Towards Extracting Semantics of Network Config Blocks

Kazuki Otomo[1], Satoru Kobayashi[2], Kensuke Fukuda[2], Osamu Akashi[2], Kimihiro Mizutani[3], Hiroshi Esaki[1]
1 University of Tokyo, 2 National Institute of Informatics, 3 Kindai University

***Abstract*—Configuring network devices is a main task of network operators. However, understanding and consistently updating network configuration files (config) is not an easy task especially in a large-scale and complicated networks. In this paper, we propose a semantic approach to provide better understanding of such config files, different from syntax based approaches. The key idea of the work is to extract semantics of blocks of the config files by document embedding techniques in NLP. This extraction enables us to understand context of config blocks with semantic similarity metrics instead of syntax similarity ones. Furthermore, this approach can be naturally extended to additional technical documents such as vendor's manual documents to add more specific information on the semantics of configs. We first discuss the quality of the obtained semantics for several embedding techniques, by using clustering evaluations. We next demonstrate the effectiveness of our approach with two case studies with real network configs: (1) similar config block detection and (2) automatic labeling of config block with vendor's documents.**

***Index Terms*—Network, Configuration, Semantics, Embedding**

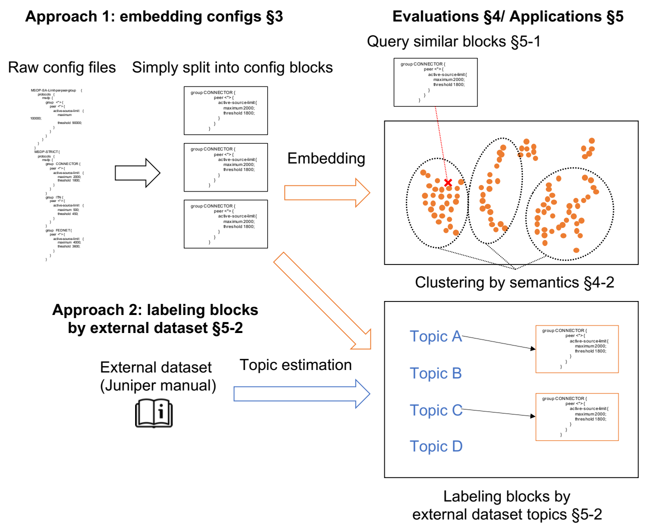

Approach 1: embedding configs §3
Evaluations §4/ Applications §5
Query similar blocks §5-1
Raw config files
Simply split into config blocks
Embedding
Clustering by semantics §4-2
Approach 2: labeling blocks by external dataset §5-2
External dataset (Juniper manual)
Topic estimation
Topic A
Topic B
Topic C
Topic D
Labeling blocks by external dataset topics §5-2

Figure 1. Overview

I. Introduction

Configuring network devices is one of the main tasks in network operation. The network devices have multiple service functions spanning multiple layers and are required to be interconnected. Thus, understanding configurations and modify them as operators intent is critical to provide sustainable network services. In particular, large-scale networks consist of multiple different vendor's equipment. For this issue, several approaches have been studied in the past literature. One approach is network verification aiming at automatically investigating the correctness of network properties (e.g., loop-free, sink-hole) with help of sophisticated analysis techniques [1]–[3]. To apply such techniques, extracting feature information from configs is a required step, and the past works built custom parsers, which is sometimes difficult to maintain for format changes by system updates or new network protocols. Another approach is to understand and compare the structure of configs. In this approach, config blocks (i.e., small pieces of configuration files) having similar syntax structure is extracted and grouped [4], [5]. This approach helps operators better understand in contents of configs. However, this approach only focuses on syntax similarities of config blocks. So, it does not consider higher-level semantics of them (e.g., a set of firewall rules in a config block).

To overcome this issue, this paper focuses on automatic extraction of semantics of config blocks in the latter approach (data-driven config embedding) for precise understanding of network configuration. In other words, we intend to automatically extract semantic information from a set of configuration instances, beyond syntax similarity. For example, we would like to find that some config blocks are different in terms of the syntax, but they are config blocks for the same services. In particular, this will be important if network operators handle multiple vendor's configs. The former approach (config emulation) cannot process different syntax configs at once due to dependencies of custom parsers.

