관 인 생 략

출 원 번 호 통 지 서

출 원 일 자	2024.12.11
특 기 사 항	심사청구(유) 공개신청(무)
출 원 번 호	10-2024-0183556 (접수번호 1-1-2024-1373509-20) (DAS접근코드C05E)
출원인 명칭	포항공과대학교 산학협력단(2-2004-043336-1)
대리인 성명	특허법인(유한)아이시스(9-2016-100121-4)
발명자 성명	홍원기 정의동 남석현
발명의 명칭	네트워크 설정 자동화 장치 및 방법

특 허 청 장

<< 안내 >>

1. 귀하의 출원은 위와 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 이용하여 특허로 홈페이지(www.patent.go.kr)에서 확인하실 수 있습니다.
2. 출원에 따른 수수료는 접수일로부터 다음날까지 동봉된 납입영수증에 성명, 납부자번호 등을 기재하여 가까운 은행 또는 우체국에 납부하여야 합니다.
※ 납부자번호 : 0131(기관코드) + 접수번호
3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고객번호 정보변경(경정), 정정신고서]를 제출하여야 출원 이후의 각종 통지서를 정상적으로 받을 수 있습니다.
4. 기타 심사 절차(제도)에 관한 사항은 특허청 홈페이지를 참고하시거나 특허고객상담센터(☎ 1544-8080)에 문의하여 주시기 바랍니다.
※ 심사제도 안내 : https://www.kipo.go.kr-지식재산제도

【서지사항】

【서류명】 특허출원서

【출원구분】 특허출원

【출원인】

【명칭】 포항공과대학교 산학협력단

【특허고객번호】 2-2004-043336-1

【대리인】

【명칭】 특허법인(유한)아이시스

【대리인번호】 9-2016-100121-4

【지정된변리사】 남정길,김형상

【포괄위임등록번호】 2017-003840-7

【발명의 국문명칭】 네트워크 설정 자동화 장치 및 방법

【발명의 영문명칭】 METHOD AND DEVICE FOR SETTING NETWORK AUTOMATICALLY

【발명자】

【성명】 홍원기

【성명의 영문표기】 HONG, Won Ki

【국적】 KR

【주민등록번호】 590928-5XXXXXX

【우편번호】 37673

【주소】 경상북도 포항시 남구 청암로 77

【거주국】 KR

【발명자】

【성명】 정의동

【성명의 영문표기】 JEONG, Eui Dong

【국적】 KR

【주민등록번호】 970927-0XXXXXX

【우편번호】 37673

【주소】 경상북도 포항시 남구 청암로 77

【거주국】 KR

【발명자】

【성명】 남석현

【성명의 영문표기】 NAM, Suk Hyun

【국적】 KR

【주민등록번호】 961030-0XXXXXX

【우편번호】 37673

【주소】 경상북도 포항시 남구 청암로 77

【거주국】 KR

【출원언어】 국어

【심사청구】 청구

【공지예외적용대상증명서류의 내용】

【공개형태】 논문공개

【공개일자】 2024.06.10

【이 발명을 지원한 국가연구개발사업】

【과제고유번호】 2710007872

【과제번호】 00392332

【부처명】 과학기술정보통신부

【과제관리(전문)기관명】 정보통신기획평가원

【연구사업명】 차세대네트워크(6G)산업기술개발(R&D)

【연구과제명】 6G 네트워크 통합 지능평면 기술 개발

【과제수행기관명】 포항공과대학교 산학협력단

【연구기간】 2024.04.01 ~ 2026.12.31

【취지】 위와 같이 특허청장에게 제출합니다.

대리인 특허법인(유한)아이시스 (서명 또는 인)

【수수료】

【출원료】 0 면 46,000 원

【가산출원료】 32 면 0 원

【우선권주장료】 0 건 0 원

【심사청구료】 20 항 1,186,000 원

【합계】 1,232,000원

【감면사유】 전담조직(50%감면)[1]

【감면후 수수료】 616,000 원

【첨부서류】 1.공지예외적용대상(신규성상실의예외, 출원시의특례)규정을 적용받기 위한 증명서류_1통

1 : 공지예외적용대상(신규성상실의예외,_출원시의특례)규정을_적용받기_위한_증명서류

PDF 파일 첨부

S-Witch: Switch Configuration Assistant with LLM and Prompt Engineering

Publisher: IEEE

Eui-Dong Jeong ; Hee-Gon Kim ; Sukhyun Nam ; Jae-Hyoung Yoo ; James Won-Ki Hong **All Authors**

Abstract:
In modern network structures that become more complex and emphasize flexibility, the demand for the automation of network management and Intent Driven Network (IDN) conti... **View more**

Metadata

Abstract:
In modern network structures that become more complex and emphasize flexibility, the demand for the automation of network management and Intent Driven Network (IDN) continues to increase. In response, technology utilizing virtualization and control plane separation has developed, and research on network automation based on this is being actively conducted. However, limited studies focus on building an automated network in environments comprised of traditional switches that lack support for these advanced functionalities. Consequently, this study presents a technology proposal that creates the CLI command for existing commercial switches by incorporating user requests conveyed through natural language. For this purpose, we applied Large Language Model (LLM) for generating and Network Digital Twin for verification environment.

Published in: NOMS 2024-2024 IEEE Network Operations and Management Symposium

Date of Conference: 06-10 May 2024

DOI: 10.1109/NOMS59830.2024.10575007

Date Added to IEEE *Xplore*: 02 July 2024

Publisher: IEEE

ISBN Information:

Conference Location: Seoul, Korea, Republic of

ISSN Information:

Contents

Authors

Figures

References

Citations

Keywords

Metrics

S-Witch: Switch Configuration Assistant with LLM and Prompt Engineering

Eui-Dong Jeong, Hee-Gon Kim, Sukhyun Nam, Jae-Hyoung Yoo, and James Won-Ki Hong

Department of Computer Science and Engineering, POSTECH, Pohang, Korea
{justicedong, sinjint, obiwan96, jhyoo78, jwkhong}@postech.ac.kr

***Abstract*—In modern network structures that become more complex and emphasize flexibility, the demand for the automation of network management and Intent Driven Network (IDN) continues to increase. In response, technology utilizing virtualization and control plane separation has developed, and research on network automation based on this is being actively conducted. However, limited studies focus on building an automated network in environments comprised of traditional switches that lack support for these advanced functionalities. Consequently, this study presents a technology proposal that creates the CLI command for existing commercial switches by incorporating user requests conveyed through natural language. For this purpose, we applied Large Language Model (LLM) for generating and Network Digital Twin for verification environment.**

***Index Terms*—Network Configuration Automation, Intent Driven Network, Large Language Model, Prompt Engineering**

I. Introduction

In recent years, the field of networking has been characterized by escalating diversity and complexity. As networks evolve, the demand for systems that can respond dynamically to these changes and execute intricate configurations with minimal human intervention has become paramount. In response to this need, innovative paradigms such as Zero Touch Networks (ZTN) and Intent Driven Networks (IDNs) have emerged. These technologies have advanced on the backbone of foundational frameworks like Software Defined Networks (SDN) and Network Function Virtualization (NFV).

However, conventional equipment that has already been manufactured cannot support for the aforementioned modern network paradigms, and even now, equipment designed for hardware optimization is difficult to support this.

Addressing this limitation, this paper proposes a novel approach to bridge the transition towards IDN. We focus on automating the configuration of traditional networking equipment, thereby reducing the dependence on manual intervention. Specifically, we introduce a network configuration assistant capable of generating device-specific configuration CLI commands based on user intents expressed in natural language. The source code has been released at https://github.com/euidong/S-Witch.

II. Background

A. Intent Driven Network

The usage of networks has increased dramatically due to the growth of streaming services, real-time video services, and cloud gaming. The evolution of Internet of Things (IoT) and AI technologies has further increased the demand for specific capabilities like low-latency and high-capacity transmission. Trying to address these demands manually would markedly burden both capital and operational expenditure (CAPEX/OPEX). In response to these challenges, the IDN emerged to build a network that can automate such management according to user needs. IDN represents an evolved network that is capable of self-optimizing based on the intentions set forth by network operators. It autonomously manages network transmission, verification, deployment, and configuration tasks, thereby ensuring the system's ongoing operation and stability [1]. Notably, 3GPP underscores that the 'intent' provided to an IDN specifies the 'what' rather than the 'how' [2] (Fig. 1). This means that even individuals without detailed knowledge of network management should be able to achieve their objectives merely by stating their desired outcome.

Consequently, our research endeavors to empower users to freely configure their devices by merely inputting their requirements, circumventing the necessity to understand the intricacies of traditional network equipment configuration.

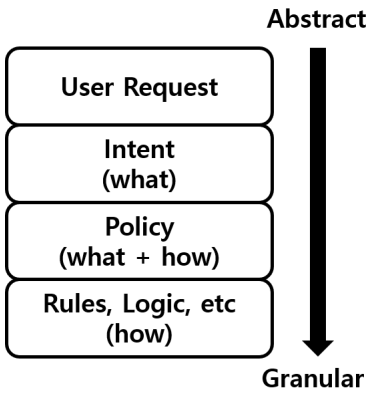


Fig. 1. Relation between rule, policy and intent

B. Large Language Model and Prompt Engineering

In the field of Natural Language Processing (NLP), the Seq2Seq model [3] was introduced to address the task of Machine Translation [4], which involves converting text from

one language to another. Subsequently, the Transformer [5] architecture, which enhances the Seq2Seq model by integrating an attention mechanism, has demonstrated remarkable effectiveness not only in Machine Translation but also in tasks involving prediction and generation. Moreover, the advent of Large Language Models (LLMs), which are developed by expanding the depth and breadth of layers in models based on the Transformer architecture, has led to even higher performance. These LLMs, built upon extensive language data sourced from the internet and literature, are adept at understanding various contexts. Consequently, the focus of Deep Learning research has shifted from seeking performance enhancement through the introduction of new, task-specific models and datasets to leveraging LLMs by providing them with appropriate textual prompts [6].

