An Ontology-Based Information Extraction System for bridging the configuration gap in hybrid SDN environments

A. Martínez*, M. Yannuzzi*, J. E. López de Vergara†, R. Serral-Gracià*, W. Ramírez‡

*Networking and Information Technology Lab (NetIT Lab), Technical University of Catalonia (UPC). Spain
†Department of Electronics and Communications Technologies, Autonomous University of Madrid (UAM). Spain
‡Advanced Network Architectures Lab (CRAAX), Technical University of Catalonia (UPC). Spain
Email: {annym, yannuzzi}@ac.upc.edu

***Abstract*—Hybrid Software-Defined Networks (SDNs) are growing at a remarkable speed, so network administrators need to deal with the configuration of a plethora of devices including OpenFlow elements, traditional equipment, and nodes supporting both OpenFlow and traditional features. The OpenFlow Management and Configuration Protocol (OF-CONFIG) is positioned as a solid candidate for the remote configuration of OpenFlow devices, but the fact that OF-CONFIG relies on NETCONF for its transport constrains its potential considerably. Indeed, the lack of comprehensive and standardized data models has hindered the utilization of NETCONF itself in traditional networks, and will likely confine OF-CONFIG to an elementary set of configurations until the expected data models arrive. In this paper, we present a semantic-based approach that eases and automates the configuration of network devices while complementing the capabilities of OF-CONFIG and NETCONF. Our main contributions can be summarized as follows. First, we have formalized the semantics of the switch/router configuration domain using the Web Ontology Language (OWL). Second, we have developed an Ontology-Based Information Extraction (OBIE) system from the Command-Line Interface (CLI) of network devices. Third, we have defined a learning algorithm that enables automated interpretation of CLIs' configuration capabilities in heterogeneous (multi-vendor) network scenarios. The potential of our approach is demonstrated through experiments carried out on different network elements.**

I. Introduction

With network programmability in the spotlight, Software-Defined Networks (SDNs) have rapidly become a major trend in the ICT field. Telecom providers and device vendors concur that SDNs will lay down the foundation for next-generation networks, given their potential for achieving higher flexibility and openness while dramatically reducing costs. A paradigm shift of this nature can clearly transform network management practices, and pave the way for reaching the desired goal of network automation. The OpenFlow protocol [1], and most recently the OpenFlow Management and Configuration protocol (OF-CONFIG) [2], have become key components for controlling and managing SDNs. OpenFlow standardizes the interactions between an SDN controller and the switches under its control. However, it does not provide the functions that are required for configuring queues, ports, assigning IP addresses or any other configuration toward the device. The OF-CONFIG protocol was recently defined by the Open Networking Foundation (ONF) precisely to that end.

A crucial part of the OF-CONFIG specification is that the configurations are transported on NETCONF [3] —a protocol which also provides mechanisms for the configuration of devices in traditional networks. Unfortunately, NETCONF itself has not gained momentum yet, so it remains to be seen if it will finally become the protocol of choice [4]. In this regard, industry sources state that nearly 95% of network devices are still configured through proprietary Command-Line Interfaces (CLIs) [5]. The reason for this is the lack of comprehensive and widely accepted data models. This gap was recently filled by YANG [6], a candidate language for developing standardized data models for NETCONF. Still, four years after its standardization, only few YANG data models have found broad acceptance [7].

Another relevant aspect is that SDN is certainly not necessary for all parts of the network [8]. Moreover, a full replacement of the underlying infrastructure is neither affordable nor feasible for many administrators, which indicates that SDNs will need to coexist and interact with traditional networks for several years. For this reason, hybrid approaches to SDN —a mix of SDN-enabled and traditional network devices— are positioned as strong candidates to ease the transition to new and more flexible network environments [9], [10]. Accordingly, legacy infrastructures will continue to play a crucial role in the SDN future, and will likely give place to new challenges and opportunities in the management field. Although OpenFlow and the elementary configurations supported in OF-CONFIG can suffice for managing OpenFlow devices, network configuration tasks clearly entail much more than configuring flows. Consider for instance requirements such as the configuration of usernames and administrative privileges for authenticated access through CLIs, the configuration of a link-state routing protocol (*e.g.*, OSPF), or a switching protocol (*e.g.* MPLS). Accordingly, the heterogeneity of hybrid SDNs will require of a flexible management model where configurations are not only performed on a per-flow basis.

In this context, this paper presents a semantic-based approach for complementing the capabilities of OF-CONFIG and NETCONF for the remote configuration of network devices. The main goal is to develop a non-disruptive solution that supports the configuration of hybrid devices, while complying with key drivers for SDNs, *e.g.*, abstraction and automation of management tasks. To this end, we propose an Ontology-Based Information Extraction (OBIE) System from the CLI of network devices. Overall, our approach exploits the knowledge already available in CLIs, in an effort to automatically reconcile the semantic and syntactic differences between heterogeneous configuration environments. Our contributions are as follows. First, we formalize the semantics of the switch/router configuration domain using the Web Ontology Language (OWL). Second, we present our OBIE system from the CLI of network devices. Third, we develop an ontology-based learning algorithm that enables automated and highly precise interpretation of CLI configuration capabilities in heterogeneous (multi-vendor) network scenarios.

The remainder of this paper is organized as follows. Section II briefly motivates our approach in the context of current research. Section III introduces our semantic approach and describes the architecture and learning algorithm of our OBIE System. Section IV presents the experimental results carried out over the configuration space of two widely used routers. Finally, Section V concludes the paper.

II. Related work

The research community has devoted considerable efforts to overcome the complexities in network management [11]. Some of these efforts have explored alternative fields to standardization which can further assist in networking. For instance, ontologies from Artificial Intelligence (AI) offer a promising path to achieve interoperability in domains where standards have not succeeded. In the networking domain, several initiatives have also explored the AI path to target network management [12], [13], [14], [15], [16], [17]. Given the broad scope of network management and the numerous functions that it entails, many of these solutions approach different aspects of the problem; for example, the need to unify underlying network management data models [16], autonomic network management [15], [17], integration of management data [14] or the issue of multi-vendor configuration management [12], [13]. For an in-depth study on the application of semantic technologies to the network management domain we refer readers to [18].