The key idea of our approach is to extract semantics of config blocks by using document embedding techniques popular in natural languages processing (NLP) (see also Figure 1). These embedding techniques enable us to compare config blocks with semantic similarity metrics. A difficulty to apply NLP techniques to configs is that config blocks include small number of words compared to other documents and many of them are variables (e.g., IP address). In order to address these problems, we discuss and design effective pre-processing of the configuration instances for the document embedding (§III-B). We apply several document embedding techniques to pre-processed config blocks in order to understand more suitable embedding techniques in our problem (§III-C). Furthermore, we assign topic labels obtained from other technical documents (i.e., Juniper manual documents) to config blocks (§III-D). By these methods, we provide the semantic similarity of config blocks and their meanings.

Our preliminary evaluation result using config files in Internet2 [6] demonstrates that our method successfully obtains clusters of config blocks sharing the same semantics; they are not necessary to have the same syntax (§IV-B). Furthermore, we also explain two case examples to show the effectiveness of our approach. The first case shows querying config blocks in semantic similarity (§V). The results show that our embedding approach can detect semantically relevant, but different syntax cases. The second case shows the possibility of combining config blocks with external human-readable data. Although our case study is preliminary and only use a small number of dataset, it shows the posibility that our embedding method helps us automatically assign appropriate topic labels of external data (i.e., document of the target system) to config blocks.

II. Related work

Analyzing config files of network equipment has been widely studied. The main direction of the analysis is extracting abstracted information (such as network topology, network policies, etc.) with a parser which precisely split configuration files along with the vendor specific format. Fogel et al. proposed Batfish [1], a network validation tool based on a general parser of configuration files. Several improved methods are proposed for flexible verification [2] and partially automated verification [3] on the basis of Batfish. These works are focusing on reconstructing operational model from configuration files. These parser-based analyses have a general weakness: They require precise and strict parsers for all supported vendors and systems. We have to keep refining the parsers in order to adapt configuration format changes on system updates of the network devices. In contrast, this paper focuses on a data-driven approach that do not depend on the vendors and systems.

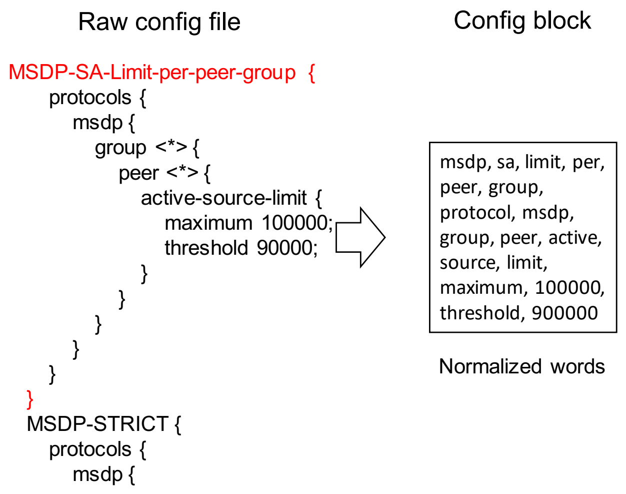
Mining raw configuration data have been also conducted in some past works [4], [5]. These works focus on finding insights from the changes of configuration files. However, these works only see changed lines (i.e., configuration commands) and do not consider structure or semantics of whole configuration file. In our analysis, we use the structure and semantics in the network configuration files to obtain vendor-independent knowledge of network configurations automatically.

In network log analysis, there are several works using semantic approaches [7], [8]. They focus on extracting semantics by using the NLP techniques (document clustering or word embedding). The nature of dataset is similar between configuration files and log data (e.g., short size text, domain specific representations, etc.). Thus, we expect that similar approach works also in configuration file analysis.