Notably, specific data tailored for the task of translating natural language input into CLI commands is extremely limited. This limitation arises not only from the absence of a supervised dataset that directly correlates specific requirements with corresponding CLI commands but also from the scarcity of datasets comprising high-quality CLI commands. Hence, traditional Deep Learning models, which typically rely on extensive and task-specific datasets, are not the most suitable approach for this challenge. Instead, this study proposes to address the issue by employing various prompting techniques, such as Few-Shot Learning, with pre-trained LLMs.

C. Network Digital Twin

A Digital Twin is a sophisticated, dynamically evolving system designed to monitor, manage, and enhance physical entities throughout their lifecycle. These entities can encompass a wide array, from machines and humans to human-centric objects and even intricate smart cities [7]. Specifically, a Network Digital Twin (NDT) is a specialized form of a Digital Twin, tailored for network infrastructures. It empowers network designers to fine-tune various components and configurations virtually, without incurring physical costs, by creating a digital replica of the actual network. This virtual model facilitates numerous experimental scenarios, such as crafting an efficient network topology, devising traffic management strategies, and proactively predicting and enhancing network performance.

In our research, we recognized that the data produced by the LLM may sometimes be flawed, a phenomenon known as 'hallucination'. So, if there is a process that can directly execute the generated data in the NDT and obtain feedback, it can serve as a good foundation for future research. Therefore, we intended to design and implement this process using GNS3 [8] as small example of NDT.

III. Related Work

Several attempts have been made to construct an IDN using NLP or LLM, and apply them in the form of an assistant. Initially, Jacobs et al. [9] proposed a structure that extracts intent from natural language inputs using NLP, maps it to predefined intents, and then executes policies corresponding to each intent. This structure introduced a novel approach by enabling network configuration through natural language. However, it had limitations in terms of supported functionalities and was only targeted at OpenFlow [10] switches. After that, Jacobs et al. [11] developed an assistant named Lumi for academic environments, utilizing a Language Model. This assistant identified network configuration functions based on user requirements and presented a methodology for executing these functions. The process involved defining the types of intents through user customization and matching these intents with the functions to be executed, necessitating clear definitions and precise initial design. However, creating such detailed designs for every network is challenging and does not fit with current requirements that emphasize network flexibility.

Subsequently, Lin et al. [12] and Dzeparoska et al. [13] utilized LLMs and Few-Shot Learning to map natural language input to specific intents and predefined functions. Their research accelerated automated network construction by enabling operations like the creation, retrieval, and deletion of Virtual Network Functions (VNFs). Nevertheless, these studies were limited to handling switches created by existing vendors and relied on a VNF-dependent framework. Furthermore, the metrics presented in the validation process only assessed the accuracy of mapping natural language to predefined intents.

While previous research focused on mapping intents to predefined functions, policies, or rules, this study aims to generate configuration commands for traditional switch devices. And, we also differentiate ourselves from previous research by proposing a process for applying these commands to NDT to obtain feedback.

IV. S-Witch: Switch Witch

S-Witch is an assistant designed to facilitate network configuration based on CLI commands for devices such as L2 switches and L3 switches (Routers) that constitute a network. We aim to automate this process using LLMs. To achieve this, we leveraged the open-source library LangChain [14] to design and implement an application based on LLMs, developed an API Server using FastAPI [15], and utilized the network emulator GNS3 [8] as an NDT.

A. Overall Architecture

Fig. 2 depicts the overall structure of the architecture. Our system is comprised of three modules:

1) S-Witch Module: This module receives user inputs related to network configuration and outputs the actual configuration commands. It constructs this functionality by connecting LLMs through step-by-step prompting. Additionally, an API Server was established to enable other systems to utilize this module. The S-Witch module was built using LangChain and FastAPI.
2) Digital Twin Service: This module evaluates whether the commands generated by the assistant can be executed successfully. GNS3 was utilized to construct this environment.

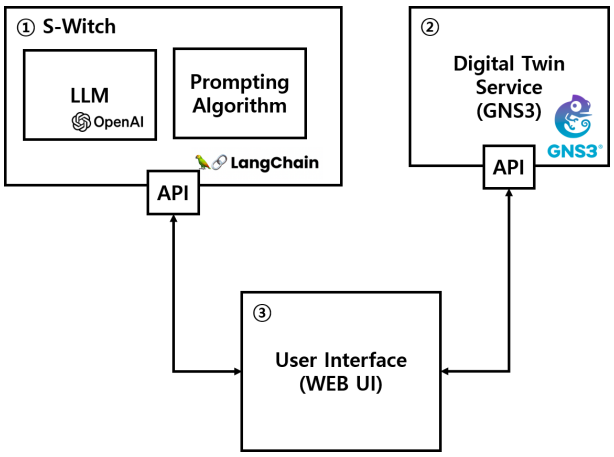


Fig. 2. Overall Architecture

3) User Interface: This is where the user inputs data. In this environment, users can communicate with the assistant through chatting and can review the results applied to the NDT. The implementation extended the WEB UI provided by GNS3 for this purpose.

Subsequent sections describe the specific internal implementation methods and finer points of the modules. The forthcoming content will elaborate on the architecture of S-Witch module, Prompt Engineering methods, NDT, and WEB UI.

B. S-Witch Module Architecture

We have implemented an assistant that chains LLMs across multiple stages to meet requirements. For this end, we employed LangChain which allows the bundling of LLMs, prompting methods, and additional tools into a single chain, providing monitoring capabilities. Using this, we constructed a structure as shown in Fig. 3.

We process three types of input—(1) previous chat history, (2) current question, and (3) network topology information—to fulfill the requirements in the form of CLI commands. This processing occurs over four stages:

1) Summarizing Chat History: Directly providing requirements to the LLM can hinder optimal performance. Hence, the most common method to convey contextual information is to pass along previous discourse content, assisting the LLM in its inferential process. While including more content can enhance the LLM's performance, the general limitation of input token size due to the Transformer architecture must be considered. With extensive chat history, summarizing becomes essential as it's not feasible to include everything.
2) Designing Blueprint: Directly outputting CLI commands for topology settings based on user requirements is typical, but there are limitations to this approach. The challenges are twofold: (1) performance issues, as requesting configurations for all devices at once may lead to omissions and inconsistencies. And (2) the limitation on the number of output tokens due to the Transformer architecture, making it difficult to address large topologies. To tackle these, the first step is to ask questions about the entire network configuration settings that need to be performed to meet the requirements, ensuring consistency across the network.
3) Allocating IP/subnet mask: Ideally, each device's port would have an IP and subnet mask assigned by the user, but more often than not, this doesn't happen. Hence, it's necessary to allocate them. Using the blueprint and network topology information from the previous steps, this process assigns suitable IPs, subnet masks, etc., to each device's port.
4) Generating Each Device CLI Command: As mentioned, it's impossible to handle CLI commands for all devices in one go. Therefore, each device's settings are requested in parallel from the LLM, and the results are compiled to extract the necessary CLI commands for each device.

C. Prompt Engineering

In designing the LLM Chain, the following prompting strategies were applied to achieve better performance:

- Assigning a Persona: To clarify the context as not just a chat system but specifically for network operation, a network operator persona was assigned.

> You are a network operator, and I will request network configuration from you through chat. In the request process, I will convey the network topology, the content of the conversation so far, and my requirements.

- Explaining the 4-Stage Execution Procedure: Each stage's LLM is unaware of the overall operation, so information about the entire 4-stage process is provided, and the current stage is clearly indicated. This helps improve inference performance by providing context information.

> – In the first stage, the previous conversation is received as input and goes through a process of summarizing it. If there is no previous conversation content, the step is skipped.
> – In the second stage, the overall network design method to achieve the requirements is determined. For example, determine routing protocol, etc.
> – In step 3, the task of assigning an IP address and subnet mask to each port of the topology is performed according to the overall design.
> – In step 4, the task of creating a CLI command for each device is performed.

