%	81.82%	78.84%	50%	36.11%
#Edges in $Test_2$	17	6233	49782	50523	12105	6987	176	397	8	72
%P2P edges in $Test_2$	94.12%	15.66%	77.31%	32.01%	40.78%	1.30%	18.18%	21.16%	50.00%	63.89%
#Edges in $Train$	75	13372	9776	5561	728	228	0	0	0	0
%P2P edges in $Train$	98.67%	2.29%	83.81%	28.14%	62.36%	1.75%	NA	NA	NA	NA

Table 3: $Test_2$ accuracy and characteristics of $Test_2$ and $Train$ for different distance patterns.

	04/14	06/14	07/14	08/14	09/14	10/14	11/14	12/14	01/15	02/15
#Edges in graph	195068	197271	198160	204703	204704	210205	207781	210195	212543	204959
$\|Test_1\|$	11386	11730	11940	12106	12284	12582	12586	12431	12215	12078
$\|Test_2\|$	123225	122423	123330	124450	126228	125237	127048	129581	130097	140321
Coverage by $Test_1 \cup Test_2$	69.01%	68.00%	68.26%	66.71%	67.66%	65.56%	67.20%	67.56%	66.96%	74.36%
$Test_1$ Accuracy	92.85%	92.54%	92.40%	92.52%	92.49%	92.54%	92.53%	92.45%	92.66%	92.42%
$Test_1$ Precision-P2C	95.85%	95.51%	95.46%	95.76%	95.46%	95.83%	95.72%	95.62%	96.09%	95.87%
$Test_1$ Precision-P2P	87.86%	87.53%	87.25%	87.05%	87.37%	87.03%	87.14%	87.07%	86.83%	86.62%
$Test_1$ Recall-P2C	92.92%	92.83%	92.66%	92.58%	92.86%	92.51%	92.63%	92.64%	92.55%	92.35%
$Test_1$ Recall-P2P	92.88%	92.02%	91.94%	92.42%	91.78%	92.60%	92.35%	92.11%	92.60%	92.57%
$Test_2$ Accuracy	92.07%	92.30%	92.06%	90.86%	92.25%	92.33%	92.40%	92.22%	92.24%	91.14%
$Test_2$ Precision-P2C	92.65%	93.00%	92.49%	90.42%	92.78%	93.02%	92.83%	92.81%	92.92%	90.12%
$Test_2$ Precision-P2P	91.39%	91.50%	91.56%	91.36%	91.62%	91.57%	91.94%	91.61%	91.54%	92.21%
$Test_2$ Recall-P2C	92.73%	92.86%	92.73%	92.18%	92.41%	92.40%	92.39%	92.00%	91.90%	92.37%
$Test_2$ Recall-P2P	91.30%	91.62%	91.28%	89.44%	92.02%	92.25%	92.40%	92.46%	92.60%	89.92%

Table 4: Performance metrics of the model on various IRL graphs

is created by merging the daily AS topology data published by IRL. We note that the AS graph for February 2015 is created using the AS topology data for the first 14 days of the month as data for the rest of the month are not available at IRL's website. The IRL graphs are more complete than our graph G because we do not use BGP data from PCH and Internet2 to create G.

We create two test sets for every IRL graph. The first test set, denoted by $Test_1$, contains the edges that are in both the IRL graph and T but are not in $Train$. The prediction results for $Test_1$ can be validated using T. The second test set, denoted by $Test_2$, contains the edges that are in both the IRL graph and CAIDA's AS relationship inference data for the corresponding month but are not in T. The prediction results for $Test_2$ can be validated using CAIDA's AS relationship inference dataset. Table 4 shows the performance metrics of our model on the two test sets for the 10 IRL graphs. The table also shows the size of each IRL graph and the size of the two test sets for each IRL graph. We see that the accuracy on $Test_1$ is between 92.40% and 92.85%. The accuracy on $Test_2$ is between 90.86% and 92.40%, and 8 out of the 10 graphs have over 92% accuracy. The results show that while our model is trained using degree and distance attributes computed from our November 2014 BGP graph G, the model achieves consistently high accuracy on other BGP graphs.

