Poster: Word embedding for deployment descriptors in NFV

Wassim Sellil Atoui[1,2], Imen Grida Ben Yahia[1] and Walid Gaaloul[2]
[1]Orange Labs, Châtillon, France
[2]Telecom SudParis, Samovar, Evry, France
{wassimsellil.atoui, imen.gridabenyahia}@orange.com, walid.gaaloul@mines-telecom.fr

***Abstract*—Virtualized networks such as NFV enable the automation of the network and decrease the need for human interventions. They ensure the availability of adequate compute, storage, and network resources for network services. To automate the network, the NFV orchestrator coordinates and manages automatically the resources to meet the requirement of network functions and services. To date, network function deployment descriptors are manually designed by service providers. Learning automatically from deployment descriptors is an important step toward fully automating the network management. In this work, we propose a word embedding approach that learns from a set of deployment descriptors a vector representation. This approach is shown to be useful for recommending deployment descriptors.**

***Index Terms*—Network Function Virtualization, NFV, Word Embedding, Deployment descriptors, VNFD.**

I. Introduction

Currently, in NFV, the onboarding and the deployment of new virtual network function (VNFs) or network services (NSs) requires specifications provided by the service owner about the deployment requirements. The specifications are deployment descriptors that describe for a network service [1], the virtual network function (VNFs) that compose it, the topology of interconnecting the VNFs, and the data flow direction between them. Also, the deployment descriptors for the VNFs indicate the deployment and operational behavior in terms of connectivity, interface, and virtualized resource requirements.

Learning automatically from deployment descriptors is an important step toward fully automating the network management. In this work, we propose a word embedding approach for deployment descriptors, that learns from a set of deployment descriptors a vector representation for each element that compose the descriptors. The vector representation captures the semantic similarity between the elements which helps to solve diverse problems such as deployment description completion, recommendation, and classification [1]. Word embeddings are very effective in solving many Natural language processing (NLP) problems such as Automatic summarization, Machine translation, Named entity resolution, Sentiment analysis, Information retrieval, etc. They are a numerical word representation where words that have the same semantic have a similar representation. In NLP, word embeddings are learned from text based on the distributional hypothesis that emphasis that words that occur in the same contexts tend to have similar meanings. The context of deployment descriptors is different from the text. The context of an element is not a flat window of words. We propose therefore a graph model for deployment descriptors that we use to extract the context for each descriptor element. More precisely, we focus our attention on VNF descriptors (VNFDs) and propose a model to represent the elements that compose them.

Annex to ISBN 978-3-903176-28-7

The rest of the paper is organized as follows: In section II, we describe a hierarchical model for the VNFD elements. This model is used afterward to learn the word embedding from the deployment descriptors. In section III, we conduct some experiments to evaluate the performance of the proposed word embedding approach and discuss the results. Finally, in section IV, we conclude our work and present future perspectives.

II. Learning word embedding for VNFD deployment descriptors

In this section, we present an implementation approach of word embedding for deployment descriptors in NFV. More particularly, we focus our attention on VNFDs. In our previous work [1], we have proposed to use word embedding directly on the VNFD files. For that, we proposed to breakdown the files into a set of tokens. Tokens are elements that constitute the VNFD files, it include the data model elements and also the special characters that structure the VNFD file, i.e. indentation, text return, tags, brackets, etc. The shortcomings of this approach are in two folds: Firstly, we used a flat context (direct surrounding tokens) to learn the embeddings, which is less efficient given that the structure of the VNFDs is hierarchical. Secondly, the proposed approach is strongly dependent on the file structure of the VNFDs. To overcome these shortcomings, we propose an abstract model for the VNFDs. This model is used to extract a more significant context for learning word embedding. The model represents the VNFD elements independently of their file structure in a hierarchical way. Moreover, the relation type between the VNFD elements is used to augment the context of the VNFD element. We first start in this section by defining the VNFD model. Afterward, we describe briefly our approach to learning the word embeddings.

A. VNFD model

VNFD is a configuration file that describes the NFV infrastructure resource requirements for a VNF in a service provider environment and the operational behavior of the VNF including lifecycle events (e.g. scaling, upgrading).

The ETSI NFV [2] has released a specification that defines the requirements for the structure and format of a VNFD. The VNFD is composed of one or many virtual deployment units (VDUs) that describe the deployment resources and operation behavior of a VNF component (VNFC). VDUs are virtual machines that host the VNF or parts of it. Each part of the VNF is a VNFC and can be deployed on one or more VDUs. Each VDU is characterized by, among others, the software image loaded on it and the resources needed to deploy it. A VDU describes mainly the virtual compute (VC), virtual storage (VS) and virtual memory (VM) resources that are necessary for deploying a VNFC and it could be linked via connection points (CPD) to other VDUs or to external VDUs that belong to other VNFs via external CPD. Virtual links in the VNFD indicate how the VDUs are connected and via which CPD.

