2019 International Conference on Computing, Networking and Communications (ICNC): Machine Learning for Communication and Networking

Using Natural Language Constructs and Concepts to Aid Network Management

Abhishek Dwaraki*, Richard Freedman†, Shlomo Zilberstein†, Tilman Wolf*
*Department of Electrical and Computer Engineering
{adwaraki,wolf}@umass.edu
†College of Information and Computer Sciences
{freedman,shlomo}@cs.umass.edu

***Abstract*—The increasing complexity of networks together with technological trends that allow for fine-grained control and programmability have made network management a pressing challenge. In this work, we propose to harness the vast amounts of network management data that are available from different sources in an automated system that can infer context and semantics. We present an argument for a Network Processing Language that is based on the ideas of natural language processing. Our approach shows how concepts, such as collocations, can be applied to network management data. We demonstrate the effectiveness of our approach to detect route prefix and sub-prefix hijacks. This work presents one step toward effectively using automated tools for network management in complex, programmable networks.**

I. Introduction

Networked systems today operate at an unprecedented scale. Data communication networks provide core functionality for today's distributed services and applications that span personal, business, and government use. Various developments in technology, networking or otherwise, continue to put a massive strain on these networked devices in our daily lives. Virtualization is re-defining hyper-scale on-demand computing by pushing the boundaries of how much can be achieved on a single hardware device. Containerization is revolutionizing operational time-scales in DevOps and traditional design-develop-test cycles. The Internet of Things promises to connect every device to the Internet. Each of these technologies place demands of varying nature on communication networks, such as real-time responsiveness, rapidly changing traffic workloads or security concerns [1].

The advent of programmable networks that provide fine-grained control of network traffic have given rise to new challenges in network management. It is now critical to maintain correct operation of these networks to avoid adverse business outcomes. Unfortunately, existing network management methodologies have not evolved at the same pace as these networking technologies and architectures. Subsequently, current network management practices do not provide adequate solutions for highly dynamic, programmable environments. This is summed up perfectly by:

> *The widespread integration of these applications into our daily lives raises the bar for network management, as users elevate their expectations for real-time interaction, high availability, resilience to attack, ubiquitous access, and scale. [1]*

Our research attempts to explores a key question in network management – can the power of machine learning be efficiently harnessed such that network administrators can make better decisions in highly dynamic environments? Specifically, we propose an approach to network management that is conceptually based on statistical Natural Language Processing (NLP).

Today, data is available in abundance. Such data includes log files, routing tables, BGP tables, IP tables etc. With machine learning making great advances, we believe this data can be put to good use. Computational linguistics tries to characterize and explain the expressions appearing in the language, taking into account semantics and context [2]. What if network management could be improved with the help of a framework developed specifically to understand network vocabulary, terminology, semantics and context? Such an approach that can be trained to analyze the semantics and context behind network paradigms, etc. We believe that network management can definitely benefit from a Network Processing Language framework.

The contributions of this paper are as follows:

1) We state the case for developing a network processing language,
2) We demonstrate how collocations can be used to process network management data, and
3) We evaluate a BGP dataset with these approaches and show that it can be used to successfully detect route prefix and sub-prefix hijacks.

The primary motive of this paper is to present the need for a framework that can contextually and semantically construct reasoning when provided with network state and information. We utilize BGP routing data only to illustrate the feasibility of using natural language constructs as a source of inspiration to analyze network data.

Section III discusses the need to start embedding intelligence in the network itself, with the aim of improving network management. Subsequent sections III-A and III-B deal with the potential advantages of developing a network processing framework and constructs from statistical NLP. Section IV characterizes the dataset and tools used. An illustration of

978-1-5386-9223-3/19/$31.00 ©2019 IEEE

detecting route prefix hijacks on BGP data is discussed in Section V and preliminary results are presented in Section VI. Section VII summarizes our contributions.

II. Related Work

Traditionally, machine learning has been used in some form or the other to aid network analysis. Pytheas [3] explores a data-driven method to optimize quality of experience (QoE) using machine-learning techniques. Li et al. provide a novel machine learning based approach for efficient traffic classification [4]. A really old approach by Sasisekharan et al. [5] demonstrates that such machine-aided and data driven approaches are not new to networking or network management. They have been around for a long period of time, but now are at the forefront of research and development due to their capabilities and applicabilities to problems being seen in present-day networks, especially the newer architectures.