With respect to ontologies applied for configuration management, the work in [12] proposes the use of ontologies to describe a NETCONF workflow. However—with the somentioned limitations of NETCONF—herein we focus on the use of CLIs for the configuration of network devices. Moreover, the research in [13] is the one mostly aligned to our work. In that paper, an ontology-driven framework is proposed to address the semantic heterogeneity of network management domains. Their main contribution is a semantic similarity function that enables mappings between ontologies. They further apply their solution to the router configuration domain and model CLI commands on ontological structures. However, their framework assumes that ontologies are given in advance for each device, but the task of building ontologies is already challenging and sufficiently complex by itself.

Despite of the numerous initiatives regarding the application of ontologies and AI-based techniques to approach network management, the exploitation of semantic technologies continues to be a research challenge. There are still many paths to be explored in order to benefit from such technologies. Moreover, none of these works have explored the field of OBIE as a means to enable interoperability in the network configuration domain. We strongly believe that a solution which can assist in the configuration of network devices whenever OF-CONFIG or NETCONF are not suitable, is less a matter of developing new ways of managing the network or adding new protocols that boost complexity, but more of adapting well-known techniques that have proven to be absolute trends for years in the configuration field. For this reason, legacy Command-Line Interfaces can help bridge the gap between traditional and SDN-based networking.

The rationale behind our approach is that almost every network element —either legacy or SDN-enabled— features a CLI for configuring the device, and thereby control and manage the technologies and protocols that run through them. CLIs are often large and heterogeneous in content, structure and semantics. However, they provide information —encoded in the form of natural language— aimed to help and guide network administrators in the manual use of the configuration environment. Indeed, this information could be automatically acquired and transformed into useful configuration knowledge. The notion of extracting knowledge from already available text-based resources has been a field of increased interest among researchers of many other domains as a means to enable numerous applications (*e.g.*, question-answering, decision support systems, etc.). To the best of our knowledge, CLIs have never been explored from an OBIE perspective in the network management arena.

III. Approach

As stated earlier, our central hypothesis is that configuration knowledge can be automatically acquired from already available CLIs, by exploiting both *(i)* the information provided in the *"help"* feature, and *(ii)* the knowledge inherent to the hierarchical arrangement of commands. To this end, we have developed an Ontology-Based Information Extraction (OBIE) System, which relies on shallow Natural Language Processing (NLP) tools and other semantic technologies to exploit the information natively provided in CLIs. Furthermore, we have developed an ontology for the switch/router configuration domain which will be used to guide the Information Extraction (IE) process. Overall, our system aims to identify relevant information from the CLI in an effort to determine the semantics of the configuration space.

We anticipate that the methodology developed in this paper is general in scope, so it can be extrapolated to other domains

wherein the configuration environments are distributed, heterogeneous, command-based, and most importantly, hierarchical. For instance, our general approach can be applied for the configuration of distributed medical equipment in a health facility, printer stations in large corporations, etc.

A. Architecture Overview

The general architecture of our OBIE system is shown in Fig. 1. Observe that our system takes two inputs, namely, *(i)* the switch/router CLI —as natively provided by vendors— and *(ii)* the switch/router configuration ontology —specified by us. Furthermore, the output of our system is a device-specific ontology, which is the result of populating the domain-ontology with instances of configuration commands according to the semantics of each CLI space. These device-specific ontologies are further stored in a repository to enable potential functionalities of third-party applications, *e.g.*, a Network Management System (NMS) requesting the commands for setting an interface IP address for dissimilar device models.

Our system's architecture is based on a modular design which accommodates several components into two functional blocks, namely, the *offline* and the *online* modes (cf., Fig. 1). The *offline* mode resolves the *semantic* and *syntactic* dissimilarities between CLIs, in an effort to automatically bring its semantics to a common reference model. The way in which we exploit CLI information to aid in the instantiation of commands and how these disparities are actually resolved is the core of our Semantic Learning Engine and will be further explained in Section III-D. The *online* mode provides an interface to third-party applications (*e.g.*, a Network Management System) which can significantly benefit from a system that abstracts the complexities of multi-vendor configuration environments. Without such a tool, network administrators are forced to gain specialized expertise for every available device. Overall, the *offline* and *online* modes aim to mitigate the efforts related to the configuration of devices in heterogeneous environments. The execution of both functionalities completes the configuration cycle, from the semantic definition of every configuration space to the automatic retrieval of commands, regardless of the underlying heterogeneity.

B. The Ontology for the Switch/Router Configuration Domain

The developed ontology for the switch/router configuration domain represents the conceptualization of the domain in the most generic and neutral way —*i.e.*, regardless of vendor's specifics. The main driver for building such an ontology is based on the fact that the knowledge expressed in CLIs —apart from proprietary developments— is mostly related to well-known concepts and technologies. This is mainly because device manufacturers tend to keep their products close to standards, as a means to ease interoperability and comply with regular configuration features. Nevertheless, the use of terminologies to name commands and variables is what most likely differs among vendors —either because different representations are used for the same semantics (a syntactic problem) or the complete opposite, same terminological representations are used for different semantics (a semantic problem). In light of this, it is not what CLIs provide what mostly concerns network administrators, but most importantly, the way in which this knowledge is expressed —*i.e.*, the use of dissimilar terminologies and their corresponding semantics.

