Generative AI for Low-Level NETCONF Configuration in Network Management Based on YANG Models

Gergely Hollósi[‡], Dániel Ficzere[‡&], Pál Varga[‡]

[‡]*Dept. of Telecommunications and Artificial Intelligence*, *Faculty of Electrical Engineering and Informatics*
Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary
[&]*IIoT Division, AITIA International Inc.*, 48-50 Czetz J. Str. H-1039 Budapest, Hungary
corr.: hollosi.gergely@vik.bme.hu

***Abstract*—The NETCONF protocol, standardized by the IETF, is a cutting-edge solution for configuring network entities and offers an alternative to SNMP in modern network devices. Due to the complexity of configuration protocols and the challenges in creating valid configurations, generative AI solutions are promising for converting textual prompts into configuration descriptors. However, the potential of LLMs to generate NETCONF configurations has not been explored in the literature. This paper addresses this gap by evaluating the performance of five different LLMs – including Llama3, an open-source, on-premises capable model – in creating NETCONF configurations using the widespread YANG data models. In order to create valid network configurations using generative AI, this paper proposes a pipeline for integrating domain knowledge into LLMs without additional training and highlights common shortcomings and errors that prevent the generation of valid configurations. The findings indicate that the use of LLMs is promising for this task, but the current state-of-the-art is not yet mature enough for immediate industrial application in complex cases.**

***Index Terms*—network management, service management, NETCONF, YANG, SNMP, generative AI, LLM, RAG, XML, mib, gpt, llama3**

I. Introduction

State-of-the-art generative AI solutions – such as large language models (LLMs) – are widely applied in various topics with huge success [1], and extensive research is going on to utilize them in network and service management. The capabilities of Large Language Models (LLMs) are extensively explored across various aspects of network management, including fault analysis, performance management, provisioning, and configuration management. While Simple Network Management Protocol (SNMP) [2] is commonly used for network configuration and monitoring, new standards from the Internet Engineering Task Force (IETF), such as the NETCONF protocol [3], have also been adopted for network configuration. NETCONF uses XML for data encoding, and the YANG [4] data modeling language is currently the most common way to define data structures for NETCONF.

Although the NETCONF protocol is widely used, it is a relatively complex protocol utilizing a high number of so-called YANG models to be, human-friendly" (as is spoken by IETF) in describing various states of network devices and entities. To ease the creation of NETCONF configurations, it is tempting to apply artificial intelligence (AI) to create valid NETCONF configurations based on textual instructions or prompts. However, to our knowledge, there were no attempts to discover the capabilities of LLMs to generate low-level network configurations with NETCONF.

This paper aims to address this gap by providing a structured exploration of the problem. It investigates the capabilities of various widely available Large Language Models (LLMs) in generating NETCONF configurations across different network scenarios and validates their responses to present aggregated results. Additionally, the paper introduces a novel pipeline, similar to Retrieval Augmented Generation (RAG), to incorporate domain knowledge (such as YANG models) into LLMs without requiring training or embedding. The performance of both the LLMs and the pipeline is evaluated and presented in a consistent manner. We have crafted 12 scenarios and evaluated 7 LLM cases – including 5 language models as is and 2 extra cases with domain augmentation in the newly proposed pipeline. Altogether 3 independent runs were evaluated, resulting in 252 test cases.

Still, before we start – a generic question may arise: would machine-generated network configuration ever be as precise and good as human experts create it? Is it even worth going down the road prompting LLMs to create NETCONF? First, we can use generative AI as a support to give expert advice, examples, and templates, and we humans can fill the gaps towards the perfect solution. Second, we can prompt the LLMs to create those configuration descriptions, and we can evaluate the results, again correcting the answer to make it perfect (we are already gaining time, most probably). The feasibility of this second stage is what this paper evaluates with current LLMs by applying a specialized pipeline for augmenting domain knowledge.

As a third and ultimate prompting automatization solution, we can create test cases, even executable ones, using another (independent) LLM to test the solution and correct it until it becomes perfect. It is not the expert way to do it, but tedious,

978-3-903176-66-9

automatized testing with such a feedback loop can make it just as perfect as a human expert would come out with the solution.

The paper is structured as follows. Section II presents the related material to network management, highlighting the related protocols, standards and state-of-the-art AI-based solutions for network management. Section III describes the methodology applied in the paper, and describes the proposed pipeline to incorporate domain knowledge for low-level configurations. Section IV presents the result of the evaluation of different LLMs, while Section V concludes the paper. Section VI proposes some future research topics regarding the NETCONF configuration generation.