A holistic view of network management will tell us many problems that programmable architectures face are not even new. They are newer manifestations of solved problems from legacy networks that are re-appearing due to the scale and levels of dynamism present today.

Some of the conceptual ideas for our research are tenets of linguistic science, which are explored and explained in lucid detail by Manning and Schutze in their seminal book "Foundations of Statistical Natural Language Processing" [2]. This research borrows just some fundamental concepts to demonstrate that these can be generalized and extended to networking too. We do this by applying these concepts on public BGP routing data from the RIPE-RIS database.

The Border Gateway Protocol is a research world unto itself, to the extent that the subject matter is voluminous. Much of our understanding of ASes, their relationships, BGP anomalies and misconfiguration has been drawn from [6]–[11]. Machine learning has previously been used to analyze and understand BGP data. Li et al. [12] use machine learning to extract features from RouteViews [13] data and present an Internet Routing Forensic Framework to reason about BGP anomalies. [14] talks about a 2-stage machine-learning model for BGP anomaly detection and [15] approach the same problem by considering prefix visibility.

A comprehensive summarization of a wide-range of BGP anomaly detection techniques is presented in [16]. We use ARTEMIS [17] as our primary source of information for state-of-the-art and to draw parallels with our methods of detecting BGP route anomalies.

III. Augmenting Network Management with Intelligence

Over the years, advances in network management have lagged behind those in protocols, application, architectures and designs. But as we keep falling behind the curve, it is only going to get harder to manage the extremely dynamic networking devices and architectures in use today. For network management to improve, the network needs to change [1]. It is essential to infuse the network with a certain degree of intelligence that will allow it to assist us in making decisions. We do not envision this approach in the context of entirely replacing the human operator, but bolstering the quality of information being provided, thus leading to enhanced decision making capabilities.

A. The Need for Network Language Processing

The advantages of network programmability are well documented [18]–[20] with programmatic networks being extensively researched in both the industry and academia [21]. Google's B4 WAN [22] is a prime example of a wide-area programmable deployment. The presence of various open source programmable frameworks such as OpenContrail [23], OpenStack, OP-NFV etc. bear evidence of industrial interest in the area.

In recent times, there has been a surge of interest in developing languages for networks. Most of the languages have been dictated by domain and application requirements rather than general purpose network programmability. One of the more notables ones advocating for platform and protocol independence on the data-plane has been P4 [24]. There have been other ones such as NetKAT that are based on mathematical foundations and equational theory [25], rather than ad-hoc programmability, Frenetic [26] that facilitates building SDN applications in a facile manner. But most of these languages focus on packet manipulation, data-plane operations, telemetry and application programming. Their focal point has been to abstract the network away from the data-plane, thus raising the level of abstraction away from hardware, and making the network easier to program.

If network programmability can be applied to almost all the aspects and component layers of the architecture, then why not for network management? It is our belief that developing a network processing language framework for managing these new, hyper-scale architectures merits thorough investigation.

B. Statistical Sequence-Based Analysis

Drawing parallels from [2], we can define two basic questions, which if answered, should be able to broadly capture the essence of a language. (i) what are the kinds of things networking devices are "saying"?, and (ii) what do these things tell us about the "world" (in our case, the network)?

There is a clear distinction here with the scope of these two questions. The first question deals primarily with structural aspects of the language. The latter addresses semantics and a contextual understanding of the information, relative to the greater whole. In a networking context, we know what devices are "saying". This is network state, both hard and soft-state [27]. At this point, the state is constitutionally just raw information, without structure, semantics or context. A network language framework will allow this data to have a "linguistic" structure. A structure, such as NetKAT's [25] will help extract knowledge from it. This helps put information in context to the "network world", making it useful and providing us with better insights into network operation, thus improving network management.