III. Semantics extraction from network configs

A. Methodology overview

In this section, we introduce our config embedding method. The config embedding consists of three processes: parsing, normalizing, and embedding as shown in Figure 1. First of all, we pre-process configuration files, more specifically splitting



```
MSDP-SA-Limit-per-peer-group {
    protocols {
        msdp {
            group <*> {
                peer <*> {
                    active-source-limit {
                        maximum 100000;
                        threshold 90000;
                    }
                }
            }
        }
    }
}
MSDP-STRICT {
    protocols {
        msdp {
```

```
msdp, sa, limit, per,
peer, group,
protocol, msdp,
group, peer, active,
source, limit,
maximum, 100000,
threshold, 900000
```

Figure 2. An example of splitting and normalizing a config file

them into config blocks and normalization of appeared words in the block (§ III-B). Next, we apply word embedding methods to obtain corresponding document vector representations (§ III-C). Furthermore, we obtain the vector representation of external documents such as technical documents to assign the topic labeling in the documents to the config blocks (§ III-D).

B. Pre-processing

The pre-processing step consists of two sub steps: split and normalization.

First, in order to build a document dataset, we split a raw config file into config blocks. The key concept of our approach is roughly splitting a config file to pieces of config blocks and obtaining semantic information without customized parsers. Thus, the granularity of config blocks can be determined arbitrarily. Our preliminary experiment shows following intuitive trends; finer granularity of the config blocks yields more pinpointing semantic information (and vice versa).

In splitting config files, the boundary of the blocks depends on vendors. Some vendors use blank line(s) as the separator, others use a more structured style such as brackets, or json-like formats. We assume the bracket-based separation in this work due to our dataset format (see § IV-A). Note that any separation rule can be selected depending on the structure of the target data. We split configs by the pairs of curly braces with four indentations as shown in Figure 2. After that, we use a config block as a document, and a raw config file is now considered as several documents (i.e., config blocks).

Then, we normalize words in parsed config blocks. Word normalization is a very basic preprocessing in NLP. We first split words by blank for each config block. Then, we remove symbols, lemmatize words, etc (this process is based on [7]). In particular in network configurations, a lot of variables (e.g., IP address, interface name, port number, etc.) appear. In this work, our strategy is to keep variables as much as possible in order to gain semantic information of the configuration file unlike our original parsing [7]. For example, interface names or port numbers (such as well-known ports) would have important semantics in network domain. However, IP address are masked due to its large diversity.

Figure 2 shows an example of block parsing word normalization. "Raw config file" describes a part of the input config file. As shown in red color, config file consists of some blocks enclosed in curly braces. As mentioned above, we use curly braces with four indentation as boundaries to split. In the block parsing, we empirically remove the outermost pairs of curly braces and its title because the blocks by the outermost pairs are too coarse to compare the semantics in our dataset. In addition, we make the split config file flatten and normalize each word in the block. Now we have lists of normalized words, and we call it as a normalized config block. One input for following embedding method is a normalized config block.

C. Embedding split config files

Next, we apply sentence embedding methods to normalized config blocks. In this work, we compare four well-known methods: Latent Dirichlet Allocation (LDA) [9], word2vec [10], Smooth Inverse Frequency (SIF) [11], and Top2Vec [12]. These embeddings assign vector representations to config blocks on the basis of their semantics. Thus, they enable us to quantitatively compare config blocks.

LDA is the most widely used method of topic modeling. Topic modeling assumes that every document is probabilistically generated from latent topics. LDA estimates word probability for each topics and topic probability in given dataset. An input of LDA is a config block and an output is a distribution of topics. We use the output distribution as an embedding of config block. Thus, we are able to measure semantic similarity by comparing topic distributions.

Word2vec is a basic method of embedding word representation. By training word2vec with config blocks, each word has one distributed representation (i.e., vector). We use a mean distribution of words as a distributed representation of a config block.

SIF is a simple but strong method to embed sentences with pre-trained distributed representations of words. A distributed representation of sentence by SIF is weighted average of word distribution, minus the principal component. In this work, we use word2vec as pre-trained distributed representations of words.