II. Related works

Network management encompasses a range of tasks and technologies dedicated for maintaining, monitoring, and optimizing computer networks. Advances in artificial intelligence, particularly in LLMs and RAG, have created new opportunities to enhance these tasks. LLMs can automate and improve various aspects of network management, such as fault prediction, configuration optimization, and security protocol enhancement through sophisticated data analysis and pattern recognition. RAG leverages the capabilities of LLMs and incorporates relevant data retrieval, further increasing the accuracy and effectiveness of network management solutions. This section introduces the main tasks of network management and highlights how NETCONF and YANG provide standardized approaches for network configuration and management across different vendors, addressing inconsistencies found in CLI and SNMP. Additionally, it briefly explores the applicability of LLMs, focusing on enhancing network management tasks.

A. Network Management

One of the most influential taxonomies of network management functions comes from the ITU-T recommendation series M.3000, called Telecommunications Management Network. ITU-T M.3400 – TMN management functions [5] – defines the essential tasks of fault management, configuration management, accounting management, performance management, and security management, later commonly referred to as FCAPS. These functions are applied to all layers of TMN, including the business, service, network, and element layers. Applying FCAPS to Network and Service Management is still a current practice, showing the resilience of TMN FCAPS principles.

Network management entities require information exchange from/to the managed nodes in order to support (or enforce) FCAPS functions. The next subsections provide a brief overview of widely used protocols and modeling languages for this purpose – such as SNMP, NETCONF and YANG.

1) Simple Network Management Protocol: SNMP is used to address network management tasks and to facilitate the management and monitoring of networked devices in a standardized and efficient manner. SNMP has been integral to network management for decades, offering real-time monitoring by collecting data on device performance, traffic statistics, and error rates, thus providing a comprehensive view of network health. SNMP also allows remote configuration of network devices, facilitating changes in settings, firmware updates, and operational modifications.

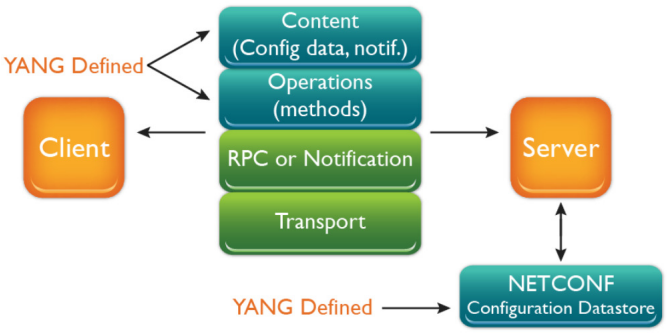

Fig. 1. The NETCONF protocol defines the NETCONF datastore and related operations through Remote Procedure Calls (RPC). The operations and the configuration contents are defined as YANG models, which define the structure of content data.

SNMP, despite its widespread use, has several notable disadvantages. It suffers from limited scalability, security issues (notably v1 and v2c versions) and high complexity. The lack of standardization in Management Information Bases (MIBs) across different vendors leads to inconsistencies and difficulties in multi-vendor network management. The protocol's request-response nature can result in slow response times under heavy load, and it provides limited contextual information about the data retrieved, making it challenging to understand the broader network state or correlate data effectively. These disadvantages have driven the development and adoption of more advanced network management protocols like NETCONF and YANG [6], [7].

2) NETCONF protocol: NETCONF addresses the need for a programmatic, cross-vendor interoperable interface to manage configuration state, providing a robust and scalable method to communicate configuration changes or complete configurations to devices, thereby streamlining the process compared to automating CLIs with scripts. The NETCONF protocol features a layered architecture, with a core RPC layer running over secure transports like SSH, and an operations layer that provides specific commands to manipulate configuration state (see Figure 1). Additionally, NETCONF supports advanced features like notification subscriptions and a modular design that promotes interoperability through capability exchanges between clients and servers [8], [3].