Here, we elaborate on a few concepts from NLP that we use in our research. A sequence is a collection of entities (here, words) that occur one after the other. In our BGP example, it might be the *peer-address* and *peer-asn* occurring one after the other. They just appear together, and possess a pseudo-semantic meaning, only telling us where the BGP message is coming from. Alternatively, a "collocation" is by definition a pair or group of words that are habitually juxtaposed. In linguistic parlance, this means "the whole is perceived to have an existence beyond the sum of the parts" [2]. For example, the words in the ASPATH are semantically related to each other, and offer us more information together as a whole. Even if the ASPATH is broken down into individual bigrams, each bigram still retains a semantic significance, because it tells us what edges are more popular than the others (based on pure statistical inference). We utilize collocations in BGP dataset analysis to identify prefix announcements/withdrawals and pathlets that make up the ASPath.

We look for collocations and sequences in the resulting text in both cases of analyzing a corpus of BGP message or a live stream. The subtle difference between a collocation or a sequence comes from the input. A collocation finder works extremely well on large corpora, which would be the offline approach. The parser, while looking to infuse semantic meaning, evaluates collocations not just when they occur, but their occurrences in the corpora as a whole. But then, a live BGP stream does not afford us the luxury of a huge corpora. Consequently, we look for sequences in each BGP message and then condition the sequence statistically, building conditional probability distributions around the sequences.

IV. Tools and Datasets

We build our dataset using publicly available control-plane monitoring resources, such as RIPE's Routing Information System (RIS) [28]. RIS is connected to 21 route collectors in geographically diverse locations, peering with $\approx$ 300 ASes. The route collectors provide both RiB dumps and updates collected from monitors.

We also use CAIDA's BGPStream [29] framework. This enables live streaming of BGP messages from the Route-Views [13] project, CAIDA's own OpenBMP and RIPE's RIS collectors. Some RIS collectors live-stream updates and information in available on other collectors typically within minutes. OpenBMP also supports low-latency streaming for applications that do monitoring at shorter time-scales.

Table I illustrates a BGP update message in human-readable format. The corpus that we use to train the offline model uses a slightly different, one-entry per line format that is lighter to process and quicker to generate.

TABLE I: Sample BGP Message

BGP Field	BGP Field Value
TIME	09/07/17 00:00:04
TYPE	BGP4MP/MESSAGE/Update
FROM	195.66.224.159 AS8966
TO	195.66.225.241 AS12654
ORIGIN	IGP
ASPATH	8966 17557 9260 132165
NEXTHOP	195.66.224.159
ANNOUNCE	111.88.223.0/24 111.88.222.0/24

A. Offline Modeling

The dataset from the RIPE/RIS database for the offline modeling is prepared as follows:

- We use five of the RIS collectors, RRC01, RRC04-07. The collectors dump BGP message at five-minute intervals. Consider this as one time epoch.
- We form the dataset over a period of one month, from September 8th, 2017 to October 8th 2017. We pick five epochs every day, from each collector.
- We use the BGPStream [30] site to look for potential BGP hijacks that have occurred in the one-month time-frame. Subsequently, we collect the message dumps for time-epochs identified with potential prefix hijacks. This forms our test dataset that we run against our trained model.

B. Modeling Based on Live BGP Streams

For the online model, we pick a random time from April 8th 2018 and go back one year. We then consider a 30-minute window starting from this random timestamp and collect announcements from a RIS collector picked at random from RRC01, RRC04, RRC05, RRC06 and RRC07. This process is repeated 360 times, and we end up with 10800 hours worth of BGP updates. The model is trained as the updates are live-streamed using PyBGPStream, a Python bindings package to BGPStream [29].

V. Detection Methodology

In this section, we elaborate on how to detect Type-0, Type-1 and sub-prefix hijacks. Borrowing from [17], the percentage of invisible, higher-order hijacking events tend to pollute smaller sections of the Internet. Subsequently, here we deal with only Type-0, Type-1 and sub-prefix hijacking of both types.

A. Overview

Section III-B elaborates on the constructs used for our analysis. Here, we illustrate how to actually detect hijacks based on a model that is combination of an offline and online approach. We use collocations, or pairs/triplets/quadruplets of words that occur more frequently than expected judged on the frequency of their individual words. Collocations are a whole beyond the sum of the parts. Finding collocations primarily requires the frequencies of words and their occurrence in the context of other words. They, more often than not, need to be filtered to get rid of juxtapositions that do not make sense, thus only retaining useful content terms. We then score each sequence (we use sequence here to establish the notion of a collocation) of words with an association measure, to determine the relative likelihood of each sequence being a collocation. Our approach utilizes a combination because of

some limiting factors of streaming updates. When we stream information, each message is independent of the other and bigrams end up being simple sequences of words. As a result, the semantic understanding provided by their appearance in a corpus is lost. To correct for this phenomenon, we first train the model on a corpus of BGP updates downloaded from the RIPE-RIS [28] database, then we further train the model on live streams to keep it up-to-date.