The ontology has been defined taking the knowledge provided by networking experts, in addition to information extracted from configuration manuals. For this reason, key concepts and relations of the switch/router domain were unambiguously identified and formally encoded in OWL. Although our solution is general in scope, the ontology must meet a minimum of requirements. First, the domain lexicon is integrated in the ontology. Second, the ontology must be defined in two layers, namely, a *resource* layer and an *operation* layer (cf., Fig. 2) —very similar to the approach followed by authors in [19]. The former defines the entities, concepts and

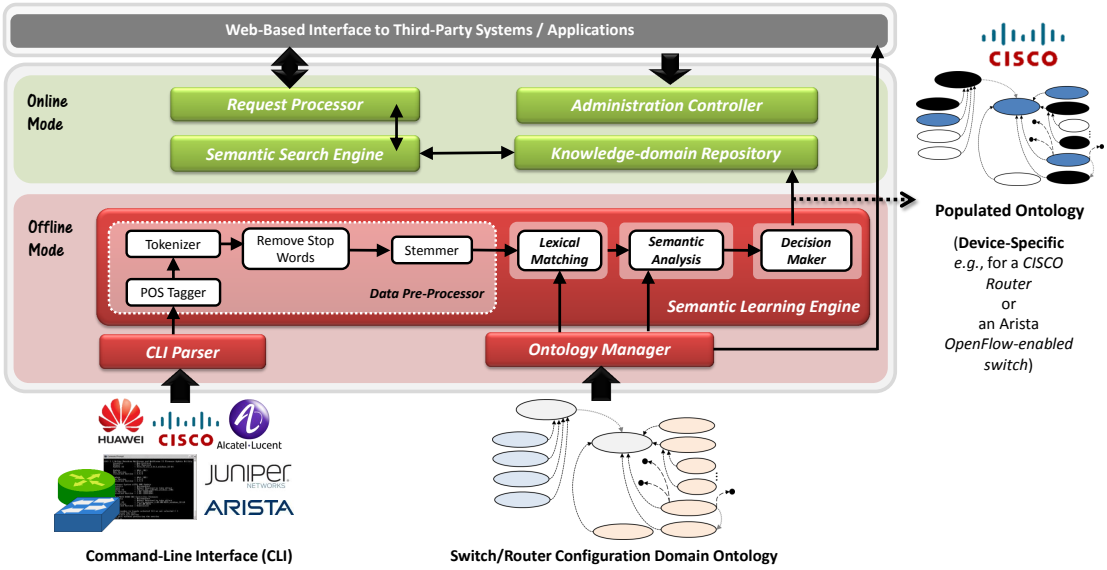

Fig. 1: General architecture of our OBIE system for switch/router command instantiation from CLIs.

resources of the domain —both, physical (*e.g.*, an interface or a LAN port for the routing domain) and virtual (*e.g.*, a routing protocol or the OSPF hello interval). Moreover, the latter defines the functional concepts of the domain, *i.e.*, the set of operations that can be performed over virtual and physical resources —*e.g.*, configure a router host-name or remove a static IP route, etc. Notice that concepts in the *operation* layer are specified in terms of verb phrases (*e.g.*, set, delete, configure, show, etc.) and semantically associated to concepts in the *resource* layer (cf., Fig. 2). In short, a *resource* represents a component that can be supplied or consumed in an *operation*.

The resulting ontology represents over 600 resources and near 320 operations [20]. We have developed our ontology, based on the use of all OWL constructs (classes, individuals, properties, restrictions, etc.). We have defined hierarchical (*i.e.*, taxonomic *"is-a"* type of relations) and non-hierarchical relationships between concepts, in an effort to improve the semantics of the domain. Moreover, we have modeled user-defined data-types using the pattern facet restriction feature of OWL2 to define custom types to match regular expressions. This feature will enable us to validate domain-specific types of data, *e.g.*, to identify ranges or an IPv4 address. Notice that the knowledge encoded in the ontology will help us resolve ambiguity. For example, we can determine if the occurrence of a term corresponds to a certain concept by identifying an address format or a measurement unit.

C. Off-line Functionality

The overall aim of the *offline* functionality is to instantiate configuration commands into their corresponding semantic classes, in an effort to automatically build device-specific ontologies from their CLIs. This mode is comprised of three modules, namely, the CLI Parser, the Ontology Manager and the Semantic Learning Engine (cf., Fig. 1).

The **CLI Parser** provides the functions to breakdown the CLI into its structural parts, namely, *commands* (cmd) or *variables* (var) and *help descriptors* (help) (cf., Fig. 3). The reason for doing this is that only commands and variables are aimed to be instantiated. The information provided in the help descriptors will be used to determine the semantics of each level and assist in the disambiguation process. The fact that "helps" are aimed to guide network administrators on the manual use and interpretation of the CLI, makes this information likely to

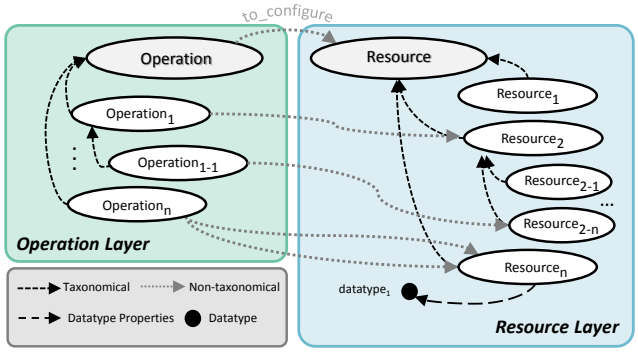

Fig. 2: Layered structure of the Domain-Ontology.

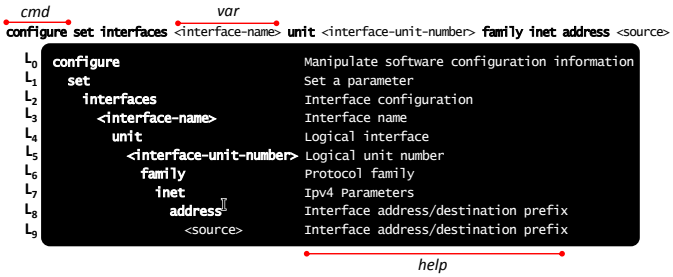

Fig. 3: Example of a typical router configuration statement.

reference common networking terminologies —instead of custom and vendor-related ones. All in all, the semantics provided by vendors must converge at some point. Otherwise, their solutions would be unattainable and impractical, and the learning curve expensive. Though networking-related terms can occasionally be misleading, an "IP Address" will have to be referred as such if vendors want to dominate in a market already flood by serious interoperability issues. Even if helps do not explicitly provide a reference for disambiguation, the context —*i.e.*, information of contiguous levels— can help determine a term's sense. This feature will be further exploited in our learning approach. Furthermore, the parser also browses through the hierarchy in order to build the complete set of executable configuration statements —i.e, valid sequences of commands and variables which semantically represent one or several configuration operations. Each of these statements are target of instantiation within our system. A typical configuration statement is shown in Fig. 3.