3) YANG: NETCONF uses XML to encode network management data, but existing XML schema languages like XSD and RelaxNG are insufficient for specifying all NETCONF-specific information. XSD, though formally used in NETCONF specifications, is difficult for humans to read and verify, while RelaxNG is somewhat easier but still needs to be more convenient when extended for NETCONF needs. The YANG data modeling language addresses these issues by offering a highly readable, compact domain-specific language

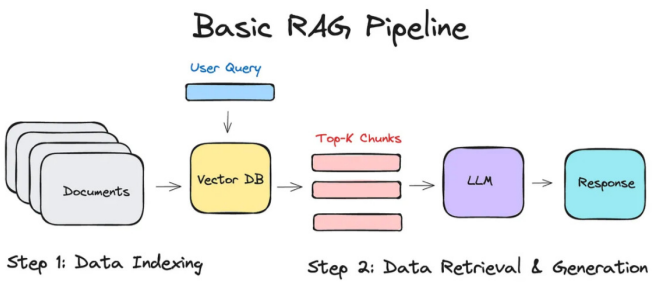

Fig. 2. The most basic Retrieval Augmented Generation pipeline [12]. The documents (and document chunks) are embedded and stored in a vector database, from where they are fetched based on semantic similarity. The prompt is then augmented with the fetched documents and passed to the LLM.

for defining NETCONF data models, along with YIN, an XML representation of YANG, enabling standard XML tool usage and conversions to XSD and RelaxNG [8], [4].

B. Applicability of LLMs

Language Models (LLMs) are sophisticated AI systems designed to comprehend and generate human language, marking a significant advancement in natural language processing. Their ability to understand context, generate coherent text, and perform various language tasks autonomously has revolutionized the field. LLMs find wide-ranging applications across domains such as natural language understanding, where they excel in sentiment analysis, named entity recognition, and language translation for customer service, content moderation, and automated translation services [9], [10], [11].

RAG plays a crucial role in enhancing the capabilities of LLMs by combining the strengths of both retrieval and generation mechanisms. In RAG, an LLM is augmented with a retrieval component that searches a large corpus of documents or data to find relevant information, which is then used to generate more accurate and contextually relevant responses (see Figure 2). This approach significantly improves the performance of LLMs in applications requiring precise and up-to-date information, such as question answering, knowledge management, and personalized content creation. By leveraging external knowledge bases, RAG addresses the limitations of LLMs related to knowledge cutoffs and memory constraints, ensuring that the generated content is both informed and contextually appropriate [13], [14].

LLMs can significantly enhance computer network management tasks through their proficiency in natural language understanding and processing. They interpret queries, commands, and reports related to network status, performance monitoring, and troubleshooting. LLMs automate monitoring processes, interpret alerts in natural language, and provide contextual insights for faster incident response. They also assist in managing network policies, configurations, and documentation by generating and updating information based on natural language instructions. Integrating LLMs into network management processes improves efficiency, decision-making, and overall operational effectiveness in maintaining and securing network infrastructure [15].

C. Network Managment using LLMs

LLMs hold significant potential in network management by automating and simplifying complex tasks. They can translate high-level policies and natural language descriptions into precise network configurations, reducing the likelihood of human error. LLMs can generate formal specifications, create API/function calls for SDN and automation protocols, and develop routing algorithms from high-level descriptions. Furthermore, they can produce low-level configurations for both existing and new protocols based on documentation, enhancing efficiency and interoperability in managing modern network environments [15]. Interoperability can also be enhanced by protocol message translation – a task that has been sought after many decades and now can be automated by LLM-supported protocol message translators [16]. Another application area is synthetic test data generation, through which network management practices – especially for fault and security – can be eased significantly [17].

Wang et al. investigate the use of LLMs to simplify network configuration and reduce errors by translating high-level policies and descriptions into low-level configurations and Python code [18]. They systematically explore the use of emerging LLMs to facilitate network device configuration and the development of routing algorithms from high-level requirements and natural language descriptions. They explore the potential applications of LLMs across four distinct network configuration tasks:

1) generating high-level requirements into formal specification formats [19], [20], [21];
2) translating high-level requirements into API/function calls for SDN and automation protocols [18];
3) writing code for routing algorithms based on high-level descriptions [22], [23];
4) generating low-level configurations for existing and new protocols based on input documentation [24], [25], [26].

In this paper, we are only dealing with the fourth task, generating low-level NETCONF configurations based on YANG models with different LLMs.

III. Methods

The main goal of our research is to determine whether state-of-the-art LLMs are capable of generating proper network device configurations based on precise instructions described by the network administrator. To this end, we do not deal with the understanding of the network functions and architecture, only the low-level configuration generation capabilities are investigated.

To evaluate the capabilities of the LLMs, basically, we created 12 instructions based on the dummy network presented in Figure 3. The instructions include different network management and administration topics of various complexity, ranging from setting IP address of an interface to configure TSN bridge gate states (see Table I). These instructions are

then handed over to different LLMs with the appropriate system prompt prefix: „ Write me a NETCONF configuration based on the instructions below:”.