Collocations, frequencies and their probabilities work well if given a sizeable corpora to work with. With BGP data, this is not a concern since there is an inordinately large amount of data available to work with. Also, any analysis of a natural language involves pre-processing to prepare the corpora. In the case of NLP, it would be normalizing the text, converting everything to lower-case, removing punctuation and stop-words etc. Our pre-processing involves normalizing fields found in the messages etc, tokenization etc. It is the network equivalent of preparing a natural language corpus for analysis.

B. Detecting Type-0/1 Hijacks

Origin-AS or Type-0 hijacks are by far the simplest ones to observe. These are route hijacks in which an AS announces its ownership of a prefix, that is neither its own, nor does it have authorization to originate an announcement. Suppose *AS1 - 192.168.0.0/23* is a valid mapping, if another AS, say *AS2* suddenly advertises *192.168.0.0/23*, that constitutes a Type-0 hijack.

Understanding the data to be analyzed is critical. The key to detecting a Type-0 route hijack is the prefix and its originating AS, that is the last entry of the ASPATH. Thus, we look for collocations of size 2, or bigrams, where either the left value is a prefix accompanied by a string consisting of only numerals, or vice versa. Other bigrams can be discarded. In BGP MRTs, the prefix appears first, followed by the ASPATH. If the order were to be swapped, it would be trivial to just find bigrams over these two entities, which would definitely result in one of the bigrams being an advertisement. Since our data is the positioned in the opposite order, we can do one of two things: (i) swap them around and find bigrams, or (ii) adjust the window size for finding collocations.

The latter approach uses a look-ahead parameter to form bigrams. Since the prefix is followed by the ASPATH, and self-prepending is prevalent, we process the ASPATHs to remove AS-prepending and adjust the window size accordingly to find bigrams. The frequencies of occurrence of the bigram itself, in conjunction with the frequency of each component of the bigram occurring with the other, dictate the level of correlation between the components. This information, analyzed over a sizeable corpus, will most definitely establish a certain confidence level for prefix advertisements that normally occur together.

It would be fair to question the notions of a bigram, as it appears to be a tuple of two components, but subtleties lie in the interpretation. Tuples are just two words put together with no real relationships, but an n-gram (and thus a bigram) establishes a semantic relationship between the two components themselves put in perspective of the entire corpus. Detecting Type-1 route hijacks follows a similar line of reasoning, except that we look for collocations of order 3, rather than 2. Again, in this case, we end up with substantially more collocations and these have to be filtered. For example, our *test corpus* of a 140-odd documents results in $\approx$ 22 million collocations. When we filter these to take into account only announcements, the number drops to $\approx$ 3 million.

TABLE II: RiB Prefix Corpus Sizes

Collector	RiB Prefix Count
RRC00	771388
RRC01	771133
RRC04	750499
RRC05	737207
RRC06	756660

C. Sub-Prefix Hijacks and Potential Misconfiguration

One of the concerns of using NLP approaches for BGP anomaly detection is its ability to detect sub-prefix hijacks. In this section, we illustrate that it is definitely possible to identify sub-prefixes using language processing techniques.

To identify sub-prefixes with natural language methods, we have to consider them as strings of characters and not IP addresses. The relationship between a sub-prefix and its super-prefix (or at many pairs for that matter) is not very different when they are treated as strings and not IP addresses. For example, *64.106.0.0/17* is the super-prefix of *64.106.64.0/18*. The difference between both strings is a single character transformation. It is possible to transmute one string into the other by either deleting one character and adding the other or replacing the character in question. This concept of string similarity is termed as word-distance and is defined by the number of operations it takes to transform a string into another. Possible operations are additions, deletions, replacements and transpositions. There are various fuzzy string matching techniques that use different measures of distance. One of the well-known ones is the "Levenshtein distance" that considers only additions, deletions and substitutions.