Top2Vec extracts topics from document and word vectors in jointly embedding space. Top2Vec consists of four parts: document embedding by doc2vec [13], UMAP [14] embedding, and HDBSCAN [15] clustering, calculates topic vector in UMAP-reduced space and finds topic words from detected clusters. In this work, we use doc2vec embeddings and their UMAP-reduction as embeddings of config blocks.

Now, config blocks have distributed vector representations for each method. By querying a config block, we can find other similar config blocks by calculating similarity of distributed representations. In this work, we adopt the cosine similarity as the similarity metric.

Table I
MANUAL CLASSIFICATION OF CONFIG BLOCKS

#blocks	Classes
102	policy-statement
79	prefix
45	interface
9	routing
3	class, msdp
2	radius, scheduler
1	aggregation, alias, auth, bfd, bgp, cps, ddos, dns, fpc, igmp, isis, ldp, load-balance, mld, mpls, multicast, ntp, pim, port, redundancy, rewrite-rule, rsvp, ssh, syslog, transit

D. Assigning topics to config blocks

By using external data that describes the target system of a config file, such as the manual, references, etc, we assign labels to the config blocks, by matching config blocks and the external documents. The crucial point of this method is that we do not need to prepare appropriate keywords in advance (by human labeling).

In this work, we first train a LDA model with external data, and then, we estimate topics of config blocks with learned LDA model. We use Juniper manual documents [16] as training dataset of LDA due to the constraint of the config dataset. A topic of Juniper manual documents is assigned to each config blocks. Each topic has representative keywords, thus these words can be considered as labels/annotations to the config blocks. We believe that this automatic labeling is beneficial to discuss similar config blocks in multiple vendors.

IV. Evaluation

Now, we evaluate whether our proposed method successfully extracts the semantics of config blocks. In particular, we show the difference of embedding methods discussed in § III-C in terms of clustering performance (§ IV-B).

A. Dataset

For the evaluation, we use a publicly available configuration files of network equipment [6]. As shown in [17], the configuration files are of Internet2 network from May 2015. This data consists of 10 router config files. In this work, we use one router config file. The config file has 271 blocks and 8,296 lines. The contents of the config file are mainly routing, filtering, and interface descriptions.

B. Comparison of embedding methods for config block clustering

With the embedding techniques introduced in § III-C, the config blocks can be represented as multi-dimensional vectors. It means we can compare config blocks based on their semantic similarity. Clustering config blocks is one of the effective applications of the semantic representation: it enables network operators to recognize blocks of related objects and thus decrease operational failures. Here, we compare the four embedding methods in the clustering use.

Table II
CONFIG BLOCK CLUSTERING RESULTS

	LDA	word2vec	SIF	Top2Vec
ARI	0.525±0.105	0.813±0.019	0.717±0.051	0.240±0.134
#clusters	5.9 ± 0.9	4.0 ± 0.0	7.3 ± 0.8	4.1 ± 1.5
coverage	55.9% ± 3.5	51.3% ± 1.8	48.6% ± 1.4	94.1% ± 3.4

As a ground truth of config block clustering, we manually assign 33 classification labels, representing functions or key words, to each config block. The classification labels and the number of config blocks for each class are shown in Table I. Note that here our task is not labeling but clustering, so the ground truth is just the clusters of blocks by the classes (i.e., without labels) in this evaluation. In this work, we empirically adopt HDBSCAN [15] for clustering. HDBSCAN provides clusters based on data density, and it can leave some data points as outliers that do not belong to any clusters (i.e., no other similar data points).

As the task is not labeling but clustering, we cannot use standard precision-recall evaluation. Instead, we use Adjusted rand index (ARI) [18], a widely used evaluation metric for clustering. It checks whether each pair of data points belongs to the same cluster or not in terms of predictions and annotations. Higher ARI indicates that obtained clusters are more similar clusters to annotated ones. Still, we cannot discuss the quality of clusters only with ARI because there are predominant classes and they extremely affect the metric. So, we also compare the number of clusters and coverage (i.e., ratio of classified blocks except outliers). A large number of clusters means that the model finds many semantic groups of config blocks. Furthermore, large coverage means that the model classifies many config blocks. Note that small coverage does not always mean negative results, because our ground truth has many classes with only 1 config blocks that should not form clusters.