TABLE I
CONFIGURATION SCENARIOS WITH THE PRECISE INSTRUCTIONS PROVIDED TO THE LLMS IN THE FOLLOWING FORMAT: ”WRITE ME A NETCONF CONFIGURATION BASED ON THE INSTRUCTIONS BELOW. INSTRUCTION: ...”

Scenarios	Instruction
1	Set the eth0 interface to 192.168.0.1/24 ip address.
2	Set the eth0 interface to 192.168.0.1 ip address with 255.255.255.0 netmask.
3	Set 172.16.0.2 as the default gateway.
4	Set the static route for the 192.168.2.0/24 network which can be reached via the 172.16.0.2 address.
5	Set up the RIP routing for the 192.168.1.0/24 and 192.168.2.0/24 networks. Use RIPv2 and no auto-summary.
6	Set the DNS server ip address to 8.8.8.8.
7	Set the NTP server address to pool.ntp.org.
8	Set up an Ethernet interface named "swp0", enable it and configure it as a bridge port with the following parameters: a queue maximum SDU size of 1024 bytes for traffic class 0, enable the gate, set the initial gate state to 0, set the base time (1700 seconds and 5000 nanoseconds), set the cycle time (11700/1000000000), and cycle time extension (0). Additionally, set gate control parameters as: Entry 0 sets the gate state to 1 with a time interval of 6200 and Entry 1 sets the gate state to 5 with a time interval of 5500.
9	Create a vlan with id 10 for the interface eth1, add the 10.0.0.1/24 to this vlan and set up the link.
10	Set up a destination NAT where every incoming tcp packets which arrives to the port 80 forwarded to the 192.168.0.2:80 address.
11	Set up a source NAT where every packets from the 192.168.0.0/24 domain translated to 172.16.0.1 address.
12	Instruction 1 + 3 + 4 + 6

Yet, NETCONF configurations are based on the YANG modeling language, enabling the configuration protocol to describe complex configuration states. Also, YANGs are dynamically loaded, making it possible to use different state models in any network management software. While publicly available LLMs are somewhat familiar with the well-known YANGs (e.g., included in various RFCs), generating configurations is more effective if the LLMs receive the YANG models on which the answer shall be based. It is clear that the high number of YANGs makes it impossible to hand over all the YANGs to the LLMs: on the one hand, the performance of the LLMs is degraded by providing too many unconcerned YANGs for a specific query, while on the other hand, the limited context window size poses hard lines to the length of the prompts. That is why another solution shall be investigated.

Commonly, domain knowledge can be included in LLMs through training or by using RAG pipelines. The latter can be applied to publicly available LLMs (e.g. ChatGPT) and requires significantly less resources. Our proposed solution for extending the knowledge of the LLM at hand with selected YANG models can be seen in Figure 4. RAG pipelines create embedding vectors for documents from domain knowledge and store the embedding vectors in a vector database for later lookup. However, embedding description languages created for computers is not straightforward, and the embedding

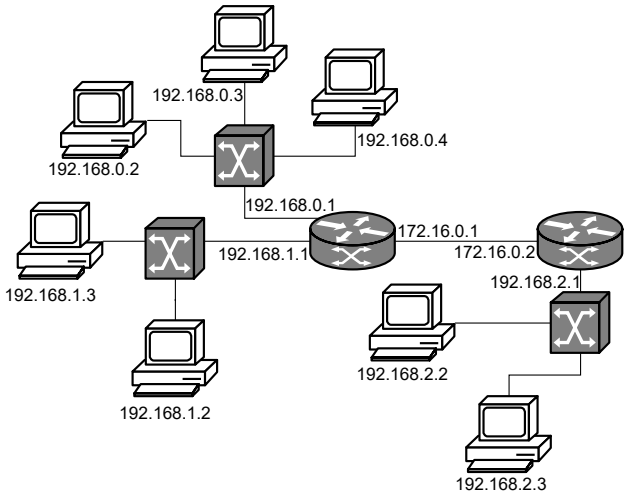

Fig. 3. Network topology to introduce some basic configuration instructions.

algorithms are optimized for readable texts. Similar to RAG pipelines, our solution utilizes a YANG lookup module, which accesses a YANG database (YANG DB). YANG lookups can be implemented in a couple of ways: e.g., for recurring problems, they can provide predefined YANGs, or the user might select a topic to which YANGs were previously assigned. Here, we propose a different method: from the YANG database, all the YANG name-description pairs are generated and fed into a prompt for the LLM. The LLM is then asked which YANGs are needed for the specific query. This enables the knowledge and intelligence of the LLMs to be incorporated into the pipeline while bypassing the problem of the embedding of YANG documents into vector databases. Generally, the limited number of available YANGs makes it possible for this method to be scalable.