The **Ontology Manager** provides an interface to the domain ontology. More specifically, it supports the functions required to *(a)* load the ontology, *(b)* enable access to OWL constructs —via the OWL API [21]— and *(c)* expose available configuration operations via Web Services to third-party applications.

The **Semantic Learning Engine** includes the algorithms for extracting configuration knowledge from heterogeneous CLIs. This module can be further differentiated in four components, namely, *(i)* Data Pre-processor, *(ii)* Lexical Matching, *(iii)* Semantic Analysis, and *(iv)* Decision Maker (cf., Fig. 1). The approach to Information Extraction (IE) will be described in the next section.

D. An Ontology-Based Learning Approach

To achieve the overall goal of our system we target the process of IE in two stages. In ***Stage 1***, we look in the CLI for *(i)* verb phrases and *(ii)* relevant concepts of the switch/router domain with respect to particular components of the *resource layer* of our domain ontology. In ***Stage 2***, we bring this knowledge together and reason over it, in an effort to determine the operation that best defines the semantics of a level, with respect to the components of the *operation layer*. In order to ease explanations, we will consider the configuration statement shown in Fig. 3 through all this Section. Notice that the sequence of commands and variables semantically represent the configuration of an interface IP address.

The first component in the IE workflow is the **Data Pre-**

Processor which combines shallow NLP tools for basic data pre-processing. The process includes the following: a *Part-Of-Speech (POS) Tagger* for verb phrase identification, a *Tokenizer* that separates data into tokens and Removes Stop Words, and a *Stemmer*, which reduces inflectional forms of a word to its common base form. Notice that the information present in the CLI is likely to be short and concise —sometimes even insufficient to be self-explanatory. The lack of verbosity and proper grammar restricts the content of CLIs to *(i)* concepts (*e.g.*, Level_2 and Level_3 in Fig. 3) and *(ii)* verb phrases (*e.g.*, Level_1 in Fig. 3). This significantly simplifies the scope of the semantic search.

After that, we perform **Lexical Matching**, *i.e.*, we make extractions with respect to particular components of the domain ontology. Consider the ontology to be a semantic graph (cf., Fig. 4 (a)) where nodes represent concepts and edges relations between concepts. The output of this stage is a set of "activated" nodes for every identified entity in the CLI —the notion of "activated" nodes is depicted in Fig. 4 (b)). Lexical matching allows both partial and exact matching. In the case for which a term matches several ontological concepts, we keep the Least Common Subsumer (LCS), *i.e.*, the most concrete taxonomic ancestor. To illustrate this, consider Level_2 in our example. The CLI information for this level is "Interface Configuration". Accordingly, candidate concepts (lexical matching) will be: $\langle interface \rangle$, $\langle ethernet - interface \rangle$, $\langle 100G - Interface \rangle$, $\langle 10G - Interface \rangle$, and $\langle interface - name \rangle$. From the ontological structure, we know that these concepts have a subsumption relationship. The $\langle ethernet - interface \rangle$, $\langle 100G - Interface \rangle$ and $\langle 10G - Interface \rangle$ concepts are subtypes of $\langle interface \rangle$, while $\langle interface - name \rangle$ is an attribute (*i.e.*, an ontological data property) of the latter. In the absence of information we select the LCS, *i.e.*, we generalize to the $\langle interface \rangle$ concept. Moreover, if exclusive properties of a concept are discovered in subsequent levels, we can further select a specification of the concept. This functionality is performed by the inference stage of the Semantic Analysis. Notice that there are cases for which candidate concepts do not have a LCS. In these cases, concepts are considered disjoint, *i.e.*, only one can accurately define the semantics of the given CLI term. Further clustering and semantic relatedness will aid in the disambiguation for these scenarios. To illustrate this case consider the following. For Level_8 in our example, the lack of verbosity in the CLI can generate ambiguity between the ontological concepts, $\langle mac - \underline{address} \rangle$ and $\langle ip - \underline{address} \rangle$. Although in principle we lack information to disambiguate between both concepts from a lexical perspective, we know in advance that only one can properly define the semantics of the given term. For this reason, we identify them as disjoint concepts and further semantic analysis will allow us to select the best candidate concept.

It is important to realize that even if concepts are not identified by lexical matching, mainly because of the use of custom or dissimilar terminologies, the **Semantic Analysis** can identify relevant concepts by inference. Therefore, we do not make limited use of the ontology —such as names of classes— moreover, we use the ontological structure to enhance our assessment. Notice that due to CLIs intended purpose, it is reasonable to think that they cannot be conceptually radical as they are expected to converge at some point to domain referents (e.g., protocols and standards). The Semantic Analysis stage can be further differentiated into, clustering, inference and semantic relatedness computation, as shown in Fig. 4.

Clustering and **Inference** are iterative stages where "activated" resources are grouped into semantic clusters (cf., Fig. 4 (c)). We form clusters between *directly* connected resources of adjacent levels. Notice that the notion of nodes being part of fully interconnected clusters is based on the premise that commands are arranged in the hierarchy by association —*i.e.*, commands become more specific down in the tree structure. Accordingly, domain concepts in contiguous levels are expected to be semantically related to a certain extent —directly or not. The degree to which entities are actually related will depend on the granularity of the CLI —which varies for every vendor. For this reason, clustering is not sufficient and we require a means to measure the degree to which they are semantically related. Furthermore, we perform semantic inference (cf., Fig. 4 (d)) with a two-fold purpose. First, to identify relevant concepts which were not identified by lexical matching but which can be derived from a semantic analysis. For this, we reason over equivalent axioms of the ontology, which will allow us to "activate" a concept whenever a set of conditions is fulfilled for a *defined* class. For example, if we identify the $\langle system \rangle$, $\langle date \rangle$ and $\langle time \rangle$ concepts (cf., nodes m, o and p in Fig. 4 (b)), we can infer from the equivalent axioms of our ontology that we are referring to the system "$\langle clock \rangle$" and further "activate" this concept (cf., node n in Fig. 4 (d)). Observe that this is an iterative process, so after node n is activated, nodes m, o, p, and n will be clustered together. The second purpose of the semantic inference is to generalize or specify already