We utilize this metric to list similar prefix strings extracted from an exhaustive list of prefixes observed during the training phase. This "prefix-corpus" can also be built using the CAIDA AS-relationship dataset [31] listing all advertised prefixes. It is then trivial to eliminate the candidates from this list that do not share a network space overlap with the sub-prefix. One might be led to question the advantage of this approach, because trie-based subnet lookups are faster and it is just a reverse-trie traversal to obtain all the super-nets for a certain prefix. To explain this, we look to Mahajan et al's study of BGP misconfiguration [7]. The study shows that a lot of short-lived routes are indeed misconfiguration. In this eventuality, it might not exactly be a sub-prefix, but one that appears to be lexically similar. A list of such strings helps us narrow down any similar looking prefixes (if there are no super-prefixes present). We can subsequently estimate the probability of this prefix belonging to any of the ASes that own prefixes on the

candidate list. This is an advantage that trie-based approaches cannot provide.

D. A Note on Type-N, N≥2 Hijacks

We have not dealt with Type-N hijacks as part of this work, mainly due to the difficulty in finding a sufficient number of Type-N hijacks to test the model with (in a simulated scenario). Since the primary focus of this paper is to make a case for developing a network processing language framework and BGP data analysis only serves as the means to that end, we reserve this for future investigation. That being said, it is not very difficult to cast the Type-N hijack problem as a collocation finding problem.

One approach to accomplish this would be to vary the window size for collocation finding until a best-fit is found for accuracy. [32] provides a good starting point to experiment with this approach. The window size can be set to accommodate the average ASPATH length on the Internet and varied around it to observe accuracy variations.

VI. Evaluation Methods and Experimental Results

We now analyze the results of our experiments with respect to the model's accuracy in predicting outcomes. Doing this allows us to evaluate whether the model was able to successfully utilize the semantics of collocation to arrive at the right result. To evaluate the collocation-based classification, receiver operating characteristic (ROC) curves are plotted and the area under the ROCs for the Type-0 and Type-1 hijacks calculated. The ROC curves give us a good estimate of how the classifier is working.

The sub-prefix matching evaluation is carried on the dataset constructed in Section IV-A. A prefix corpus is also constructed from the dataset, that consists of all the unique prefixes contained in it. First, we pick 100 prefixes at random from the prefix corpus. A subnet is generated for each prefix (provided that the subnet itself is not present in the corpus). It is then matched against all prefixes in the corpus to find strings that are most similar to the prefix, using the Levenshtein distance metric. If the super-prefix picked earlier is part of the candidate list, it is counted as a successful sub-prefix identification. The hijack analysis then proceeds as described already for regular Type-0 and Type-1 hijacks with evaluation being performed for the sub-prefix:ASN mapping.

The test dataset contains $\approx$ 2.9 million genuine prefix announcements and 45000 route hijack announcements.

These tests are run for RiB prefixes from five collectors, all of them being approximately the same size (to keep the evaluation consistent). The precision of sub-prefix matching and the execution time are compared to ensure that this is not a bottleneck. Table II shows the number of unique prefixes present in the respective collector's dataset.

A. ASPath Change Detection

We use one experiment from Blazakis et al. [8] to demonstrate that we can successfully utilize the concept of collocations to extract useful information from network data. The authors in [8] describe an ASPATH edit distance metric (AED), based on the concept of Levenshtein distance to track changes to the ASPATH. Using a Levenshtein distance-based metric normally means making code changes to account for the ASPATH being treated as a special case of a string. The same results can be achieved with what we term the bigram edit distance, or BED. A bigram edit distance metric can accomplish the same without the need for any external code changes or modifications to the Levenshtein distance algorithm.

In one of the experiments, the authors try to characterize how the AED behaves on a global scale. For this example, they consider BGP routing data from the week starting December 21, 2004 through December 28, 2004. The authors reason that a BGP route-leak during this period should show up definitely on the AED characterization. Figures 1 and 2 illustrate the cumulative distribution of the BED over the specified time period. The route leak incident is clearly reflected in the BED trace for December 24, 2004, very similar to the results that Blazakis et al. demonstrate in their research.