After receiving the necessary YANG models from the YANG lookup, the original instruction prompt is augmented with the selected YANGs. For huge models (like GPT-4), multiple YANGs are passed, but for small LLMs, only the most relevant ones are kept. The augmented instructions are then passed to the LLM, which responds with the NETCONF configuration based on the YANGs in a well-defined XML format.

In the evaluation, five different LLMs are investigated, four publicly available huge LLMs and one locally deployed, open-source LLM:

- OpenAI’s GPT-3.5
- OpenAI’s GPT-4
- OpenAI’s GPT-4o
- Google’s Gemini
- Meta’s Llama3-7B model

The YANGs are collected from the appropriate RFCs and are publicly available on the internet. The five LLMs are evaluated using a zero-shot setting, where only the raw instruction is

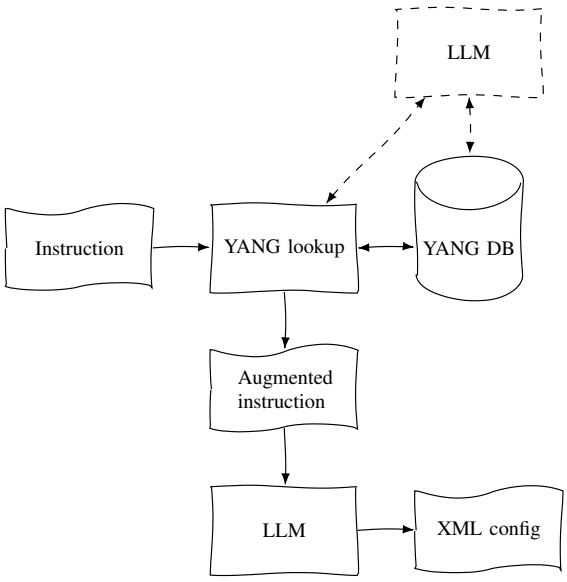

Fig. 4. The proposed pipeline for augmenting the original instructions with YANG models, neccessary for enabling the LLMs to answer in a strict and well-defined format. The pipeline basically povides domain knowledge to the otherwise general LLMs. Dashed line interactions and functions are optional, but there are instances where their use is appropriate, as detailed in the text.

passed to the LLMs. Also, GPT-4o and Llama3 are evaluated using the pipeline presented in Figure 4.

IV. Results

For the 12 scenarios and 7 LLM cases (5 language models and extra 2 with the proposed architecture), 3 independent runs were evaluated, resulting in 252 test cases. The raw zero-shot evaluations are represented by the names of the LLMs, and the proposed pipeline is evaluated for the GPT-4o (denoted as GPT4oY) and Llama3 (denoted as Llama3Y). The responses of the LLMs are preprocessed by `xmllint`, since it sometimes contains top-level `rpc` tags, or `config` XML tags. After preprocessing, the syntax and validation are done by the `yanglint` software bundled in the libyang[1] package. We used *config* type validation with the required YANGs provided.

However, raw YANG validation is not enough; we need to ensure that the response complies with the instructions. This is done by human validation; each test cases are evaluated by a human. Sometimes, the strict validation of the configuration fails, but the response is conceptually fine – for this reason, we present the results in three categories:

- VALID — The response is validated by the YANG models, and is conceptually right.
- MINOR ISSUES — The response validation failed despite being conceptually correct, mainly due to namespace violations or incorrect YANG models (see later).
- MAJOR ISSUES — The response is clearly incorrect, with missing or misused XML tags, an incorrect tree hierarchy, or syntax errors.

[1]https://github.com/CESNET/libyang

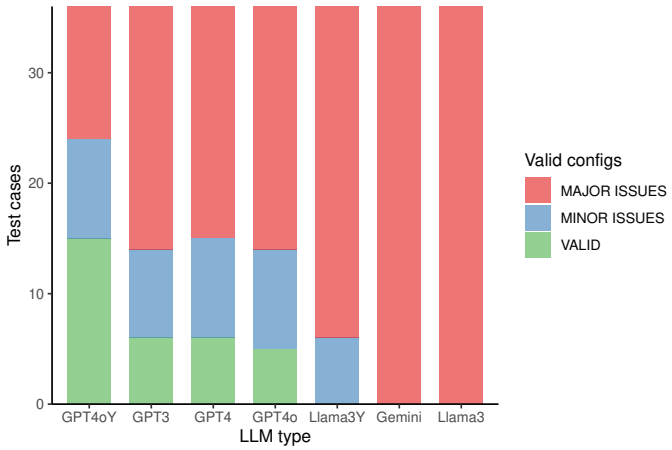

Fig. 5. The test case results for the examined LLMs. GPT4oY and Llama3Y are the cases where the previosly presented pipeline were used (Figure 4).