- Providing Few-Shot Examples: A few examples of the expected output form are provided to ensure more accurate completions (see Listing 1).

```
- {"device": "PC1", "command": "ip
  192.168.1.2 /24 192.168.1.1", "comment":
  "configure PC1 ip address and default
  gateway address"}
```

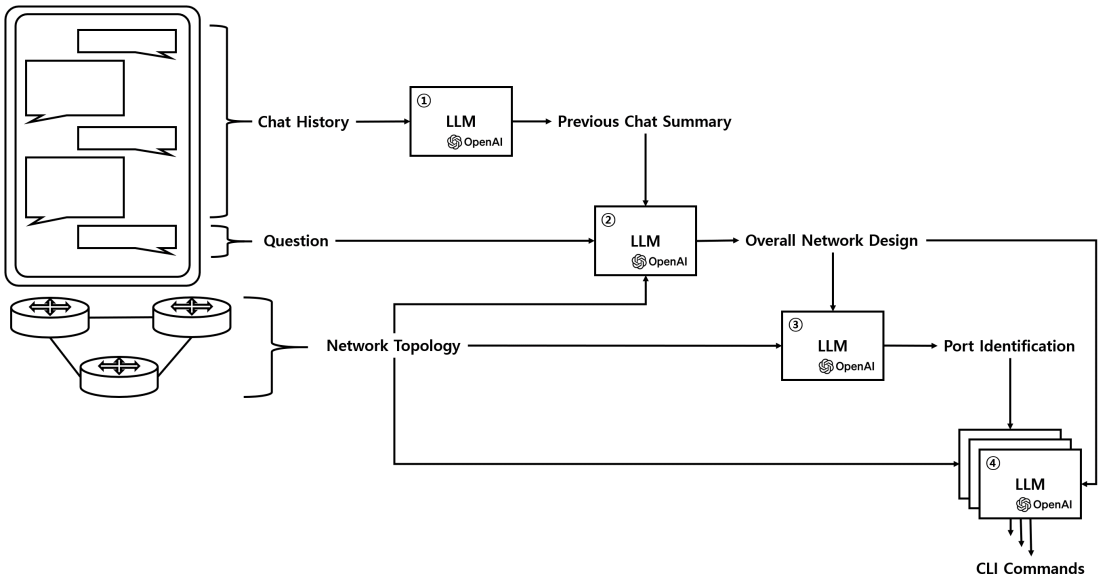


Fig. 3. S-Witch Module Architecture

```
- {"device": "PC2", "command": "ip
    192.168.2.2 /24 192.168.2.1", "comment":
    "configure PC2 ip address and default
    gateway address"}
- {"device": "R3", "command": "", "comment":
    "No command is required."}
...
```

Listing 1. Few-shot Example

D. Network Digital Twin

Even with the aforementioned steps for output generation, LLMs may not always produce accurate outputs. Hence, it's crucial to have a virtual environment where these outputs can be practically tested. In this implementation, GNS3 is utilized as the NDT. GNS3 allows the construction of topology, emulation of switches using virtual OS, and the execution of tasks to build the topology. Moreover, it enables remote command execution for each device, allowing direct execution of generated CLI Commands. GNS3 also supports the Snapshot feature, enabling users to take snapshots before and after applying each command, facilitating easy testing of the virtual environment. All these functionalities are accessible through the GNS3 API.

E. WEB UI

While GNS3 provides a desktop application interface, it also offers a WEB UI. This WEB UI allows users to perform tasks such as adding new devices to construct a topology and to view and set various device information. To this existing functionality, we added a chatting feature and integrated it with the S-Witch assistant. Through this integration, network topology information can be extracted in a JSON format, as shown in Fig. 4. When the chatting history and requirements are included in this information, we can acquire the necessary CLI commands for each device.

Listing 2 is an illustration of how network topology (Fig. 5) is actually represented when transmitted.

V. Results

Before presenting the results, it's important to describe the setup used. We utilized the gpt-3.5-turbo-1106 model from OpenAI's LLM models, capable of handling an input of 16,385 tokens in its context window. Additionally, a basic network topology was developed using GNS3, as illustrated in Fig.6.

Initially, we established the following requirement: "I want to be able to send and receive pings between all nodes on the network topology." The outcomes achieved through the S-Witch system's resolution of this are presented in Listing 3 and 4. Observing the router configuration (Listing 3), one can see the assignment of suitable IPs and the implementation of the OSPF routing protocol. In the PC configuration section (Listing 4), PC5 is notably well-set with appropriate IP, subnet

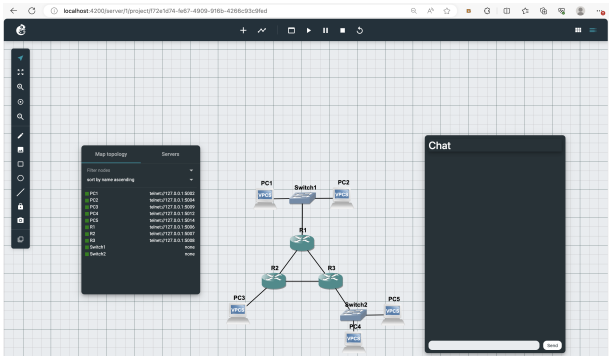


Fig. 4. WEB UI

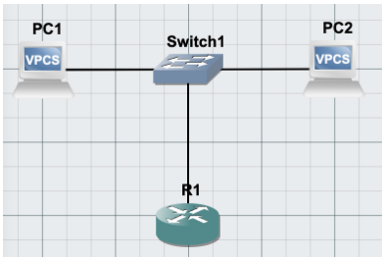


Fig. 5. Simple Network Topology

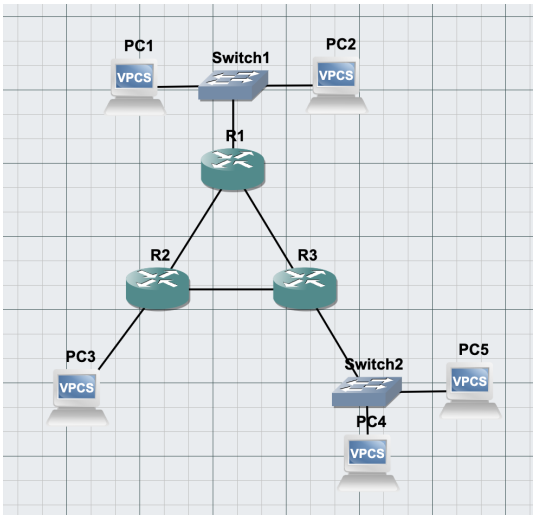


Fig. 6. Network topology, consisting of three routers and several PCs

```
"topology": {
   "link_info": [
      {
         "link_id":"link-id-1",
         "link_type":"ethernet",
         "nodes":[
            {
               "node_id":"node-id-1",
               "port_number":0
            },
            {
               "node_id":"node-id-2",
               "port_number":0
            }
         ]
      },
      ...
   ],
   "node_info":[
      {
         "node_id":"node-id-1",
         "name":"PC1",
         "node_type":"vpcs",
         "ports":[
            {
               "name":"Ethernet0",
               "port_number":0,
               "link_type":"ethernet"
            }
         ]
      },
      {
         "node_id":"node-id-4",
         "name":"R1",
         "node_type":"dynamips",
         "ports":[
            {
               "name":"FastEthernet0/0",
               "port_number":0,
               "link_type":"ethernet"
            }
         ]
      },
      ...
   ]
}
```

Listing 2. JSON formatted Network Topology

mask, and default gateway. However, for PC1, not only is the IP incorrectly assigned, but the gateway is also not configured. This problem occurs because an incorrect IP is assigned in step 3, IP allocation. Although it was emphasized that each device's connection and subnet conflicts must be taken into account when allocating IPs in step 3, allocation based on these was not done properly. This is because LLM itself has limitations in processing mathematical operations. To solve this, it is generally common to apply a separate tool. In fact, when performing a mathematical calculation in OpenAI's ChatGPT [16], instead of leaving it directly to the LLM, the method was adopted by asking the LLM to write a code that can calculate it and then executing it. Therefore, further research is needed on IP allocation, which will be discussed in detail in the Discussion Section.

```
Name: R3
CMD: enable
configure terminal
interface GigabitEthernet0/0
ip address 192.168.3.2 255.255.255.0
no shutdown
interface GigabitEthernet1/0
ip address 192.168.5.2 255.255.255.0
no shutdown
interface GigabitEthernet2/0
ip address 192.168.7.1 255.255.255.0
no shutdown
exit
router ospf 1
network 192.168.3.0 0.0.0.255 area 0
network 192.168.5.0 0.0.0.255 area 0
network 192.168.7.0 0.0.0.255 area 0
exit
exit
copy running-config startup-config
exit
Desc: Set the IP address of Router3 on the
    interfaces GigabitEthernet0/0, 1/0, and
    2/0 and configure OSPF routing.
```

Listing 3. CLI command for Routers, we requested a command to set up each device (CMD) and obtained an explanation (Desc).

```
Name: PC1
CMD: ip 192.168.1.1 255.255.255.0
Desc: Assign IP address and subnet mask to PC1

...

Name: PC5
CMD: ip 192.168.1.5 255.255.255.0 192.168.1.1
Desc: Configure PC5 IP address and default
    gateway address
```

Listing 4. CLI command for PCs, we requested a command to set up each device (CMD) and obtained an explanation (Desc)

Next, we requested the blocking of traffic to PC3 through R1, resulting in the output depicted in Listing 5. Prior to execution, we anticipated the use of an Access List for blocking, and indeed, we could confirm that the Access List was appropriately configured for PC5 (192.168.1.3).

```
Name: R1
CMD: ...
access-list 101 deny ip any host 192.168.1.3
access-list 101 permit ip any any
...
```

Listing 5. CLI command for Blocking

Lastly, we also experimented with the application of VLANs. We issued a request to "separate the broadcast domains of PC4 and PC5," resulting in an output as shown in Listing 6. We confirmed the successful generation of commands for setting up VLANs on Switch2. However, despite the presence of a single link leading to R3, commands for trunk mode were not generated unless explicitly specified in the requirements.

```
Name: Switch2
CMD: ...
interface Ethernet1
switchport mode access
switchport access vlan 10
exit
interface Ethernet2
switchport mode access
switchport access vlan 1
exit
...
```

Listing 6. CLI command for VLAN

VI. Discussion

In the process of generating CLI commands, our system demonstrated success in certain areas while encountering challenges in others. This necessitates a discussion on potential improvements.