B. Hijack Detection Results

Figures 3 and 4 are the ROC curves for the Type-0 and Type-1 route hijacks respectively. Since we do not know what is a good threshold to use for classification, we make use of the ROC curve to plot the specificity against sensitivity. Since the diagonal of an ROC curve describes a uniformly random classifier such as a coin flip, classifiers should minimally perform better than this. The extent to which they score higher than this dictates how much better they are performing, since the area under the ROC, or AUROC is greater. In both our cases, the true positive counts are high and false positive counts are minimal, which is why we see the curve being pulled towards the ideal (0, 1) corner of the graph.

The other avenue where the ROC helps us is to choose a good threshold value which minimizes the false positive rate and maximizes the true positives. In the case of Type-0 hijacks, a good threshold appears to be 0.98, which is very close to 1. For the Type-1 ROC curve, it is around 0.85. The false positive rate is $\approx$ 0.18 in both cases. We believe that using a maximum likelihood estimate (MLE, that is based on raw frequencies of occurrence), coupled with the huge difference in classes themselves (2.9 million to 45K), results in some lesser seen announcements to be classified as hijacks. A few ways to fix this would be to use a probability estimate other than MLE, such a Kneser-Ney distribution, that allocates certain margins for unknowns. Increasing the number of collectors that the model is getting information from is also a potential solution, followed by developing a weighted method to score the likelihood of appearance.

C. Sub-prefix Matching Results

Sub-prefix matching was performed as explained in the experimental methodology. Figures 5 and 6 plot performance of the matching algorithms on a single collector's RiB prefix corpus. We evaluate three different metrics, ratio, partial ratio

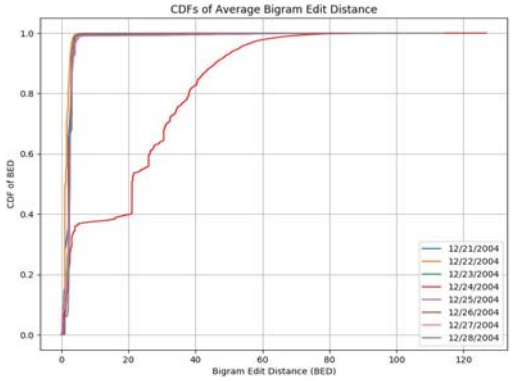

Fig. 1: CDF of Bigram Edit Distance

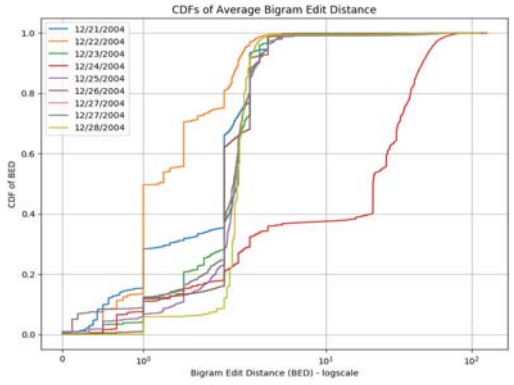

Fig. 2: CDF of Bigram Edit Distance (logscale)

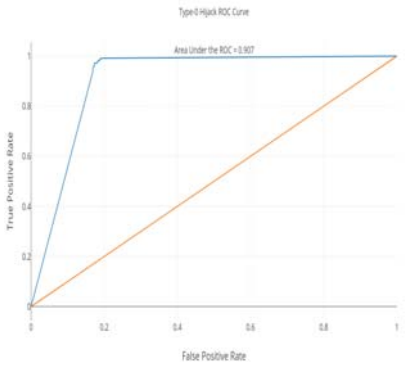

Fig. 3: Type-0 ROC Curve

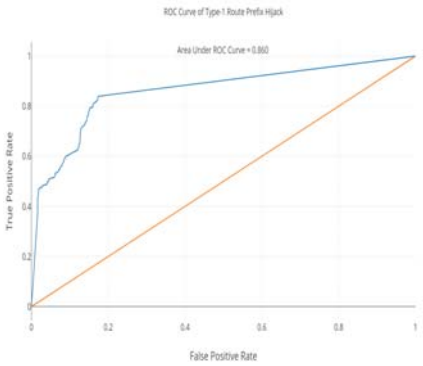

Fig. 4: Type-1 ROC Curve

Fig. 5: Sub-Prefix Matching Precision

and token set ratio. Ratio works purely on the basis of Levenshtein distance-based matching. Partial ratio evaluates potential matches based on best sub-string matches and token set ratio approach tokenizes the strings and compares their intersection with the remainder. It is clear from these two figures that pure Levenshtein distance-based matching is extremely fast and maintains a consistent level of performance, invariant with an increase in the number of requested matches. It takes about 5 seconds to return anywhere from 100-2000 matches in the candidate list. The Levenshtein distance-based metric also plateaus at 1000 requested matches and the trade-off may not be worth the extra computational effort. A 98% accuracy on prefix-matching appears to be a acceptable solution.