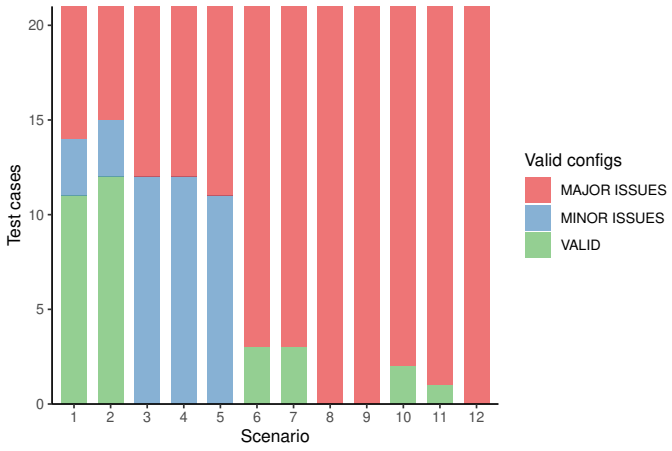

Fig. 6. The test case results for 12 scenarios.

The results grouped by LLMs are presented in Figure 5, while the results grouped by the scenarios are shown in Figure 6. Also, Table III contains the results as numerical values. The best-performing model is the flagship of the OpenAI, namely the GPT-4o model. However, using our pipeline to augment the instruction results in much higher performance (24 vs. 14 valid responses). Also, there are huge differences between scenarios. The LLMs are basically capable of generating fine results for simple tasks, but more complex ones (such as TSN and NAT) challenge them. It is interesting, that Gemini provides catastrophic results, and also raw Llama3 misses all the scenarios. Using the proposed pipeline for Llama3 helps the LLM to respond conceptually fine, but it still contains some syntax errors. GPT-3 and GPT-4 perform similarly, as do the raw GPT-4o.

For the YANG lookup module in the pipeline, each LLMs were asked which YANGs they would use for answering the instructions. Table II shows how many of the required YANGs were proposed by the specific LLM in each scenario. While most LLMs find the impor-

TABLE II
YANG MODEL LOOKUP RESULTS FOR THE LLMS IN THE DIFFERENT SCENARIOS.GREEN DOTS INDICATE THE CORRECTLY SUGGESTED YANG MODELS, WHEREAS RED DOTS DENOTE THE NECESSARY YANG MODELS THAT WERE MISSED.

Scenario	1	2	3	4	5	6	7	8	9	10	11	12
GPT3.5	●●●	●●●	●●	●●	●●●	●●	●●	●●●●●●●	●●●	●●●●	●●●●	●●●●●●
GPT4	●●●	●●●	●●	●●	●●●	●●	●●	●●●●●●●	●●●	●●●●	●●●●	●●●●●●
GPT4-o	●●●	●●●	●●●	●●●	●●●	●●	●●	●●●●●●●	●●●	●●●●	●●●●	●●●●●●
Gemini	●●●	●●●	●●	●●	●●●	●●	●●	●●●●●●●	●●●	●●●	●●●●	●●●●●●
Llama3	●●●	●●●	●●	●●	●●●	●●	●●	●●●●●●●	●●●	●●●●	●●●●	●●●●●●

tant YANG models, they miss some smaller YANGs, e.g. `iana-if-type`. Sometimes they hallucinate not existing models, like `ietf-dhcpv6-server`, or `common model`. The most underperformed scenario was again the TSN scenario, where only just the `ieee802-dot1q-bridge` was proposed. Also, for the NTP scenario, usually `ietf-ntp` was output; however, setting the NTP server requires `ietf-system`.