Table II shows the clustering results for the four embedding methods. We repeat the experiment 10 times for each method. The table shows the average values and their standard deviations. We see that word2vec and SIF achieve the better performance among them. The ARI of word2vec is the best (0.813) and stable (0.019 standard deviations). The ARI of SIF is the second (0.717) and the number of detected clusters is the highest (7.3 clusters). The ARI of Top2Vec is worst (0.240) but the coverage is the largest (94.1%). We confirmed that LDA, most widely used in other data, is an not effective method for config analysis.

In these results, word2vec is the most accurate embedding method. However, word2vec is not always the best method for further analysis. As described above, ARI can be largely affected by predominant classes. In our manual inspection, word2vec accurately extracts four clusters of predominant classes (such as policy-statement and prefix) but no other meaningful clusters of minor classes. In comparison, although SIF includes some failure in the predominant classes, it detects more clusters of minor classes which is appropriate in our manual investigation (see also § V). On the other hands, Top2Vec has a large coverage but it comes from the clusters that includes many config blocks of clearly unrelated classes. Therefore, we conclude that word2vec is the best method if one needs to classify major config blocks, and SIF is a well-balanced method if one needs to extract various semantic clusters.

In the case study, we use SIF as the embedding method so that we can focus on interesting behavior of semantic analysis.

V. CASE STUDY

Here, we provide two case studies to demonstrate the effectiveness of our semantic approach.

A. *Detecting similar config blocks*

The first case study demonstrates the ability of detecting similar semantic config blocks (as shown in query similar blocks parts in Figure 1).

We make a simple question answering system that returns similar config blocks to input blocks (query blocks) by following steps. First, we embed all the config blocks by SIF (§ III-C). Then, any words or sentences in the vocabulary of config blocks can be represented in a SIF embedding space (as shown in Figure 1). When we choose a config block as the input query from all the config blocks, the system finds similar config blocks by comparing cosine similarity of SIF embeddings. The results of the query are config blocks which potentially share the same semantics of the querying config block.

Figure 3 shows two results of finding similar config by querying two BGP-related config blocks. The query A is a partial policy-statement block (diminished due to the block parsing, see § III-B) and the result is also a policy-statement block. Although these blocks have different length and the query A loses the "policy-statement" word, we succeed to detect these blocks of same semantics. The query B is a small block of describing BGP, and the result is of policy-statement. Unlike the query A result, the format of the query B is not similar to policy-statement, but our system appropriately detect a similar config block in terms of BGP, which is a broader semantics than policy-statement. Thus, this result shows that our method successfully extracts semantics of config blocks, not syntax.

In summary, we confirm that the SIF-based embedding can recognize appropriate semantics to find similar config blocks even if the querying config blocks are short and roughly parse. This tolerance is a large advantage to apply our method for multi-vendor config analysis in the future.

B. *Labeling config blocks by external documents*

The second case study shows the power of semantics extraction in config analysis. Here, we focus on the automatic labeling of config blocks with external documents. More specifically, we label config blocks by learned LDA of Juniper documents. Thus, we can assign the semantic label by the external document to config blocks.

We first scraped Juniper documents [16] on the web and find 36,596 sections. Each section mentions a topic of configuration

Table III
LABELING CONFIG BLOCKS BY TOPIC WORDS OF JUNIPER DOCUMENTS

Manual annotation	**Topic words**	**Config block**

`policy statement`	protocol, BGP, route, OSPF, filter, term, peer, IPv6	policy-statement ISP-V6-IN {
		    term allow {
		        from protocol bgp;
		        to rib inet6.0;
		        then {
		            community add COMMERCIAL-PEER;
		            accept;
		        }
		    }
		    term reject {
		        then reject;
		    }
		}