Firstly, the domain of IP allocation exhibited some issues. As highlighted in the Results section, IP allocation necessitates an understanding of network topology and involves intricate operations such as subnet collision and range calculations. To enhance performance in this area, we suggest the formulation and implementation of specific rules, as opposed to relying solely on the LLM. Alternatively, adopting an approach similar to that used in ChatGPT, which involves coding, might yield better outcomes.

The second issue is the inability to consider details. In the Results section, we could not solve the problem in VLAN configuration without detailed description of trunk mode. In addition, it was difficult to show good performance for settings depending on the switch version, the latest technologies such as netconf or segment routing, or complex requirements. Therefore, we should consider applying Retrieval Augmented Generation (RAG) [17] to include the latest technical documentation. This is a method of delivering reference materials to the LLM so that they can be used as context information. Here, the configuration manual provided by the company is provided as reference material, and if there are QnA or additional materials, these can also be used. We actually applied this and conducted several experiments, but it did not show good performance in the process of finding documents according to questions. However, if additional research on this is conducted and successfully performed, it will be possible to build a system that can perform more specific settings.

Lastly, enhancing the system's overall performance is a consideration. The ReAct mechanism [18] suggests that performance can be improved by applying reinforcement learning techniques and utilizing real-world feedback from the outputs of the LLM. Having confirmed the feasibility of execution through NDT, if we can execute this process based on the ReAct mechanism, it holds the potential to elevate the system's performance to a higher level.

VII. Conclusion

We used LLM to create an assistant that converts network user requirements entered in natural language into CLI commands that can be used in actual network switches. In addition, NDT was linked to an environment where it could be executed and tested. We expect that the research we conducted will not only demonstrate the possibility of automating switch configuration using LLM, but will also serve as a foundation that can serve as a baseline for future research. Lastly, as discussed in the Discussion section, we have reached the level of creating commands and executing them on Digital Twin, but there are still limitations to the commands generated. Therefore, in future research, we plan to introduce new methods such as RAG, ReAct, etc. presented above to overcome this.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (2018-0-00749, Development of Virtual Network Management Technology based on Artificial Intelligence)

REFERENCES

[1] L. Pang, C. Yang, D. Chen, Y. Song, and M. Guizani, "A survey on intent-driven networks," *IEEE Access*, vol. 8, pp. 22 862–22 873, 2020.

[2] 3GPP TS 28.312, V17.5.0, 2023-09, "Intent driven management services for mobile networks".

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[4] H. Wang, H. Wu, Z. He, L. Huang, and K. W. Church, "Progress in machine translation," *Engineering*, vol. 18, pp. 143–153, 2022.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[6] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[7] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[8] "Gns3." [Online]. Available: https://www.gns3.com

[9] A. S. Jacobs, R. J. Pfitscher, R. A. Ferreira, and L. Z. Granville, "Refining network intents for self-driving networks," in *Proceedings of the Afternoon Workshop on Self-Driving Networks*, 2018, pp. 15–21.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[11] A. S. Jacobs, R. J. Pfitscher, R. H. Ribeiro, R. A. Ferreira, L. Z. Granville, W. Willinger, and S. G. Rao, "Hey, lumi! using natural language for {intent-based} network management," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021, pp. 625–639.

[12] J. Lin, K. Dzeparoska, A. Tizghadam, and A. Leon-Garcia, "Apple-seed: Intent-based multi-domain infrastructure management via few-shot learning," in *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*. IEEE, 2023, pp. 539–544.

[13] K. Dzeparoska, J. Lin, A. Tizghadam, and A. Leon-Garcia, "Llm-based policy generation for intent-based management of applications," in *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023, pp. 1–7.

[14] "Langchain." [Online]. Available: https://www.langchain.com

[15] "Fastapi." [Online]. Available: https://fastapi.tiangolo.com

[16] "Chatgpt." [Online]. Available: https://chat.openai.com

[17] 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*, vol. 33, pp. 9459–9474, 2020.

[18] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

【발명의 설명】

【발명의 명칭】

네트워크 설정 자동화 장치 및 방법{METHOD AND DEVICE FOR SETTING NETWORK AUTOMATICALLY}

【기술분야】

【0001】 본 개시는 네트워크 내 장비들의 설정을 자동화하는 장치 및 방법에 관한 것이다.

【발명의 배경이 되는 기술】

【0002】 대규모 언어 모델(Large Language Model, LLM)은 방대한 양의 텍스트 데이터를 학습하여 이해 및 생성작업을 수행할 수 있는 인공지능 모델이다. 대표적인 예로 GPT-4와 Meta의 LLaMA 등이 존재한다. LLM은 수십 억에서 수천억 단위의 파라미터를 사용하는 Neural Network 모델이다. 방대한 양의 파라미터와 인터넷의 존재하는 수많은 데이터를 통합하여, LLM은 기존의 AI가 수행할 수 없었던 문장의 문맥을 이해하고 광범위한 범위에서 효과적으로 응답을 생성할 수 있다. 이 과정에서 일반적으로 LLM은 Transformer 구조를 기반으로 설계되며, 자연어 처리 분야에서 높은 성능을 보여주고 있다. LLM은 방대한 크기의 범용 데이터셋으로 사전 학습을 진행하여 일반적인 언어 패턴과 지식을 활용할 수 있다.

【0003】 종래 연구들에서도 LLM 또는 AI를 활용한 네트워크 설정을 수행하고자 하는 시도가 존재하였다. 하지만, 종래 기술들은 다음과 같은 한계점을 가진다.

종래 기술의 경우, OpenFlow Switch에 집중하여 네트워크 설정 자동화를 구현하고자 하였다. 하지만, 이 과정에서 사용된 모델은 기초적인 단계의 Natural Language Processing (NLP) 모델이였기에 지원 기능이 제한적이였으며, OpenFlow Switch 만을 지원한다는 한계가 존재한다. 또한, 다른 종래 기술들에서는 고수준의 NLP 모델을 적용하며 교육환경에서 사용할 수 있는 네트워크 설정 도우미를 제안하였다. 이들은 자연어 입력을 해석하고, 이를 기반으로 이미 정해둔 특정 설정을 실행시키도록 하는 방식을 채택하였다. 하지만, 이는 유연한 구조를 강조하는 현대의 네트워크에는 알맞지 않다.

【0004】상기와 같이, 종래 기술의 경우, 네트워크 설정에 유연성을 가질 수 없었던 이유는 데이터의 부족이 가장 큰 원인이다. 네트워크 설정에 관한 데이터는 공개된 것이 극히 적고, 제한적이다. 이를 획득하기 위해서 직접적인 설정이 필수적인데, 이로 인하여 다양한 환경에서의 적절한 네트워크 설정을 적용하는 데에 한계가 있다.

【발명의 내용】

【해결하고자 하는 과제】

【0005】본 개시는 전술한 종래 기술의 문제점을 해결하기 위하여, 대규모 언어 모델(Large Language Model, LLM)과 네트워크 디지털 트윈(Network Digital Twin, NDT) 기법을 활용하여 네트워크 장비의 설정을 자동화하는 그 목적이 있다.

【과제의 해결 수단】

【0006】본 개시의 일 실시예에 따른 외부의 네트워크를 구축하는 장비들의 설정을 자동화하는 네트워크 설정 자동화 장치에 있어서, 설정 관련 입력을 수신하여 네트워크 장비를 설정하는 대규모 언어 모델(LLM)을 저장하는 메모리; 상기 대규모 언어 모델에 기반하여 상기 입력된 문장을 이용한 네트워크 장비 설정을 수행하는 프로세서; 및 상기 네트워크 장비 설정 결과를 표시하는 입출력 장치를 포함할 수 있다.

【0007】또한, 상기 입출력 장치는, 문장의 형태로 상기 설정 관련 입력을 수신하는 것을 특징으로 할 수 있다.

【0008】또한, 상기 입출력 장치는, 상기 문장에 포함된 사용자의 요구사항을 획득하고, 상기 프로세서는, 상기 요구사항을 상기 대규모 언어 모델에 입력하는 것을 특징으로 할 수 있다.

【0009】또한, 상기 입출력 장치는, 상기 문장에 포함된 상기 네트워크를 구축하는 장비들에 관한 네트워크 상태 관련 정보를 획득하고, 상기 프로세서는, 상기 네트워크 상태 관련 정보를 상기 대규모 언어 모델에 입력하는 것을 특징으로 할 수 있다.

【0010】또한, 상기 입출력 장치는, 상기 문장 입력 이전의 대화 이력을 수신하는 것을 특징으로 할 수 있다.

【0011】또한, 상기 입출력 장치는, 상기 대규모 언어 모델의 출력 형태에 관한 요구사항을 수신하며, 상기 프로세서는, 상기 출력 형태에 관한 요구사항에

기반하여 상기 네트워크 장비 설정 결과를 표시하는 것을 특징으로 할 수 있다.

【0012】 또한, 상기 입출력 장치는, 상기 네트워크를 구축하는 장비들에 관한 설정 매뉴얼을 수신하며, 상기 프로세서는, 상기 설정 매뉴얼을 상기 대규모 언어 모델에 입력하는 것을 특징으로 할 수 있다.