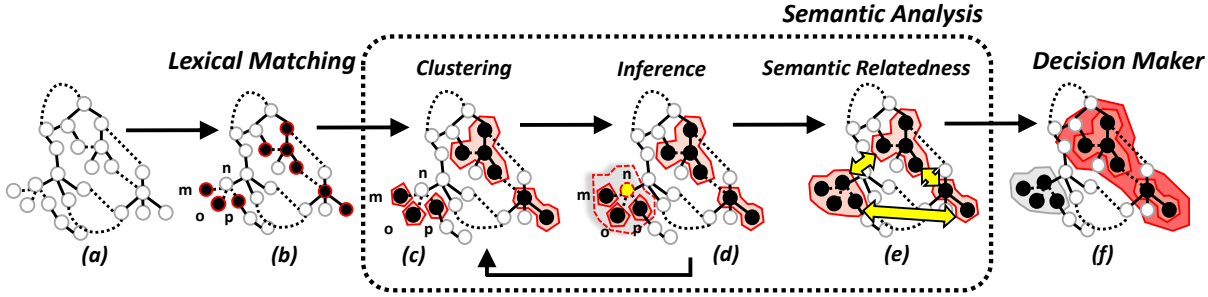

Fig. 4: Semantic Analysis: Step by Step.

"activated" concepts based on context information. For example, if the entity $\langle interface \rangle$ was activated for a certain level, and further IE identifies "exclusive" properties of a child concept (*e.g.*, $\langle ethernet - interface \rangle$), we infer that, the most specific concept is most likely to be the asserted entity.

Next, we compute the **semantic relatedness** as a means to determine the degree to which candidate resources are associated by meaning, and thus, select the concepts with higher relation. As mentioned earlier, due to the hierarchical nature of CLIs, we assume that maximum interrelated concepts are most likely representative of an executable sequence of commands. To exemplify this, let us consider the example shown in Fig. 5, where the resources *"MAC_Address"* in cluster $\mathcal{C}_C$ and *"IP_Address"* in cluster $\mathcal{C}_B$ are candidates for the CLI term "address". Observe that, both resources —and accordingly, the clusters to which they belong to— are disjoint, as only one ontological class can represent the semantics of the term. From a lexical perspective, the succinctness of the CLI is what generates ambiguity between both concepts. However, based on the contextual background, the concept *"IP_Address"* seems to be a better candidate, as it is semantically related to a higher degree to concepts identified in adjacent levels (*i.e.*, higher node density). Moreover, semantic relatedness contributes to the problem of word sense disambiguation, *i.e.*, picking the most suitable sense of the word and constraining the potential interpretation of terms in our system. In the next lines, we will explain our relatedness measurement $\mathcal{R}$.

Let $G(\mathcal{C}, R)$ be a directed graph, where the vertex $\mathcal{C}$ represents a cluster $\in G$, and the edge R represents a relationship among two adjacent clusters (cf., Fig. 6). Let $G_k \subseteq G$ represent a connected subgraph of G, and $\mathcal{C}_k^i$ be the i^{th} cluster $\in G_k$. As depicted in Fig. 6, the ontological class l within $\mathcal{C}_k^i$ shall be denoted as c_k^{il}. Equation (1) shows the relatedness measurement $\mathcal{R}$ that we developed, which consists of two components: the connection density $d(G_k)$, and the maximum information content coverage $\mathcal{I}(G_k)$.

$$\max_k \quad \mathcal{R}(G_k) = d(G_k) \cdot \mathcal{I}(G_k) \tag{1}$$

The density component $d(G_k)$ is shown in (2), and it is basically a measurement of the semantic connectivity of graph G_k. It is computed as the relation between the number of

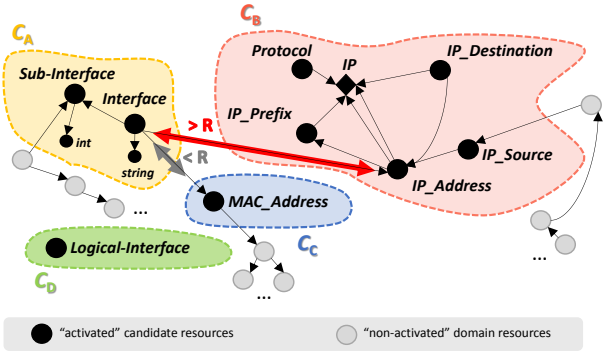

Fig. 5: The rationale behind the quantification of the Semantic Relatedness.

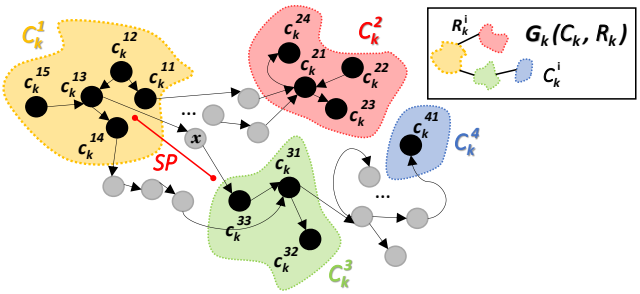

Fig. 6: An example of Semantic Relatedness.