`interface`	vlan, dhcp, subscriber, dynamic, access, profile	ge-10/2/8 {
		    description ”nms-octr Pheobus Copper”;
		    vlan-tagging;
		    mtu 9192;
		    encapsulation flexible-ethernet-services;
		    unit 1201 {
		        description ”[RE]Phoebus port 2”;
		        vlan-id 1201;
		        family inet {
		            mtu 9000;
		            filter {
		                output obs-out;
		            }
		            address 64.57.19.101/30;
		        }
		    }
		}

of Juniper equipment. We train a LDA model by using all the Juniper documents. The number of topics is empirically selected to 20. With the trained LDA model, we assign a topic of a config block in order to automatically labeling the config blocks based on semantics.

Table III shows two results of Juniper document labeling; The right column is a target config block. The middle column is representative words of predicted Juniper document topic. Manual annotation indicates our manual labeling mainly based on the format of config block. As shown in the table, assigned topics of the top one properly mention about IPv6 routing (BGP). Assigned topics of the bottom one mention about VLAN. Note that our method automatically generate keyword lists; it does not require keywords list by human. This result demonstrates a possibility that we could label appropriate topics to all the config files even for different vendors if an enough number of external documents are available.

In summary, our embedding method enables us to extract semantic information of given config blocks, and then to automatically assign appropriate topic labels to them.

VI. CONCLUSION

In this work, we apply semantic analysis to network configuration files. We confirmed that with our method it is possible to learn distributed representation based on semantics from coarsely parsed configurations unlike the ordinal parser based analysis. We conducted two experimental analysis: clustering config blocks and labeling config blocks by external data. Clustering config blocks by distributed representation achieved almost 50% ARI score comparing with manual classification. In our evaluation, word2vec and SIF worked better than other methods. We also confirmed that the LDA model trained by Juniper manual data can assign semantically related words. These results show the possibility of using semantic analysis in parser-independent way. Note that our semantics approach is complementary with parser-based approaches in the further usages: For example, configuration files can be precisely parsed with the parser-based approaches in a device, and obtain broader associations of configurations in related protocols or different vendor devices with our semantic approach. Our future work has two directions; analyzing multi-vendor configuration files at once by our embedding approach, and labeling more human-readable annotations for config blocks by connecting config blocks and external information sources (such as user manuals).

ACKNOWLEDGMENT

This work is partially supported by JSPS Kakenhi 20H04185.

REFERENCES

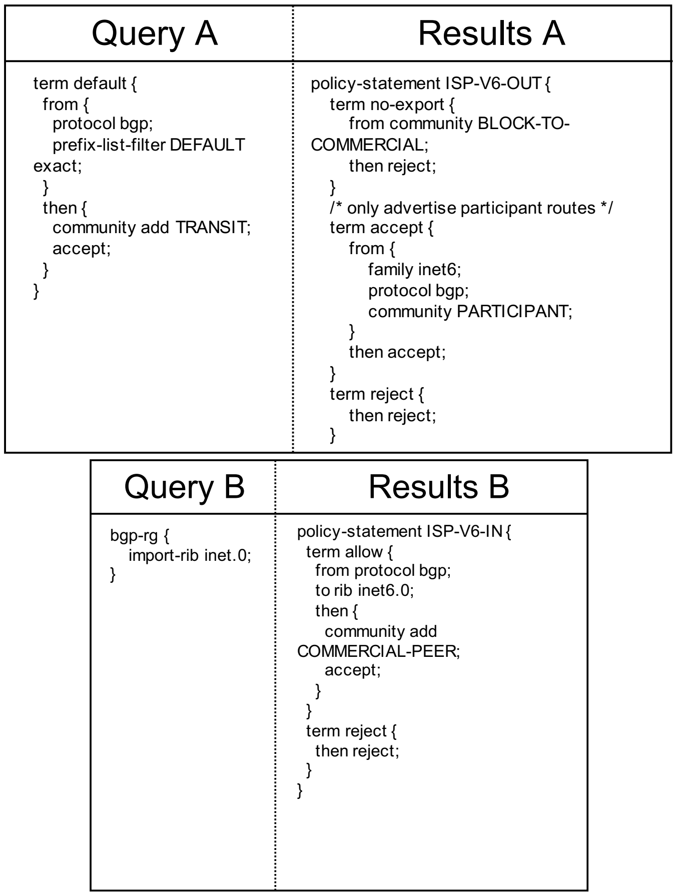
[1] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan, R. Mahajan, and T. Millstein, “A General Approach to Network Configuration Analysis,” in *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’15)*, 2015, pp. 469 – 483.