【0013】 또한, 통신 장치를 더 포함하며, 상기 프로세서는, 상기 통신 장치를 통해 연결된 네트워크 디지털 트윈(Network Digital Twin)을 통해 상기 네트워크 장비 설정 결과를 구현하고, 상기 구현 결과를 상기 입출력 장치를 통해 표시하는 것을 특징으로 할 수 있다.

【0014】 또한, 상기 프로세서는, 상기 네트워크 디지털 트윈으로부터 상기 네트워크 장비 설정에 관한 검증 데이터를 수신하고, 상기 검증 데이터를 상기 대규모 언어 모델에 반영하는 것을 특징으로 할 수 있다.

【0015】 또한, 상기 프로세서는, 상기 검증 데이터를 이용하여 상기 대규모 언어 모델을 재학습하며, 상기 재학습된 대규모 언어 모델을 이용하여 상기 네트워크 장비들에 관한 설정을 최종적으로 수행하는 것을 특징으로 할 수 있다.

【0016】 본 개시의 다른 실시예에 따른 외부의 네트워크를 구축하는 장비들의 설정을 자동화하는 네트워크 설정 자동화 장치의 네트워크 설정 자동화 방법에 있어서, 상기 네트워크 설정 자동화 장치의 메모리가, 설정 관련 입력에 기반하여 네트워크 장비를 설정하는 대규모 언어 모델(LLM)을 저장하는 단계; 상기 네트워크 설정 자동화 장치의 입출력 장치가, 상기 설정 관련 입력을 수신하는 단계; 상기

네트워크 설정 자동화 장치의 프로세서가, 상기 대규모 언어 모델에 기반하여 상기 입력된 문장을 이용한 네트워크 장비 설정을 수행하는 단계; 및 상기 입출력 장치가, 상기 네트워크 장비 설정 결과를 표시하는 단계를 포함할 수 있다.

【0017】 또한, 문장의 형태로 상기 설정 관련 입력을 수신하는 단계를 포함할 수 있다.

【0018】 또한, 상기 문장에 포함된 사용자의 요구사항을 획득하는 단계, 및 상기 요구사항을 상기 대규모 언어 모델에 입력하는 단계를 포함할 수 있다.

【0019】 또한, 상기 문장에 포함된 상기 네트워크를 구축하는 장비들에 관한 네트워크 상태 관련 정보를 획득하는 단계, 및 상기 네트워크 상태 관련 정보를 상기 대규모 언어 모델에 입력하는 단계를 포함할 수 있다.

【0020】 또한, 상기 문장 입력 이전의 대화 이력을 수신하는 단계를 포함할 수 있다.

【0021】 또한, 상기 대규모 언어 모델의 출력 형태에 관한 요구사항을 수신하는 단계, 및 상기 출력 형태에 관한 요구사항에 기반하여 상기 네트워크 장비 설정 결과를 표시하는 단계를 포함할 수 있다.

【0022】 또한, 상기 네트워크를 구축하는 장비들에 관한 설정 매뉴얼을 수신하는 단계, 및 상기 설정 매뉴얼을 상기 대규모 언어 모델에 입력하는 단계를 포함할 수 있다.

【0023】또한, 외부의 네트워크 디지털 트윈(Network Digital Twin)을 통해 상기 네트워크 장비 설정 결과를 구현하는 단계, 및 상기 구현 결과를 상기 입출력 장치를 통해 표시하는 단계를 포함할 수 있다.

【0024】또한, 상기 네트워크 디지털 트윈으로부터 상기 네트워크 장비 설정에 관한 검증 데이터를 수신하는 단계, 및 상기 검증 데이터를 상기 대규모 언어 모델에 반영하는 단계를 포함할 수 있다.

【0025】본 개시의 외부의 네트워크를 구축하는 장비들의 설정을 자동화하는 네트워크 설정 자동화 장치에 있어서, 외부로부터 데이터를 송수신하는 통신 장치; 설정 관련 입력을 수신하여 네트워크 장비를 설정하는 대규모 언어 모델(LLM)을 저장하는 메모리; 상기 대규모 언어 모델에 기반하여 상기 입력된 문장을 이용한 네트워크 장비 설정을 수행하고, 상기 통신 장치를 통해 연결된 네트워크 디지털 트윈(Network Digital Twin)을 통해 상기 네트워크 장비 설정 결과를 구현하는 프로세서; 및 상기 네트워크 장비 설정 결과를 표시하는 입출력 장치를 포함할 수 있다.

【발명의 효과】

【0026】본 개시에 의하면, 대규모 언어 모델을 활용하여 네트워크 설정 성능을 개선시킬 수 있다.

【0027】또한, LLM을 이용할 경우, 다양한 데이터를 학습하여 일반적인 상황에서도 높은 성능을 나타낼 수 있으며, 네트워크 설정에 필요한 정보를 문맥 학습

(Context Learning)과 검색 보강 생성(RAG)을 통해 더욱 정확히 처리할 수 있다.

【0028】 또한, NDT를 도입하여 검증을 수행함으로써, 네트워크 설정의 안정성과 신뢰성을 확보할 수 있다.

【0029】 또한, 네트워크 설정에 관한 데이터를 NDT를 통해 충분히 수집하고, 지속적으로 모델의 성능을 개선할 수 있다.

【0030】 또한, NDT에 적용된 설정과 이후에 수집된 설정 결과를 통해 피드백을 LLM에 지속적으로 전달하여 네트워크 설정의 정확도와 효율성을 높일 수 있다.

【0031】 또한, NDT 기법을 이용하여 기존 데이터셋의 한계를 극복하고, 더욱 다양한 환경에서의 네트워크 설정을 지원할 수 있다.

【0032】 또한, 사용자는 별도로 제공되는 인터페이스를 통해 네트워크 설정 도우미와 소통하며, 생성된 네트워크 설정과 검증 결과를 확인할 수 있다.

【0033】 또한, 사용자 인터페이스를 통해서 사용자가 네트워크 설정 과정에 능동적으로 참여할 수 있도록 지원할 수 있다.

【도면의 간단한 설명】

【0034】 도 1은 본 개시의 실시예에 따른 네트워크 설정 자동화 방법을 나타낸 흐름도이다.

도 2는 본 개시의 실시예에 따른 네트워크 설정 자동화 장치의 구성을 나타낸 블록도이다.

도 3은 본 개시의 실시예에서 사용자의 입력을 수신하여 네트워크 설정을 자

동으로 수행하는 전체 시스템 구조를 도시한다.

도 4 및 도 5는 대규모 언어 모델에 입력되는 데이터를 도시한다.

도 6은 네트워크 디지털 트윈에 설정 결과를 구현하는 과정을 도시한다.

도 7은 네트워크 설정에 관한 검증 결과를 실제 가용 네트워크에 적용하는 과정을 도시한다.

도 8은 네트워크 설정 자동화를 위한 사용자 인터페이스의 하나의 예를 도시한다.

【발명을 실시하기 위한 구체적인 내용】

【0035】[본 명세서의 용어 설명]

【0036】이하에서 설명되는 모든 실시 예들은 본 개시의 이해를 돕기 위해 예시적으로 나타낸 것이며, 여기에 설명된 실시 예들과 다르게 변형되어 다양한 실시 형태로 실시될 수 있다. 또한, 본 개시를 설명함에 있어서, 관련된 공지 기능 혹은 공지 구성요소에 대한 구체적인 설명이 본 개시의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우, 그 구체적인 설명은 생략하도록 한다.

【0037】첨부된 도면은 개시의 이해를 돕기 위해서 실제 축척대로 도시된 것이 아니라 일부 구성요소의 치수가 과장되게 도시될 수 있으며, 각 구성요소들에 참조번호를 기재할 때, 동일한 구성요소들에 대해서는 다른 도면에 표시되더라도 가능한 한 동일한 부호로 표시하였다.

【0038】또한, 본 개시의 실시 예의 구성 요소를 설명하는 데 있어서, 제1, 제2, A, B, (a), (b) 등의 용어를 사용할 수 있다. 이러한 용어는 그 구성 요소를 다른 구성 요소와 구별하기 위한 것일 뿐, 그 용어에 의해 해당 구성 요소의 본질이나 차례 또는 순서 등이 한정되지 않는다. 어떤 구성 요소가 다른 구성요소에 '연결', '결합' 또는 '접속'된다고 기재된 경우, 그 구성 요소는 그 다른 구성요소에 직접적으로 연결, 결합 또는 접속될 수 있지만, 그 구성 요소와 그 다른 구성요소 사이에 또 다른 구성 요소가 '연결', '결합' 또는 '접속'될 수도 있다고 이해되어야 할 것이다.

【0039】따라서, 본 명세서에 기재된 실시 예와 도면에 도시된 구성은 본 개시의 가장 바람직한 실시 예에 불과할 뿐이고 본 개시의 기술적 사상을 모두 대변하는 것은 아니므로, 본 개시에 대한 다양한 변형 실시 예들이 있을 수 있다.

【0040】그리고, 본 명세서 및 청구범위에서 사용된 용어나 단어는 통상적이거나 사전적인 의미로 한정되어서는 안되며, 발명자는 그 자신의 개시를 가장 최선의 방법으로 설명하기 위해 용어의 개념을 적절하게 정의할 수 있다는 원칙에 입각하여 본 개시의 기술적 사상에 부합하는 의미와 개념으로 해석되어야만 한다.