"activated" entities along the shortest path between any pair of clusters in graph G_k, and the total number of connections (*i.e.*, the number of relations between entities) in those shortest paths.

$$d(G_k) = \frac{\sum_{i=1}^{|\mathcal{C}_k|-1} \sum_{j=i+1}^{|\mathcal{C}_k|} [\mathcal{A}(\mathcal{SP}(\mathcal{C}_k^i, \mathcal{C}_k^j)) - 1]}{\sum_{i=1}^{|\mathcal{C}_k|-1} \sum_{j=i+1}^{|\mathcal{C}_k|} \mathcal{H}(\mathcal{SP}(\mathcal{C}_k^i, \mathcal{C}_k^j))} \quad \leq 1 \tag{2}$$

More specifically, let $\mathcal{C}_k^i$ and $\mathcal{C}_k^j$ be a pair of clusters in G_k, and let $\mathcal{P}(c_k^{il}, c_k^{jp})$ denote a path between a pair of entities $c_k^{il} \in \mathcal{C}_k^i$, and $c_k^{jp} \in \mathcal{C}_k^j$. The shortest path between two clusters is defined as $\mathcal{SP}(\mathcal{C}_k^i, \mathcal{C}_k^j) = \min\ \mathcal{P}(c_k^{il}, c_k^{jp}),\ \forall\ c_k^{il}, c_k^{jp}$ in clusters $\mathcal{C}_k^i$, and $\mathcal{C}_k^j$, respectively. To illustrate this, consider the paths between the clusters $\mathcal{C}_k^1$ and $\mathcal{C}_k^3$ as shown in Fig. 6. In this case, the shortest path between any pair of entities (c_k^{1l}, c_k^{3p}), *i.e.*, paths with source in cluster $\mathcal{C}_k^1$ and termination in $\mathcal{C}_k^3$, or vice-versa, is $\mathcal{SP}(\mathcal{C}_k^1, \mathcal{C}_k^3) = [(c_k^{13}, x), (x, c_k^{33})]$. Now, let the function $\mathcal{A}(\mathcal{P})$ return the total number of "activated" entities (*i.e.*, the ontological classes) in path $\mathcal{P}$. In our example, $\mathcal{A}(\mathcal{SP}(\mathcal{C}_k^1, \mathcal{C}_k^3)) = 2$, which are c_k^{13} and c_k^{33}. Observe that the source of a path $\mathcal{P}$ is always an "activated" entity —recall that the clusters are composed of activated entities only— hence the number of "active connections" along a path $\mathcal{P}$ is $(\mathcal{A}(\mathcal{P}) - 1)$ (cf. (2)). Similarly, the function $\mathcal{H}(\mathcal{P})$ in the denominator of (2) returns the total number of hops in path $\mathcal{P}$. For instance, in the example shown in Fig. 6, $\mathcal{H}(\mathcal{SP}(\mathcal{C}_k^1, \mathcal{C}_k^3)) = 2$. Observe that when the clusters $\mathcal{C}_k^i$ and $\mathcal{C}_k^j$ are not adjacent, the shortest path can traverse other clusters. Hence, in a connected graph, the number of activated entities always satisfies $\mathcal{A}(\mathcal{SP}(\mathcal{C}_k^i, \mathcal{C}_k^j)) \geq 2$.

The second term of the relatedness measurement $\mathcal{R}$ is the Information Content $\mathcal{I}(G_k)$, which is shown in (3).

$$\mathcal{I}(G_k) = \sum_{i=1}^{|\mathcal{C}_k|} \sum_{l=1}^{|c_k^{il}|} t_k^{il} . m_k^{il} . o_k^{il} \tag{3}$$

This term represents a measurement of the knowledge covered by the ontological classes in their corresponding clusters. Let t_k^{il} denote the number of CLI terms that triggered the activation of an entity $c_k^{il} \in \mathcal{C}_k^i$ in the semantic graph G_k. Observe that in (3), the contribution of an entity c_k^{il} to the domain knowledge is weighted by two factors, m_k^{il}, and o_k^{il}. Let, m_k^{il}

be the *matching* factor of the l^{th} entity, which is 1 for entities identified by perfect match; otherwise, its value is chosen as $\frac{1}{(e+1)}$, with e the total number of entities identified for the same CLI term. In other words, in case of partial match, the weighting factor m_k^{il} represents the probability of being any of the e entities identified for the same CLI term —including none of them (+1).

In the example shown in Fig. 5, e = 2 for the entities triggered by the term "address", with equal probability from the information content perspective of being *"IP_Address"*, *"MAC_Address"*, or none. Moreover, the other weighting factor, *i.e.*, o_k^{il}, is a measurement of the "occurrence" of a candidate entity in the CLI, over the total number of occurrences of its exclusive disjoint entities. This measurement is computed during the IE process at the lexical matching stage.

Observe that we compute the semantic relatedness $\forall\ G_k \subseteq G$, that is, over the total number of connected cluster subgraphs of G. As indicated in (1), the relatedness measurement that we chose is the maximum obtained $\forall\ G_k$. It is worth mentioning that, even though at first sight our model might look a bit intricate, its computation is actually quite straightforward. The nature and hierarchical structure of CLIs typically yields a small number of interrelated clusters, and more importantly, as outlined in Fig. 1, this subsystem operates in offline mode, so the only and fundamental goal is the accuracy of the OBIE process. Indeed, the results that we present in Section IV confirm the strengths of our model and the approach proposed in this paper. Also observe that, although the ontology and some of the descriptions made in this Section are application-specific —*i.e.*, they consider particular features of CLI environments for routers —the essence of our model can be generalized and applied to other contexts, especially, those that rely on hierarchical CLIs for device configuration.

Finally, in the **Decision Maker** (cf., Fig. 4) we select the set of resources that best define the semantics of the CLI, as those belonging to the clusters with highest semantic relatedness. Furthermore, we combine these resources with the identified verb phrases and look for the most suitable atomic operations on a per-level basis. Afterwards, we instantiate commands and variables into the domain-ontology and chain them according to the hierarchical structure of the CLI. When the system has navigated over the complete CLI, the specific-ontology is stored in a repository according to the device model and OS version for further online queries from external (third-party) applications.

E. On-line Functionality

As stated earlier, the *online* functional mode supports the semantic retrieval of configuration statements. It semantically resolves configuration requests on the basis of heterogeneity and automatically retrieves the sequence of commands for a given device model. It provides a web-based interface for third-party applications willing to outsource the task of configuration adaptation to our system. Moreover, the online process supports advanced functions that enable format adaptation for well-known domain concepts. For example, it automatically performs subnet format adaptation. Notice that this knowledge is embedded in the domain ontology and adaptation functions are automatically triggered whenever an input differs from the expected type.