5. CONCLUSION AND FUTURE WORK

We present a new machine learning approach to edge type prediction in AS graphs. Given an AS graph, our method uses the AdaBoost algorithm to train a model that predicts the edge types (p2p or p2c) in the graph using two node attributes computed from the graph– degree and minimum distance to a Tier-1 node. We created a BGP graph using November 2014 BGP data from Route Views and RIPE, and trained a model for the graph using ground truth AS relationships extracted from BGP communities. The model achieves 93.95% and 92.31% accuracy on $Test_1$ (validated by ground truth) and $Test_2$ (valided by CAIDA's AS relationship inference data), respectively. Furthermore, our model achieves similar acuracy on ten BGP graphs derived from IRL's AS topology data.

While we only consider BGP graphs in this work, our method can be used to build models for AS graphs derived from other data sources. In our future work we will build models for traceroute graphs and IRR graphs. We will also use other AS attributes (e.g., AS type) to improve the prediction accuracy.

prediction results using CAIDA's data.

Acknowledgments

We would like to thank Kris De Brabanter for his helpful comments and discussions.

6. REFERENCES

[1] *Archipelago Measurement Infrastructure.* http://www.caida.org/projects/ark.

[2] *CAIDA's AS Relationship Inference Dataset.* http://data.caida.org/datasets/as-relationships/serial-1/.

[3] *The DIMES Project.* http://www.netdimes.org/new/.

[4] *Internet AS-level Topology Archive*. http://irl.cs.ucla.edu/topology/.
[5] *The Internet Routing Registry*. http://www.irr.net.
[6] *Package rpart*. http://cran.r-project.org/web/packages/rpart/rpart.pdf.
[7] *RIPE Routing Information Service*. http://www.ripe.net/data-tools/stats/ris/.
[8] *Tier-1 network*. http://en.wikipedia.org/wiki/Tier_1_network.
[9] *University of Oregon Route Views Project*. http://www.routeviews.org.
[10] M. Culp, K. Johnson, and G. Michailides. ada: An R package for stochastic boosting. *Journal of Statistical Software*, 17(2):1–27, 9 2006.
[11] G. Di Battista, M. Patrignani, and M. Pizzonia. Computing the types of the relationships between autonomous systems. In *IEEE INFOCOM*, volume 1, pages 156–165, 2003.
[12] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, k. claffy, and G. Riley. AS relationships: inference and validation. *ACM SIGCOMM Computer Communication Review*, 37(1):29–40, 2007.
[13] X. Dimitropoulos, D. Krioukov, G. Riley, and K. Claffy. Revealing the autonomous system taxonomy: The machine learning approach. *Passive and Active Measurements Workshop (PAM)*, 2006.
[14] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
[15] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.
[16] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, 2001.
[17] V. Giotsas, M. Luckie, B. Huffaker, et al. Inferring complex AS relationships. In *The Internet Measurement Conference*, pages 23–30, 2014.
[18] M. Luckie, B. Huffaker, A. Dhamdhere, V. Giotsas, and kc claffy. AS relationships, customer cones, and validation. In *The Internet Measurement Conference*, pages 243–256, 2013.
[19] A. Lutu, M. Bagnulo, J. Cid-Sueiro, and O. Maennel. Separating wheat from chaff: Winnowing unintended prefixes using machine learning. In *IEEE INFOCOM*, pages 943–951, 2014.
[20] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, A. Vahdat, et al. The Internet AS-level topology: three data sources and one definitive metric. *ACM SIGCOMM Computer Communication Review*, 36(1):17–26, 2006.
[21] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang. The (in)completeness of the observed Internet AS-level structure. *IEEE/ACM Transactions on Networking*, 18(1):109–122, 2010.
[22] R. E. Schapire. Explaining AdaBoost. In *Empirical Inference*, pages 37–52. Springer, 2013.
[23] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *IEEE INFOCOM*, volume 2, pages 618–627, 2002.
[24] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. *ACM SIGCOMM Computer Communication Review*, 36(5):5–16, 2006.
[25] J. Xia and L. Gao. On the evaluation of AS relationship inferences. In *IEEE Global Telecommunications Conference*, volume 3, pages 1373–1377, 2004.
[26] K. Zhang, A. Yen, X. Zhao, D. Massey, S. F. Wu, and L. Zhang. On detection of anomalous routing dynamics in BGP. In *NETWORKING 2004*, pages 259–270. Springer, 2004.