【0041】또한, 본 출원에서 사용된 단수의 표현은 문맥상 명백히 다른 것을 뜻하지 않는 한, 복수의 표현을 포함한다.

【0042】[네트워크 설정 자동화 방법: 도 1]

【0043】도 1은 본 개시의 실시예에 따른 네트워크 설정 자동화 방법을 나타낸 흐름도이다.

【0044】도 1에 도시된 바와 같이, 본 개시의 실시예에 따른 네트워크 설정 자동화 장치의 네트워크 설정 자동화 방법(S100)은 S101, S103, S015, S107, S109, S111, S113 단계를 포함하며, 상세한 설명은 하기와 같다.

【0045】먼저, 네트워크 설정 자동화 장치는 문장을 입력으로 하여 외부의 네트워크 장비들에 관한 설정 정보를 출력하는 대규모 언어 모델을 저장한다(S101).

【0046】이어서, 네트워크 설정 자동화 장치는 네트워크 장비 설정 관련 입력을 수신한다(S103).

【0047】여기서, 네트워크 설정 자동화 장치는 별도의 사용자 인터페이스를 통해 네트워크 장비 설정 관련 입력을 수신할 수 있다.

【0048】여기서, 설정 관련 입력은 문장의 형태가 될 수도 있다. 또한, 설정 관련 입력은 네트워크 장비 설정에 관한 사용자 요구사항을 포함할 수 있다. 또한, 설정 관련 입력은 네트워크 상태와 관련된 정보를 포함할 수 있다. 또한, 설정 관련 입력은 사용자와 대규모 언어 모델 사이의 과거 대화 이력을 포함할 수 있다. 또한, 설정 관련 입력은 요구 출력 형태에 관한 정보를 포함할 수 있다. 또한, 설정 관련 입력은 네트워크 관련 매뉴얼 또는 네트워크 장비의 설정과 관련된 매뉴얼을 포함할 수 있다. 여기서, 매뉴얼은 인터넷 상에 공개된 네트워크 관련 매뉴얼이

될 수 있다.

【0049】 그 다음, 네트워크 설정 자동화 장치는 대규모 언어 모델에 기반하여 문장을 이용한 네트워크 장비 설정을 수행한다(S105).

【0050】 이어서, 네트워크 설정 자동화 장치는 네트워크 장비 설정 결과 및 검증 결과를 네트워크 디지털 트윈을 통해 구현하고 표시할 수 있다(S107).

【0051】 여기서, 네트워크 설정 자동화 장치는 네트워크 장비 설정 결과를 사용자 인터페이스를 통해 출력할 수 있다.

【0052】 그 다음, 네트워크 설정 자동화 장치는 네트워크 디지털 트윈을 통해 수집되는 데이터를 대규모 언어 모델에 반영할 수 있다(S109).

【0053】 구체적으로, 네트워크 설정 자동화 장치는 네트워크 디지털 트윈을 통해 네트워크 장비들에 관한 데이터를 수집할 수 있다. 또한, 네트워크 설정 자동화 장치는 네트워크 디지털 트윈을 통해 수집된 데이터들을 대규모 언어 모델에 입력할 수 있다.

【0054】 이어서, 네트워크 설정 자동화 장치는 네트워크 디지털 트윈을 통해 수집된 데이터에 기반하여 대규모 언어 모델을 재학습시킬 수 있다(S111).

【0055】 그 다음, 네트워크 설정 자동화 장치는 재학습된 대규모 언어 모델을 이용하여 네트워크 장비 설정을 최종적으로 수행하고 그 결과를 표시할 수 있다(S113).

【0056】 [네트워크 설정 자동화 장치: 도 2]

【0057】도 2는 본 개시의 실시예에 따른 네트워크 설정 자동화 장치의 구성을 나타낸 블록도이다.

【0058】도 2에 도시된 바와 같이, 본 개시의 실시예에 따른 네트워크 설정 자동화 장치(200)는 입출력 장치(210), 프로세서(220), 메모리(230) 및 통신 장치(240)를 포함할 수 있다.

【0059】입출력 장치(210)는 외부로부터 입력 데이터를 수신할 수 있다. 입출력 장치(210)는 외부로 네트워크 장비 설정 결과를 출력할 수 있다. 입출력 장치(210)는 네트워크 장비 설정에 관한 검증 결과(네트워크 디지털 트윈 반영 결과)를 표시할 수 있다. 입출력 장치(210)는 상기한 데이터를 수신하고 결과를 출력하기 위한 사용자 인터페이스 표시하는 터치 스크린이 될 수도 있으나, 반드시 이에 한정될 필요는 없다.

【0060】메모리(230)는 프로세서의 동작에 필요한 데이터를 저장할 수 있다. 특히, 메모리(230)는 네트워크 장비 설정에 필요한 대규모 언어 모델(231)을 저장할 수 있다.

【0061】프로세서(220)는 도 1을 참조하여 설명한 네트워크 설정 자동화 장치의 동작들 중 S105, S107, S109, S111 및 S113 단계를 수행할 수 있다. 구체적으로, 프로세서(220)는 네트워크 장비 설정 관련 입력이 수신되면, 대규모 언어 모델을 이용하여 문장을 이용한 네트워크 장비 설정을 수행할 수 있다. 또한, 프로세서(220)는 네트워크 장비 설정 결과 및 네트워크 장비 설정 검증 결과를 네트워크 디

지털 트윈을 통해 구현할 수 있다. 또한, 프로세서(220)는 네트워크 디지털 트윈을 통해 수집되는 데이터를 대규모 언어 모델에 반영하고, 수집 데이터에 기반하여 대규모 언어 모델을 재학습시킬 수 있다. 또한, 프로세서(220)는 재학습된 대규모 언어 모델을 이용하여 네트워크 장비 설정을 최종적으로 수행하고, 그 결과를 표시할 수 있다.

【0062】 통신 장치(240)는 외부의 네트워크 디지털 트윈(250)으로 네트워크 장비 설정 결과를 전송할 수 있다. 또한, 통신 장치(240)는 네트워크 디지털 트윈(250)으로부터 네트워크 장비 설정 검증 결과를 수신하고, 프로세서(220)로 전달할 수 있다.

【0063】 한편, 도 2에 도시된 바와 다르게, 대규모 언어 모델(231)은 네트워크 설정 자동화 장치 외부에 구성될 수도 있다. 이 경우, 프로세서(22)는 통신 장치(240)를 통해 대규모 언어 모델에 데이터를 입력하고, 통신 장치(240)를 통해 대규모 언어 모델의 출력을 수신할 수도 있다.

【0064】 [네트워크 설정 자동화 장치 전체 구조: 도 3]

【0065】 도 3은 본 개시의 실시예에서 사용자의 입력을 수신하여 네트워크 설정을 자동으로 수행하는 전체 시스템 구조를 도시한다.

【0066】 도 3에 도시된 바와 같이, 사용자는 네트워크에 필요한 요구사항을 네트워크 설정 자동화 장치(310)의 프로세서(도 2의 프로세서(220))에 내장된 네트워크 설정 도우미에게 전달한다.

【0067】네트워크 설정 도우미는 사용자의 요구사항을 LLM을 통해서 분석하고 적절한 네트워크 장비 설정을 생성한다.

【0068】LLM을 통해서 생성된 네트워크 설정의 안정성을 확인하기 위해서 이를 바로 실제 네트워크에 적용하는 것이 아니라 우선적으로 NDT(320)를 통해 구현/실행할 수 있고, 또한, NDT를 통해 검증을 수행할 수 있다.

【0069】이를 위해서, 네트워크 설정 도우미는 생성한 네트워크 설정을 프로세서(220)에 내장된 NDT 컨트롤러에게 전달하고, NDT 컨트롤러는 이를 이용하여 NDT에 존재하는 각 네트워크 장비의 설정을 수행한다.

【0070】네트워크 장비의 설정을 완료한 이후, 프로세서는 NDT에서 테스트를 수행하며, 데이터를 수집한다. 이를 NDT 모니터가 수행하며, 최종 결과를 네트워크 설정 도우미에게 피드백으로 제공한다.

【0071】검증 프로세스를 완료한 후, 프로세서는 NDT에 적용된 설정을 실제 가용 네트워크(330)에 적용할 수 있다.

【0072】[대규모 언어 모델 입력: 도 4 - 5]

【0073】앞서 도 3을 참조하여 설명한 바와 같이, 네트워크 설정 자동화 장치의 프로세서에 내장된 서네트워크 설정 도우미가 네트워크 설정을 생성하는 과정에서 LLM을 활용한다. LLM은 다양한 목적의 데이터를 대량으로 입력하여 일반적인 상황에서 굉장히 높은 성능을 보여주고 있다. 하지만, 네트워크 설정이라는 구체적인 목적을 성공적으로 달성하기 위해서는 문맥 정보를 전달하는 Context Learning

이나 네트워크 설정에 필요한 정보를 추가로 제공하는 Retrieval Augmented Generation (RAG) 방법의 도입이 필요하다. 따라서 본 개시에서는 도 3의 네트워크 설정 도우미의 추론 과정에서 도 4 및 도 5와 같은 데이터 입력 방식을 적용한다.

【0074】 도 4 및 도 5는 대규모 언어 모델에 입력되는 데이터를 도시한다.