To illustrate the online functionality, consider the following use case. An NMS that targets programmability in multi-layer networks (*e.g.*, IP over Optical) is configured to automatically offload the traffic of a gold client over a new path whenever the traffic in the primary path goes beyond a certain threshold. In order to achieve this, current solutions rely on manually set configuration scripts —a strategy commonly used to simplify recurrent tasks— which depends on the underlying infrastructure. However, a solution of this type can actually outsource the configuration of the devices involved to our OBIE system, and request the required configuration in runtime —regardless of the underlying infrastructure— thus, decoupling network programmability from the specifics of the current network setting. Therefore, any change in the network (*e.g.*, new devices, operating system upgrades, etc.) will not affect regular ongoing processes. The online functionality has already been developed, tested and successfully validated in the framework of an European initiative, enabling management programmability in the context of multi-layer and multi-vendor networks [22].

IV. Evaluation

In order to evaluate the performance of our OBIE system, we have carried out experiments over the configuration spaces of two different network elements. One, a proprietary (*i.e.*, commercial) router and the other, an open-source routing software, namely, Juniper (Model M7i - JUNOS 10.4.R13.4) and Quagga (Release 0.99.21), respectively. We downsized the set of commands of the Juniper router to have a comparable set with respect to Quagga, including the most common set of features. Notice that, though the number of commands for a network device can be significantly large, the ones used in practice are a relative small set. For this reason, we thoroughly selected a set of 180 commands from each CLI, which encompass a broad set of functionalities —*i.e.*, not only protocol-specific configurations (*e.g.*, OSPF, SNMP) but device-related functions as well (*e.g.*, user account settings).

Overall, our system populates the domain ontology with instances of commands and variables according to the semantics derived from the CLI. In the evaluation, we measure the performance of both stages of our IE process (cf., Section III-D), namely, *resource* identification (Stage 1) and *operation* inference (Stage 2). The main motivation for reporting performance on both stages is to test the success of the proposed approach.

Regarding implementation, modules were developed in Java, as it suits the needs of integration with already available libraries for ontology management (OWL API [21]) and NLP (NLP Stanford [23]). Our system is evaluated using standard OBIE measurements, because traditional measurements are inadequate when using ontologies due to their binary behavior when determining the correctness of an instantiation. When making ontological classification, decisions are more obscure, *i.e.*, we can have "degrees" of correctness, *e.g.*, if the term

"OSPF" is classified as the concept $\langle Standard\_Protocol \rangle$ instead of $\langle Routing\_Protocol \rangle$, we are clearly less wrong than if classified as $\langle Application\_Protocol \rangle$. For this reason, we used the Balanced Distance Metric (BDM) [24] —a cost-based component that measures the degree of correctness according to the ontological structure.

The BDM scores were computed with the open-source BDM Computation Processing Resource, which is part of the General Architecture for Text Engineering (GATE) [25]. The BDM *per se* does not provide the means for evaluation. For this reason, we measured the Augmented Precision (*AP*), Recall (*AR*) and F-Measure (*AF*) as introduced in [24]. These metrics combine traditional Precision, Recall and F-Measure with the BDM. Overall, *AP* measures the correctness, *AR* the completeness and *AF* the overall quality of the instantiations —as the latter accounts for both metrics. In order to compute these metrics, a domain expert has manually built a gold standard which sets the most adequate semantic annotations from the domain ontology. We then compare extractions with the gold standard and compute augmented metrics.

The performance results are shown in Table I. Notice that we computed two sets of measurements, one for each stage of the IE process. The first and second rows show the performance metrics for the case of commercial and open-source routers, respectively. The third row shows the overall performance for both network elements. Notice that, the "overall" measures are not the mean values for the Juniper and Quagga experiments, but instead, we have merged both instantiation results and recomputed performance metrics. In general, our experimental results show that our system is capable of automatically extracting the semantics of the configuration environment from the CLI with high performance. Over 92% of configuration commands are adequately classified into their semantic categories.

Notice that the "miss-classification" of nearly 8% of the total of configuration commands is not always due to the inability of our system to take the right decisions. For instance, some of the evaluation results are affected by CLI inconsistencies, *e.g.*, when vendor's make assertions which are not strictly aligned to the domain knowledge. To illustrate this, let us consider the following example. A given CLI, arranges the MPLS commands under the "Routing Protocol" hierarchy. However, in the literature, MPLS is not actually considered a Routing Protocol. Furthermore, our system identifies the $\langle Routing\_Protocol \rangle$ concept in the early stage of lexical matching. However, further inference and clustering stages, generalize it to the $\langle Standard\_Protocol \rangle$ concept. The reason for this is that the $\langle MPLS \rangle$ concept is identified in subsequent levels and according to the ontology axioms it is not a "Routing Protocol". This means that, most likely $\langle MPLS \rangle$ and $\langle Routing\_Protocol \rangle$ cannot be at the same time candidates for a configuration statement. Given that the taxonomic ancestor of $\langle Routing\_Protocol \rangle$, (*i.e.*, $\langle Standard\_Protocol \rangle$) is semantically related to $\langle MPLS \rangle$ we do the generalization. Notice that this example is not actually a problem of miss-classification due to errors in our system, but rather a problem derived from an inconsistency between the literature and the vendor's interpretation of the domain's knowledge. Thus, we can conclude that the generalization feature of our system can certainly help to identify and bridge the gap between the CLI information and the domain knowledge model, whenever inconsistencies take place. These exceptions are out of the scope of this paper, and will be considered in our future work.

	Stage 1			***Stage 2***		
	AP	**AR**	**AF**	**AP**	**AR**	**AF**
Commercial (Juniper)	90%	89%	90%	92%	94%	93%
Open-Source (Quagga)	90%	83%	86%	92%	88%	90%
Overall	89%	85%	87%	92%	91%	92%

TABLE I: Performance Results of our OBIE Process.