[2] R. Beckett, A. Gupta, R. Mahajan, and D. Walker, “A general approach to network configuration verification,” in *Proceedings of SIGCOMM 2017*, 2017, pp. 155–168.

[3] A. Abhashkumar, A. Gember-Jacobson, and A. Akella, “AED: Incrementally synthesizing policy-compliant and manageable configurations,” in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT 2020)*, 2020, pp. 483–495.

[4] H. Kim, G. Tech, T. Benson, and N. Feamster, “The Evolution of Network Configuration : A Tale of Two Campuses Categories and Subject Descriptors,” in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference (IMC’11)*, 2011, pp. 499–512.

[5] F. Li, J. Yang, J. Wu, Z. Zheng, H. Zhang, and X. Wang, “Configuration analysis and recommendation: Case studies in IPv6 networks,” *Computer Communications*, vol. 53, pp. 37–51, 2014.



Query A

```
term default {
  from {
    protocol bgp;
    prefix-list-filter DEFAULT
exact;
  }
  then {
    community add TRANSIT;
    accept;
  }
}
```

Results A

```
policy-statement ISP-V6-OUT {
    term no-export {
        from community BLOCK-TO-
COMMERCIAL;
        then reject;
    }
    /* only advertise participant routes */
    term accept {
        from {
            family inet6;
            protocol bgp;
            community PARTICIPANT;
        }
        then accept;
    }
    term reject {
        then reject;
    }
```

Query B

```
bgp-rg {
    import-rib inet.0;
}
```

Results B

```
policy-statement ISP-V6-IN {
  term allow {
    from protocol bgp;
    to rib inet6.0;
    then {
      community add
COMMERCIAL-PEER;
      accept;
    }
  }
  term reject {
    then reject;
  }
}
```

Figure 3. Case study: finding similar config blocks

[6] M. Cheng, "Small," https://github.com/jayvischeng/Small/tree/master/ServerData2, 2015.

[7] K. Otomo, S. Kobayashi, K. Fukuda, and H. Esaki, "Latent Semantics Approach for Network Log Analysis: Modeling and its application," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, 2021, pp. 1–8.

[8] A. Tokubi, K. Otomo, S. Kobayashi, K. Fukuda, and H. Esaki, "Automatic Document Labelling to Network Log Data," *IEICE Technical Report (in Japanese)*, vol. 120, no. 186, pp. 1–6, 2020.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, no. null, p. 993–1022, Mar. 2003.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013, pp. 3111–3119.

[11] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *5th International Conference on Learning Representations, ICLR 2017 ; Conference date: 24-04-2017 Through 26-04-2017*, Jan. 2019, 5th International Conference on Learning Representations, ICLR 2017 ; Conference date: 24-04-2017 Through 26-04-2017.

[12] D. Angelov, "Top2vec: Distributed representations of topics," 2020.

[13] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014. [Online]. Available: http://arxiv.org/abs/1405.4053

[14] L. McInnes, J. Healy, N. Saul, and L. Großberger, "Umap: Uniform manifold approximation and projection," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018. [Online]. Available: https://doi.org/10.21105/joss.00861

[15] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172.

[16] Juniper networks, "Juniper networks tech library," https://www.juniper.net/documentation/, 2021.

[17] R. Birkner, E. T. H. Zürich, D. Drachsler-cohen, L. Vanbever, M. Vechev, E. T. H. Zürich, and I. Nsdi, "Config2Spec : Mining Network Specifications from Network Configurations This paper is included in the Proceedings of the," in *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*, 2020, pp. 969–984.

[18] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.