As the structure of a VNFD can be mapped to a graph, we choose graph theory to represent a VNFD model as a tree-like structure. For sake of simplicity, we restrict our definition to the general elements that compose a VNFD regardless of any data model. We consider for that matter the ETSI NFV specification [2].

Fig. 2, illustrates a basic example of a VNFD representation of a firewall. Fig. 1 shows the basic elements that compose the VNFD model representation. The nodes in the VNFD model represent the different component instances that compose the VNFD. Each component instance has one of the following types: *VNFC, VDU, VM, VS, VC or CPD* and may have one or more attributes. The root node represents an instance of the type VNFD. Fig. 2 presents an example where the root node v_1 is an instance of type VNFD and has the attribute "name" which has the value "vFirewall". Graphically, we use ":" in the node label to indicate the component from which the instance is derived. For sake of simplicity, in our example, we define one attribute for each node and we indicate it between "()" in the node.

The relations between the nodes (component instances) can be one of the following three types: *composition, allocation, or connection*. A composition relation between two nodes v_1 and v_2 indicates that the instance v_1 is composed by the instance v_2. That means that v_2 is part of v_1 and cannot exist without it. The composition relation exists typically between (i) VNFDs and VNFs, (ii) VNFCs and VDUs, and (iii) VDUs and CPDs. Graphically, a 1:1 composition relation is represented by a solid line. For example, in Fig. 2, the instance of VDU v_4 is composed of the instance of CPD v_7. In case a 1:n composition relation exists between multiple nodes, we use a *composition gateway* to connect the nodes. Graphically, the composition gateway is represented by a grey diamond with the label "C". In our example, the instance of VNFC v_2 is composed of two instances: VDU v_3 and VDU v_4.

An allocation relation between two nodes v_1 and v_2 indicates that v_2 is a resource that needs to be allocated to v_1. The allocation relation can only exist between VDUs on the one hand and VS, VC, and VM on the other hand, i.e. only the virtual storage, virtual memory, and virtual component

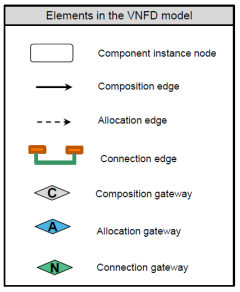

Fig. 1. Graphical representation of the elements used in a VNFD model

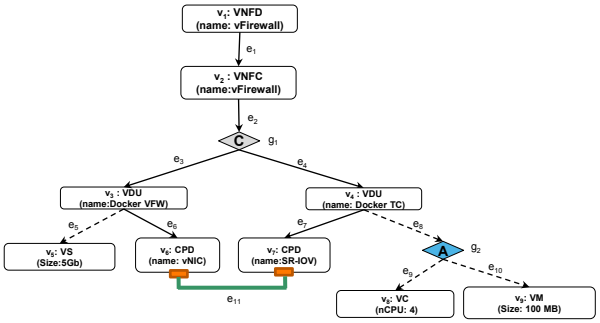

Fig. 2. An example of a vFireWall VNFD represented graphically as a tree-like structure

resources can be allocated to the VDUs. Graphically, the allocation relation is represented by a dashed line. As an example, in Fig. 2, the VS v_5 is allocated to the VDU v_3. In case multiple resources are allocated to a VDU (i.e. 1:n relation), we use an *allocation gateway* to connect the nodes. Graphically, the gateway is represented by a blue diamond with the label "A". In our example, the VC v_8 and the VM v_9 are allocated to the VDU v_4.

A connection relation can exist between the component instances of type CPD and is a bidirectional relation. It represents the virtual links that connect the CPDs of the internal VDUs or to external VDUs of external VNFs. Graphically, a connection relation is represented by a solid green line. As an example, in Fig. 2, the relation between the CPD v_6 and the CPD v_7 indicates that a virtual link connects the two connection points. We also introduce the *connection gateway* (as shown in Fig. 1) that allows to connect multiple connection points. The connection gateway is graphically represented by a green diamond with the label "N".

B. Word Embedding

The word embedding approach is SkipGram [3]. It learns a representation of the elements of the VNFD model. The goal is to learn a representation in a vector space, where each unique element in the deployment descriptor data set is positioned in the vector space such that elements that share common contexts in the corpus are located near one another in the space.