Since we pick the Levenshtein distance based metric for our string matching, we analyze its performance across RiBs from different RIPE-RIS collectors. Figures 7 and 8 illustrate its performance. Again, it is evident that the performance is fast and consistent across a variety of scenarios, at the same time maintaining a high degree of accuracy.

VII. Conclusions

In this paper, we made the case for a network processing language framework. It is our belief, based on prior research, ongoing work in the area, and current problems being faced in network management, that network management needs a novel approach of analyzing and correlating huge amounts of information. We hypothesized that network state inherently possesses semantics and structure, similar to a natural language that can be harnessed and utilized to make decisions. This hypothesis was then tested using publicly available BGP routing information to identify route prefix hijacks by analyzing the semantics of their occurrences.

Based on our results, we can confirm that our approach utilized network state semantics to *accurately* classify and predict route hijacks. These results lay a good foundation for further investigative research into developing a network processing language framework that can aid and improve network management techniques.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1421448.

References

[1] N. Feamster and J. Rexford, "Why (and How) Networks Should Run Themselves," *arXiv.org*, Oct. 2017.

[2] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

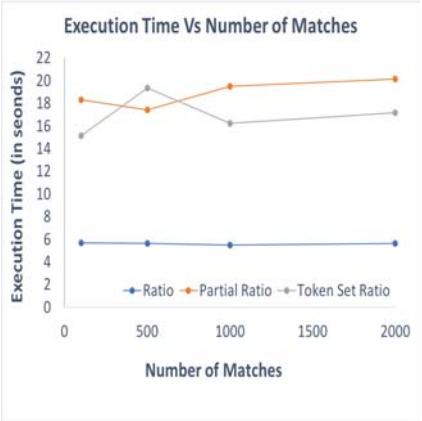

Fig. 6: Sub-Prefix Matching Runtime

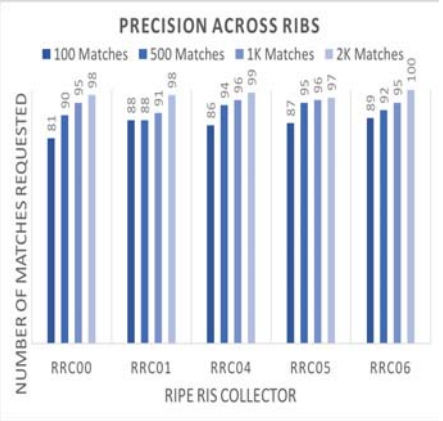

Fig. 7: Fuzzy Matching Precision

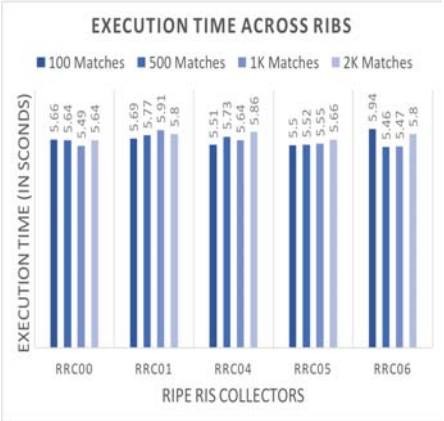

Fig. 8: Fuzzy Matching Performance

[3] J. Jiang, S. Sun, V. Sekar, and H. Zhang, "Pytheas: Enabling Data-Driven Quality of Experience Optimization Using Group-Based Exploration-Exploitation." *NSDI*, 2017.

[4] W. Li and A. W. Moore, "A machine learning approach for efficient traffic classification," in *Prof. of the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Oct 2007, pp. 310–317.