【0075】 도 4에 도시된 바와 같이, 네트워크 설정에 대한 기초적인 지식들을 LLM에게 전달하여 LLM이 사용자의 요구사항이 어떤 문맥에서 필요한지에 대한 정보를 전달하여 LLM의 생성 성능을 높일 수 있다.

【0076】 예를 들어, 네트워크 설정 자동화 장치가 수신할 수 있는 입력의 종류로, 요구사항 뿐만 아닐, 라우터 스위치의 개수와 연결 상태와 같은 현재 네트워크 상태에 대한 정보를 포함할 수 있다. 또한, 입력의 종류로 이전까지의 대화 내용이 포함될 수도 있다. 또한, 입력의 종류로, 생성하기를 원하는 출력의 형태 또는 출력 예시를 포함할 수도 있다.

【0077】 도 5에 도시된 바와 같이, 네트워크 설정 자동화 장치는 RAG(간섭 보강) 기술을 이용할 수도 있다.

【0078】 LLM의 입력 토큰 용량의 한계가 있기 때문에, 문맥 정보로 네트워크 장비의 모든 매뉴얼을 같이 전달하는 것은 불가능하다. 따라서, 요구사항에 알맞은 장비에 대한 최소한의 매뉴얼을 전달하는 것이 가능하다면, 적은 토큰을 사용하면서도 더 높은 성능을 보이는 출력을 생성할 수 있다.

【0079】예를 들어, 네트워크 설정 자동화 장치는 입력으로서 수신할 수 있는 데이터로서 매뉴얼에 한정되지 않으며, 인터넷에 업로드 되어있는 다양한 설정 참고 사항 또는 QnA 정보를 포함할 수도 있다.

【0080】[네트워크 디지털 트윈 동작: 도 6 - 7]

【0081】도 6은 네트워크 디지털 트윈에 설정 결과를 구현하는 과정을 도시한다.

【0082】도 6에 도시된 바와 같이, 네트워크 설정 자동화 장치의 네트워크 설정 도우미의 LLM의 성능을 향상시키기 위한 미세 조정(Fine-Tuning)을 수행할 수 있다.

【0083】그 방법으로서, 네트워크 설정 자동화 장치는 네트워크 디지털 트윈 상에 네트워크 장비 설정 내용을 구현하고, 이를 NDT 모니터(네트워크 설정 자동화 장치의 입출력 장치)를 통해 표시할 수 있다.

【0084】또한, 네트워크 설정 자동화 장치는 네트워크 디지털 트윈으로부터 데이터를 수집하고, 이를 대규모 언어 모델에 반영하여 대규모 언어 모델을 재학습시킬 수 있다. 이 과정에서 네트워크의 성능 지표 latency, bandwidth 정보를 활용하거나 SLA Violation과 같은 QoS 정보를 활용할 수도 있다. 예를 들어, latency를 최소화하는 설정에 더 높은 가치를 부여하거나 서비스 가용성을 최대화하는 방식의 피드백을 제공할 수 있다.

【0085】도 7은 네트워크 설정에 관한 검증 결과를 실제 가용 네트워크에 적용하는 과정을 도시한다.

【0086】도 7에 도시된 바와 같이, 네트워크 설정 자동화 장치는 네트워크 설정 내용을 네트워크 디지털 트윈에 반영하고, 네트워크 디지털 트윈을 NDT 모니터(710)를 통해 모니터링하며, 모니터링 결과에 기반하여 대규모 언어 모델을 재학습시키고, 네트워크 장비 설정 최종 결과를 실제 가용 네트워크에 적용할 수 있다.

【0087】기존의 LLM이 가지는 치명적인 단점은 불안정한 생성 등으로 인하여 생성된 네트워크 설정에 대한 신뢰를 100% 보장할 수 없다는 점이다. 따라서, 본 개시에서는 이러한 문제를 완화하기 위해서 NDT를 통한 검증을 필수적으로 포함한다. 실제 네트워크에 적용하기 이전에 검증 실험을 NDT에서 수행하여 적용 여부를 결정할 수 있다. 이 방법을 통해서 LLM이 가지는 불안정성을 완화할 수 있다.

【0088】상기한 바를 통해, LLM을 통해서 생성한 네트워크 설정을 NDT에서 실행하고, 네트워크 통신 안정성, 네트워크 지연시간 등과 같은 피드백을 획득할 수 있다.

【0089】[사용자 인터페이스: 도 8]

【0090】도 8은 네트워크 설정 자동화를 위한 사용자 인터페이스의 하나의 예를 도시한다.

【0091】도 8에 도시된 바와 같이 네트워크 설정 자동화 장치는 입출력 장치를 통해 사용자가 직접적으로 접근할 수 있는 인터페이스를 제공할 수 있다.

【0092】사용자 인터페이스 내에서, 네트워크 설정 자동화 장치는 사용자의 입력을 대화(채팅) 형태로 주고받을 수 있으며, 전체 네트워크 정보를 NDT로 구축할 수 있다.

【0093】또한, 네트워크 설정 자동화 장치는 별도로 제공되는 인터페이스를 통해서 요구사항을 수신하고 이를 대규모 언어 모델로 전달하며, 대규모 언어 모델로부터 네트워크 설정 결과를 획득한다.

【0094】[본 명세서의 해석 방법]

【0095】이상 첨부된 도면을 참조하여 본 개시의 실시 예들을 더욱 상세하게 설명하였으나, 본 개시는 반드시 이러한 실시 예로 국한되는 것은 아니고, 본 개시의 기술사상을 벗어나지 않는 범위 내에서 다양하게 변형 실시될 수 있다.

【0096】따라서, 본 개시에 개시된 실시 예들은 본 개시의 기술 사상을 한정하기 위한 것이 아니라 설명하기 위한 것이고, 이러한 실시 예에 의하여 본 개시의 기술 사상의 범위가 한정되는 것은 아니다. 그러므로, 이상에서 기술한 실시 예들은 모든 면에서 예시적인 것이며 한정적이 아닌 것으로 이해해야만 한다. 본 개시의 보호 범위는 아래의 청구범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 개시의 권리범위에 포함되는 것으로 해석되어야 할 것이다.

【청구범위】

【청구항 1】

외부의 네트워크를 구축하는 장비들의 설정을 자동화하는 네트워크 설정 자동화 장치에 있어서,

설정 관련 입력을 수신하여 네트워크 장비를 설정하는 대규모 언어 모델(LLM)을 저장하는 메모리;

상기 대규모 언어 모델에 기반하여 상기 입력된 문장을 이용한 네트워크 장비 설정을 수행하는 프로세서; 및

상기 네트워크 장비 설정 결과를 표시하는 입출력 장치를 포함하는,

네트워크 설정 자동화 장치.

【청구항 2】

제1항에 있어서,

상기 입출력 장치는,

문장의 형태로 상기 설정 관련 입력을 수신하는 것을 특징으로 하는,

네트워크 설정 자동화 장치.

【청구항 3】

제2항에 있어서,

상기 입출력 장치는,

상기 문장에 포함된 사용자의 요구사항을 획득하고,

상기 프로세서는,

상기 요구사항을 상기 대규모 언어 모델에 입력하는 것을 특징으로 하는,

네트워크 설정 자동화 장치.

【청구항 4】

제3항에 있어서,

상기 입출력 장치는,

상기 문장에 포함된 상기 네트워크를 구축하는 장비들에 관한 네트워크 상태 관련 정보를 획득하고,

상기 프로세서는,

상기 네트워크 상태 관련 정보를 상기 대규모 언어 모델에 입력하는 것을 특징으로 하는,

네트워크 설정 자동화 장치.

【청구항 5】

제2항에 있어서,

상기 입출력 장치는,

상기 문장 입력 이전의 대화 이력을 수신하는 것을 특징으로 하는,

네트워크 설정 자동화 장치.

【청구항 6】

제2항에 있어서,

상기 입출력 장치는,

상기 대규모 언어 모델의 출력 형태에 관한 요구사항을 수신하며,

상기 프로세서는,

상기 출력 형태에 관한 요구사항에 기반하여 상기 네트워크 장비 설정 결과를 표시하는 것을 특징으로 하는,

네트워크 설정 자동화 장치.

【청구항 7】

제2항에 있어서,

상기 입출력 장치는,

상기 네트워크를 구축하는 장비들에 관한 설정 매뉴얼을 수신하며,

상기 프로세서는,

상기 설정 매뉴얼을 상기 대규모 언어 모델에 입력하는 것을 특징으로 하는,

네트워크 설정 자동화 장치.

【청구항 8】

제1항에 있어서,

통신 장치를 더 포함하며,

상기 프로세서는,

상기 통신 장치를 통해 연결된 네트워크 디지털 트윈(Network Digital Twin)을 통해 상기 네트워크 장비 설정 결과를 구현하고,

상기 구현 결과를 상기 입출력 장치를 통해 표시하는 것을 특징으로 하는,

네트워크 설정 자동화 장치.

【청구항 9】

제8항에 있어서,

상기 프로세서는,

상기 네트워크 디지털 트윈으로부터 상기 네트워크 장비 설정에 관한 검증 데이터를 수신하고,

상기 검증 데이터를 상기 대규모 언어 모델에 반영하는 것을 특징으로 하는,

네트워크 설정 자동화 장치.

【청구항 10】

제9항에 있어서,

상기 프로세서는,

상기 검증 데이터를 이용하여 상기 대규모 언어 모델을 재학습하며,

상기 재학습된 대규모 언어 모델을 이용하여 상기 네트워크 장비들에 관한 설정을 최종적으로 수행하는 것을 특징으로 하는,

네트워크 설정 자동화 장치.