To gain some insight into the configuration errors, a couple of conclusions are presented. Minor issues in GPT models typically result in two different phenomena. First, XML namespace anomalies are present: the models often use default namespaces, and commonly, the default namespaces are defined one tag before or after they should be (e.g. `ietf-ipv4-unicast-routing`). The other problem is that they tend to use vendor-specific YANG, e.g., for routing, they default to the YANG `ietf-routing` defined by Cisco. This results in a conceptually fine routing configuration but is invalid in the light of RFC 8349. It is clear that complex YANGs (e.g., TSN) are challenges for the GPT models, and also, augmentation of containers in YANG models is not understood (e.g., VLAN interfaces). Also, Llama3 regularly fails at YANGs with the `choice` keyword, resulting in one additional node included in the XML.

A typical namespace anomaly can be found in the example below generated by Llama3 with the proposed pipeline. The `ietf-interfaces` namespace is defined but not used overall, resulting in an invalid yet conceptually fine XML:

```
<ietf-interfaces:interfaces
    xmlns:ietf-interfaces=
        "urn:ietf:params:xml:ns:
        yang:ietf-interfaces">
  <interface>
    <name>eth0</name>
    <description/>
    <type>ethernetCsmacd</type>
    <enabled>true</enabled>
    <ietf-ip:ipv4
        xmlns:ietf-ip="urn:ietf:params:xml:
            ns:yang:ietf-ip">
       <address>
          <ip>192.168.0.1</ip>
          <netmask>24</netmask>
       </address>
    </ietf-ip:ipv4>
  </interface>
</ietf-interfaces:interfaces>
```

Also, the Llama3 cannot understand the `choice` keyword in YANGs. This results in an unwanted `transport` tag generated, like in this example:

```
<system xmlns="urn:ietf:params:
    xml:ns:yang:ietf-system">
  <dns-resolver xmlns="urn:ietf:params:
    xml:ns:yang:ietf-system">
    <server>
      <name>dns-server</name>
      <transport>
        <udp-and-tcp>
          <address>8.8.8.8</address>
          <port>53</port>
        </udp-and-tcp>
      </transport>
    </server>
  </dns-resolver>
</system>
```

V. CONCLUSION

Different large language models were investigated to generate NETCONF configuration for network management and administration based on direct textual instructions. Beyond the five raw models, a domain knowledge-based pipeline was proposed to incorporate YANG models into language models. The conclusions are twofold. Apart from simple cases, the LLMs in their default forms are unable to generate valid NETCONF configurations based on the textual input.

The proposed pipeline makes a significant difference, especially in the case of the advanced GPT-4o model. However, complex configurations catch all the LLMs even with the knowledge of the YANG models. It is worth highlighting that there are a lot of conceptually right configurations which have minor errors. These errors have typically common roots: XML namespace problems and YANG model misconceptions. This follows from the fact that relatively few public sources can be found related to NETCONF and YANGs on the internet on which general LLMs were trained. All things considered, the seeds of understanding can be discovered in the general LLMs, but solving the problem of NETCONF configuration generation based on textual inputs requires specialized AI models.

VI. FUTURE RESEARCH

To overcome the issues presented in the paper, two main areas can be identified to fix. First, while the XML generation capabilities of LLMs are prominent, namespacing problems result in invalid XMLs. Second, LLMs have some shortcomings in understanding YANG models. Also, these two deficiencies might be connected to each other: namespacing

TABLE III
SUMMARIZING THE RESULTS FOR EVERY LLMS AND SCENARIOS. THE RESULTS IN THE TABLE ARE PRESENTED IN THE FOLLOWING FORMAT: VALID/MINOR ISSUES/MAJOR ISSUES.

Models	1	2	3	4	5	6	7	8	9	10	11	12	Model sum
GTP3.5	3/0/0	3/0/0	0/3/0	0/3/0	0/2/1	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	6/8/22
GPT4.0	3/0/0	3/0/0	0/3/0	0/3/0	0/3/0	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	6/9/21
GPT4.0-o	2/0/1	3/0/0	0/3/0	0/3/0	0/3/0	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	5/9/22
GPT4-oY	3/0/0	3/0/0	0/3/0	0/3/0	0/3/0	3/0/0	3/0/0	0/0/3	0/0/3	2/0/1	1/0/2	0/0/3	15/9/12
Gemini	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/36
LLama3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/36
LLama3Y	0/3/0	0/3/0	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/0/3	0/6/30
Scenario sum	11/3/7	12/3/6	0/12/9	0/12/9	0/11/10	3/0/18	3/0/18	0/0/21	0/0/21	2/0/19	1/0/20	0/0/21	32/41/179

problems may originate from the misunderstanding of YANG namespaces. Still, future research is needed on the YANG model's understanding of LLMs to get some insight into the YANG perception of LLMs. Since LLMs have limited access to YANG models and descriptions in public data sources (this can be the reason for Cisco vendor-based generation errors), most probably, further research requires training LLMs extensively on YANG models, especially to understand the specific YANG idioms. Furthermore, the automatic test-case generation and correction mechanisms will also be built, which should eventually lead to minimizing the faults in AI-generated NETCONF configurations.