V. Conclusions

In this paper, we have presented an Ontology-Based Information Extraction (OBIE) System from the Command-Line Interface (CLI) of network devices. Our solution aims to complement current OF-CONFIG and NETCONF protocols, whenever heterogeneity or data models hinder the remote configuration of network devices. This is particularly the case of hybrid SDN networks. To this end, we have developed an ontology for the switch/router configuration domain which is used to, *(i)* guide the IE process from the CLI, and *(ii)* enhance our assessment by exploiting the ontological structure. We have also developed an extraction method which relies on shallow NLP tools, lexical matching, clustering techniques, ontology reasoning and relatedness computation to perform the automatic instantiation of CLI commands and variables into their semantic categories. Based on the performance evaluation, we can conclude that the use of ontologies in conjunction with other semantic technologies for IE provides a promising line of research that can help mitigating the efforts of network device configuration. Moreover, our solution has the potential to enable autonomic networking, by assisting third-party applications in the execution of network device (re)configuration. To the best of our knowledge, our work is the first one using IE techniques from CLIs and ontologies enabling automated configuration of devices. We consider that this is an innovative solution in the field of network configuration management.

Our future work includes *(i)* enhancement of the learning algorithm based on historical instantiations, *i.e.*, perform semantic disambiguation taking into account previous decisions; *(ii)* performance evaluation over editable CLIs, *i.e.*, evaluate the performance of our system for augmented or customized CLIs, for instance, by including meta-information (improve verbosity); *(iii)* enhancement of the IE process from web-based resources; *(iv)* considering that there would be good suggestions already produced by our system holding for final verification, integration of a human validation stage in the final loop to improve the instantiation process to its maximum; and finally, *(v)* semantic-web integration in order to provide the ability to generate semantic contents of the switch/router configuration domain for the Web.

ACKNOWLEDGMENT

This work was supported in part by the Spanish Ministry of Science and Innovation under contract TEC2012-34682.

REFERENCES

[1] O. N. Foundation, "OpenFlow Switch Specification. Version 1.3.2," www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.2.pdf, 2013.

[2] ——, "OpenFlow Management and Configuration Protocol 1.2," www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf.

[3] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," IETF, RFC 6241, Jun. 2011, http://tools.ietf.org/html/rfc6241.

[4] A. Martinez, M. Yannuzzi, V. López, D. López, W. Ramírez, R. Serral-Gracia, X. Masip-Bruin, M. Maciejewski, and J. Altmann, "Network Management Challenges and Trends in Multi-Layer and Multi-Vendor Settings for Carrier-Grade Networks," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–1, 2014.

[5] C. Chappell, "The Business Case for NETCONF/YANG in Network Devices," Heavy Reading, White Paper, Oct. 2013.

[6] M. Bjorklund, "YANG - A Data Modeling Language for NETCONF," IETF, RFC 6020, Oct. 2010, http://www.ietf.org/rfc/rfc6020.txt.

[7] IETF, "NETCONF Data Modeling Language (NETMOD)," http://datatracker.ietf.org/wg/netmod/documents/.

[8] D. L. Goff, "SDN Challenges Discussed at Carrier Cloud Summit," 6WIND, http://www.6wind.com/blog/sdn-challenges-discussed-at-carrier-cloud-summit/.

[9] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and Research Challenges of Hybrid Software Defined Networks," *ACM Computer Communication Review*, vol. 44, no. 2, April 2014.

[10] D. Levin, M. Canini, S. Schmid, and A. Feldmann, "Incremental SDN Deployment in Enterprise Networks," in *SIGCOMM'13 Demo*, Hong Kong, China, August 2013.

[11] A. Pras, J. Schonwalder, M. Burgess, O. Festor, G. Perez, R. Stadler, and B. Stiller, "Key Research Challenges in Network Management," *Communications Magazine, IEEE*, vol. 45, no. 10, pp. 104–110, 2007.

[12] J. E. López de Vergara, V. Villagrá, and J. Berrocal, "Application of OWL-S to Define Management Interfaces Based on Web Services," in *Management of Multimedia Networks and Services*, ser. LNCS, J. Dalmau Royo and G. Hasegawa, Eds. Springer, 2005, vol. 3754, pp. 242–253.

[13] A. Wong, P. Ray, N. Parameswaran, and J. Strassner, "Ontology mapping for the interoperability problem in network management," *Selected Areas in Communications, IEEE*, vol. 23, no. 10, pp. 2058–2068, 2005.

[14] N. Lasierra, A. Alesanco, D. O'Sullivan, and J. Garca, "An autonomic ontology-based approach to manage information in home-based scenarios: From theory to practice," *Data and Knowledge Engineering*, vol. 87, no. 0, pp. 185 – 205, 2013.

[15] J. Keeney, S. van der Meer, and G. Hogan, "A recommender-system for telecommunications network management actions," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, May 2013, pp. 760–763.

[16] J. E. López de Vergara, V. Villagrá, and J. Berrocal, "Applying the web ontology language to management information definitions," *Communications Magazine, IEEE*, vol. 42, no. 7, pp. 68–74, 2004.

[17] H. Xu and D. Xiao, "Applying Semantic Web Services to Automate Network Management," in *Industrial Electronics and Applications, 2007. ICIEA 2007. 2nd IEEE Conference on*, 2007, pp. 461–466.

[18] J. López de Vergara, A. Guerrero, V. Villagrá, and J. Berrocal, "Ontology-Based Network Management: Study Cases and Lessons Learned," *Journal of Network and Systems Management*, vol. 17, no. 3, pp. 234–254, 2009.

[19] Z. Li and K. Ramani, "Ontology-based design information extraction and retrieval," *AI EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, vol. 21, pp. 137–154, 4 2007.

[20] "ONDC: Ontology for Network Device Configuration," http://www.netit.upc.edu/ondc, 2014.

[21] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL Ontologies," *Semant. web*, vol. 2, no. 1, pp. 11–21, Jan. 2011.

[22] "EU Project ONE," http://www.ict-one.eu, 2010-2013.

[23] "Stanford CoreNLP," http://nlp.stanford.edu/software/corenlp.shtml.

[24] D. Maynard, W. Peters, and Y. Li, "Evaluating evaluation metrics for ontology-based applications: Infinite reflection." in *LREC*, 2008.

[25] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters, *Text Processing with GATE (V.6)*, 2011.