[5] R. Sasisekharan, V. Seshadri, and S. M. Weiss, "Proactive Network Maintenance Using Machine Learning." *KDD Workshop*, 1994.

[6] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, 2001.

[7] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '02*, 2002, p. 3.

[8] D. Blazakis, M. Karir, and J. S. Baras, "Analyzing BGP ASPATH Behavior in the Internet," in *Proc. of the 9th IEEE Global Internet Symposium*, 2006.

[9] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "An analysis of BGP multiple origin AS (MOAS) conflicts," in *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement - IMW '01*, 2001, p. 31.

[10] J. Qiu, L. Gao, S. Ranjan, and A. Nucci, "Detecting bogus BGP route information: Going beyond prefix hijacking," in *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007*. IEEE, 2007, pp. 381–390.

[11] H. Ballani, P. Francis, and X. Zhang, "A study of prefix hijacking and interception in the internet," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, p. 265, 2007.

[12] J. Li, D. Dou, Z. Wu, S. Kim, and V. Agarwal, "An internet routing forensics framework for discovering rules of abnormal BGP events," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 55–66, Oct. 2005.

[13] *Route Views Project Page*, Advanced Network Technology Center, University of Oregon, 2003, http://www.routeviews.org/.

[14] N. M. Al-Rousan and L. Trajkovi, "Machine learning models for classification of bgp anomalies," in *2012 IEEE 13th International Conference on High Performance Switching and Routing*, June 2012, pp. 103–108.

[15] A. Lutu, M. Bagnulo, C. Pelsser, and O. Maennel, "Understanding the reachability of ipv6 limited visibility prefixes," in *Passive and Active Measurement*, M. Faloutsos and A. Kuzmanovic, Eds. Cham: Springer International Publishing, 2014, pp. 163–172.

[16] B. Al-Musawi, P. Branch, and G. Armitage, "BGP Anomaly Detection Techniques: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 1, pp. 377–396, 2017.

[17] P. Sermpezis, V. Kotronis, P. Gigis, X. Dimitropoulos, D. Cicalese, A. King, and A. Dainotti, "ARTEMIS: Neutralizing BGP Hijacking within a Minute," *arXiv.org*, Jan. 2018.

[18] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 69, Mar. 2008.

[19] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. M. Parulkar, "ONOS: towards an open, distributed SDN OS." *HotSDN*, pp. 1–6, 2014.

[20] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of sdn/openflow controllers," in *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*, ser. CEE-SECR '13, 2013, pp. 1:1–1:6.

[21] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking - Past, Present, and Future of Programmable Networks." *IEEE Communications Surveys and Tutorials*, 2014.

[22] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: experience with a globally-deployed software defined wan," in *SIGCOMM '13: Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, Aug. 2013, p. 3.

[23] S. Ankur and R. Bruno, "Opencontrail project," Juniper Networks, Tech. Rep., November 2004, http://opencontrail.org/.

[24] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4 - programming protocol-independent packet processors." *Computer Communication Review*, 2014.

[25] C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, and D. Walker, "Netkat: Semantic foundations for networks," *SIGPLAN Not.*, vol. 49, no. 1, pp. 113–126, Jan. 2014.

[26] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software defined networks," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX Association, 2013, pp. 1–13.

[27] P. Ji, Z. Ge, J. Kurose, and D. Towsley, "A comparison of hard-state and soft-state signaling protocols," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, Karlsruhe, Germany, Aug. 2003, pp. 251–262.

[28] *RIPE Network Coordination Center*, RIPE RIR, 2018, https://www.ripe.net.

[29] C. Orsini, A. King, D. Giordano, V. Giotsas, and A. Dainotti, "Bgpstream: A software framework for live and historical bgp data analysis," in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC '16. New York, NY, USA: ACM, 2016, pp. 429–444.

[30] C. Orsini, A. King, and A. Dainotti, "BGPStream: a software framework for live and historical BGP data analysis," Center for Applied Internet Data Analysis (CAIDA), Tech. Rep., Oct 2015.

[31] "The CAIDA as-relationship dataset," Center for Applied Internet Data Analysis, http://data.caida.org/datasets/as-relationships/.

[32] "The CAIDA aspath analysis," Center for Applied Internet Data Analysis, https://labs.ripe.net/Members/mirjam/update-on-as-path-lengths-over-time.