ACKNOWLEDGMENT

The research leading to these results is funded by the EU KDT-JU organization under grant agreement 101112089, within the project AIMS5.0 and from the partners' national programs and funding authorities.

REFERENCES

[1] A. Koubaa, W. Boulila, L. Ghouti, A. Alzahem, and S. Latif. Exploring chatgpt capabilities and limitations: A survey. *IEEE Access*, 11:118698–118721, 2023.

[2] J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and applicability statements for internet-standard management framework. RFC 3410, December 2002.

[3] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder. Network Configuration Protocol (NETCONF). RFC 6241, June 2011.

[4] M. Björklund. The YANG 1.1 Data Modeling Language. RFC 7950, August 2016.

[5] International Telecommunication Union - ITU-T. M.3400 TMN management functions, 2000.

[6] H. Cui, B. Zhang, G. Li, X. Gao, and Y. Li. Contrast analysis of netconf modeling languages: Xml schema, relax ng and yang. In *2009 International Conference on Communication Software and Networks*, pages 322–326, 2009.

[7] A. Amlou, A. Abane, M. Merzouki, L. A. Oucheggou, Z. Maasaoui, and A. Battou. Automated network programmability using openconfig yang models and netconf protocol. In *2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, pages 1–5, 2023.

[8] J. Schönwälder, M. Björklund, and P. Shafer. Network configuration management using netconf and yang. *IEEE Communications Magazine*, 48(9):166–173, 2010.

[9] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*, 2023.

[10] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[11] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, et al. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints*, 2023.

[12] J. Bainiaksina. How I built a Simple Retrieval-Augmented Generation (RAG) Pipeline, February 2024.

[13] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[14] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[15] H. Zhou, C. Hu, Y. Yuan, Y. Cui, Y. Jin, C. Chen, H. Wu, D. Yuan, L. Jiang, D. Wu, et al. Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities. *arXiv preprint arXiv:2405.10825*, 2024.

[16] T. Tothfalusi, E. Varga, Z. Csiszar, and P. Varga. Ml-based translation methods for protocols and data formats. In *2023 19th International Conference on Network and Service Management (CNSM)*, pages 1–5. IEEE, 2023.

[17] T. Tothfalusi, Z. Csiszar, and P. Varga. Utilizing generative ai for test data generation-use-cases for iot and 5g core signaling. In *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, pages 1–6. IEEE, 2024.

[18] C. Wang, M. Scazzariello, A. Farshin, S. Ferlin, D. Kostić, and M. Chiesa. Netconfeval: Can llms facilitate network configuration? *Proceedings of the ACM on Networking*, 2(CoNEXT2):1–25, 2024.

[19] R. Birkner, D. Drachsler-Cohen, L. Vanbever, and M. Vechev. {Config2Spec}: Mining network specifications from network configurations. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 969–984, 2020.

[20] D. Donadel, F. Marchiori, L. Pajola, and M. Conti. Can llms understand computer networks? towards a virtual system administrator, 2024.

[21] S. Shan, Y. Huo, Y. Su, Y. Li, D. Li, and Z. Zheng. Face it yourselves: An llm-based two-stage strategy to localize configuration errors via logs. *arXiv preprint arXiv:2404.00640*, 2024.

[22] R. Mondal, A. Tang, R. Beckett, T. Millstein, and G. Varghese. What do llms need to synthesize correct router configurations? In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, pages 189–195, 2023.

[23] S. K. Mani, Y. Zhou, K. Hsieh, S. Segarra, T. Eberl, E. Azulai, I. Frizler, R. Chandra, and S. Kandula. Enhancing network management using code generated by large language models. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, pages 196–204, 2023.

[24] C. Wang, M. Scazzariello, A. Farshin, D. Kostic, and M. Chiesa. Making network configuration human friendly. *arXiv preprint arXiv:2309.06342*, 2023.

[25] E.-D. Jeong, H.-G. Kim, S. Nam, J.-H. Yoo, and J. W.-K. Hong. S-witch: Switch configuration assistant with llm and prompt engineering. In *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, pages 1–7, 2024.

[26] P. Sharma and V. Yegneswaran. Prosper: Extracting protocol specifications using large language models. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, pages 41–47, 2023.