C. VNFD context augmented SkipGram

The distributional hypothesis that emphasis that words that occur in the same context tend to have similar meanings

is not applicable for deployment descriptors the same way as in NLP. As shown in the previous section, the context of each element in the deployment descriptors is not a flat set of elements that surround it. The elements are rather related directly and indirectly to other elements. Moreover, the graph structure of the VNFD model shows also that different dependencies could occur between the elements. It is therefore important to account for a hierarchical context when learning the embedding for the deployment descriptors. Also, the context needs to be augmented with the relational type that relates the VNFD elements together. We define the window size of the target element as the area to which the context elements are connected. For example, window size = 1, means that the context elements are the elements that are directly related to the target word. Window size = 2, include also the elements that are related to the context elements of the first context area.

As an illustration, We suppose that the context window size = 1. The context of the target element v_2 in Fig.2 is: (vFirewall, Docker VFW/Composition/VNFC),(vFirewall, Docker TC/Composition/VNFC), (vFirewall, vFirewall/Composition/VNFD). Note that even when vFirewall appeared two times (in v_1 and v_2), the context of the elements suggested a different meaning.

III. Experimental Results

To evaluate our experiment, we compare our augmented version of the SkipGram approach with the classical one. For that, we use our previous work [4] in which we have proposed a framework that complements a given network deployment descriptor with appropriate metadata and its associated values, selects, and recommends descriptors to ease and automate the deployment phase of VNFs. The approach is based on the classical SkipGram embedding approach and DNN methods. More particularly, we employed two methods of deep learning, CNN and LSTM.

Our framework encompasses three phases: 1) the Data preparation phase, 2) the Training phase and 3) the DNN execution phase. The preparation phase aims to transform the input set (i.e. deployment descriptors) into a data set that is used by the DNN methods in the training phase. During the training phase, the DNN methods learn two models, a CNN model for the recommendation task and an LSTM model for the completion task. During the third phase, the generated LSTM and CNN models are executed. In the preparation phase, we use the two word embedding approaches (classical and augmented Skipgram)

To evaluate the performance of both the embeddings (classical and augmented SkipGram), we focus on the effectiveness of the two deep learning models: CNN model for the descriptor recommendation task and LSTM model for the descriptor sequence prediction, with respect only to the VNFD elements.

A. Environment settings

In our experiment, we consider approximately 100 VNFDs collected from various organizations and private telco vendors. The data set include several variations of VNFDs for different VNFs such as Mobility Management Entity (MME), virtual Customer Premises Equipment (vCPE), virtualized Subscriber Data Management (vSDM), etc. The VNFDs are YAML files that follow the TOSCA data model. We implemented a python program to construct the VNFD models from the data set. The number of nodes extracted, with respect to their type, are 65 VNFC, 453 VDUs, 453 VCs, 421 VMs, 357 VSs, 520 CPs.

tab. I. Evaluation of the LSTM model using different output length

Output length	SkipGram	Context augmented SkipGram
5	0.87	0.9
10	0.82	0.85
30	0.78	0.81
70	0.6	0.69
180	0.43	0.51

tab. II. Experiments on the CNN model using different input sizes

Input Size	SkipGram	Context augmented SkipGram
20	0.24	0.29
50	0.31	0.31
100	0.44	0.47
200	0.6	0.64
500	0.84	0.88

We evaluate the CNN and LSTM model by taking into account the two embedding methods. The parameters of the models are the same as in the previous work. The accuracy of the classification is measured by varying the input length of the CNN model and the output size for the LSTM method. The results are shown in tab.I for the CNN model and tab.II for the LSTM model.

We can notice that augmented embedding gives better results. That makes sense, because the more we add accurate context information about the description, the more the CNN and LSTM can extract features and be able to classify it to the best descriptor file.

IV. Conclusion

We proposed in this work a word embedding approach for deployment descriptors, referred to as token embedding, that learns from a set of deployment descriptors a vector representation for each element that compose the descriptors. The goal is to enable service providers to learn from network service deployment descriptors. The results show that our approach can be effective in learning better embeddings for the deployment descriptors. Our next step involves developing a descriptor generation engine that could learn from the previous deployment of network services to suggest adequate configurations.

References

[1] W. S. Atoui, I. G. Ben Yahia, and W. Gaaloul, "Using deep learning for recommending and completing deployment descriptors in nfv," in *IEEE Conf. on Net. Softwarization (NetSoft)*, pp. 233–235, June 2019.

[2] ETSI, "Gs nfv-ifa 011." https://standards.globalspec.com/std/13271186/gs-nfv-ifa-011, 2019.

[3] 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 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 3111–3119, Curran Associates, Inc., 2013.

[4] W. S. Atoui, I. Grida Ben Yahia, and W. Gaaloul, "Virtual network function descriptors mining using word embeddings and deep neural networks," in *IFIP/IEEE Symp. on Integrated Net. and Service Management (IM)*, pp. 515–520, April 2019.