【청구항 11】

외부의 네트워크를 구축하는 장비들의 설정을 자동화하는 네트워크 설정 자동화 장치의 네트워크 설정 자동화 방법에 있어서,

상기 네트워크 설정 자동화 장치의 메모리가, 설정 관련 입력에 기반하여 네트워크 장비를 설정하는 대규모 언어 모델(LLM)을 저장하는 단계;

상기 네트워크 설정 자동화 장치의 입출력 장치가, 상기 설정 관련 입력을 수신하는 단계;

상기 네트워크 설정 자동화 장치의 프로세서가, 상기 대규모 언어 모델에 기반하여 상기 입력된 문장을 이용한 네트워크 장비 설정을 수행하는 단계; 및

상기 입출력 장치가, 상기 네트워크 장비 설정 결과를 표시하는 단계를 포함하는,

방법.

【청구항 12】

제11항에 있어서,

문장의 형태로 상기 설정 관련 입력을 수신하는 단계를 포함하는,

방법.

【청구항 13】

제12항에 있어서,

상기 문장에 포함된 사용자의 요구사항을 획득하는 단계, 및

상기 요구사항을 상기 대규모 언어 모델에 입력하는 단계를 포함하는,

방법.

【청구항 14】

제13항에 있어서,

상기 문장에 포함된 상기 네트워크를 구축하는 장비들에 관한 네트워크 상태 관련 정보를 획득하는 단계, 및

상기 네트워크 상태 관련 정보를 상기 대규모 언어 모델에 입력하는 단계를 포함하는,

방법.

【청구항 15】

제12항에 있어서,

상기 문장 입력 이전의 대화 이력을 수신하는 단계를 포함하는,

방법.

【청구항 16】

제12항에 있어서,

상기 대규모 언어 모델의 출력 형태에 관한 요구사항을 수신하는 단계, 및

상기 출력 형태에 관한 요구사항에 기반하여 상기 네트워크 장비 설정 결과를 표시하는 단계를 포함하는,

방법.

【청구항 17】

제12항에 있어서,

상기 네트워크를 구축하는 장비들에 관한 설정 매뉴얼을 수신하는 단계, 및

상기 설정 매뉴얼을 상기 대규모 언어 모델에 입력하는 단계를 포함하는,

방법.

【청구항 18】

제11항에 있어서,

외부의 네트워크 디지털 트윈(Network Digital Twin)을 통해 상기 네트워크 장비 설정 결과를 구현하는 단계, 및

상기 구현 결과를 상기 입출력 장치를 통해 표시하는 단계를 포함하는,

방법.

【청구항 19】

제18항에 있어서,

상기 네트워크 디지털 트윈으로부터 상기 네트워크 장비 설정에 관한 검증 데이터를 수신하는 단계, 및

상기 검증 데이터를 상기 대규모 언어 모델에 반영하는 단계를 포함하는,

방법.

【청구항 20】

외부의 네트워크를 구축하는 장비들의 설정을 자동화하는 네트워크 설정 자동화 장치에 있어서,

외부로부터 데이터를 송수신하는 통신 장치;

설정 관련 입력을 수신하여 네트워크 장비를 설정하는 대규모 언어 모델(LLM)을 저장하는 메모리;

상기 대규모 언어 모델에 기반하여 상기 입력된 문장을 이용한 네트워크 장비 설정을 수행하고,

상기 통신 장치를 통해 연결된 네트워크 디지털 트윈(Network Digital Twin)을 통해 상기 네트워크 장비 설정 결과를 구현하는 프로세서; 및

상기 네트워크 장비 설정 결과를 표시하는 입출력 장치를 포함하는,

네트워크 설정 자동화 장치.

【요약서】

【요약】

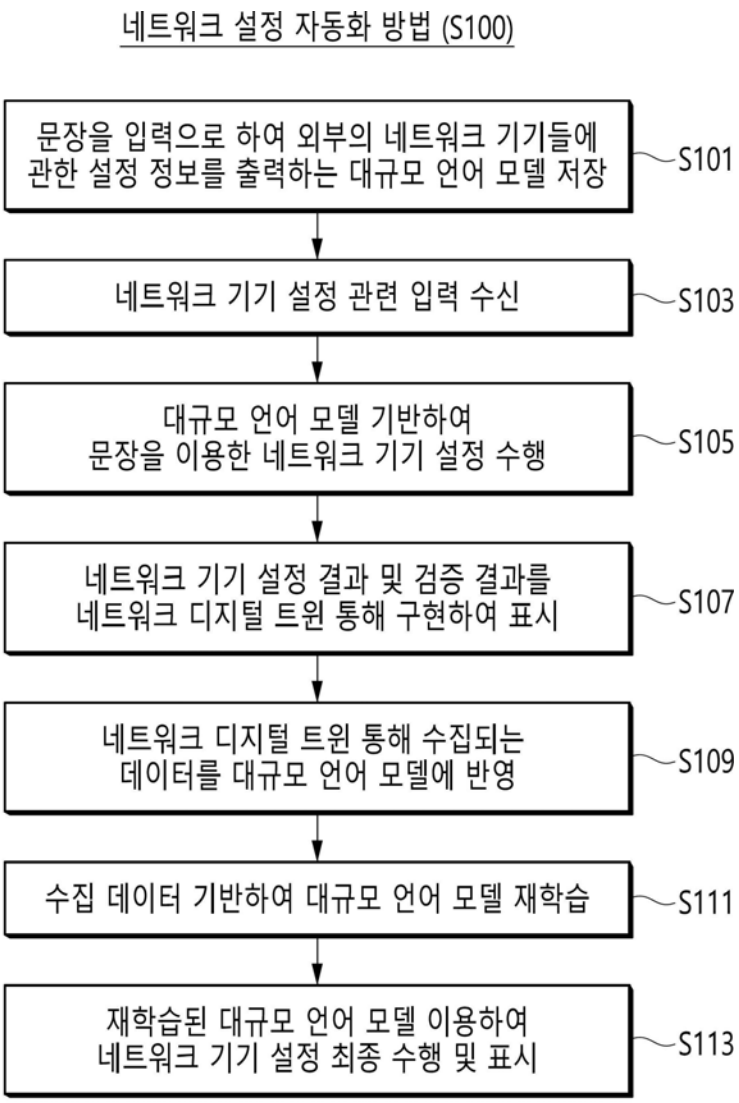
본 개시의 실시예에 따른 외부의 네트워크를 구축하는 장비들의 설정을 자동화하는 네트워크 설정 자동화 장치는 설정 관련 입력을 수신하여 네트워크 장비를 설정하는 대규모 언어 모델(LLM)을 저장하는 메모리, 대규모 언어 모델에 기반하여 입력된 문장을 이용한 네트워크 장비 설정을 수행하는 프로세서, 및 네트워크 장비 설정 결과를 표시하는 입출력 장치를 포함한다.

【대표도】

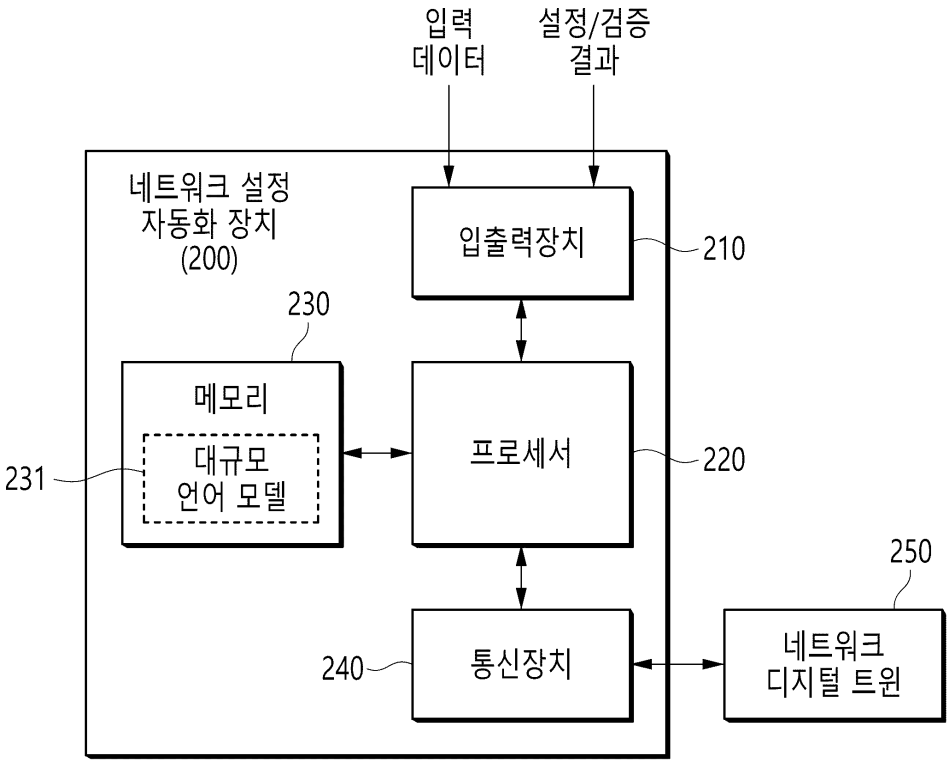
도 1

【도면】

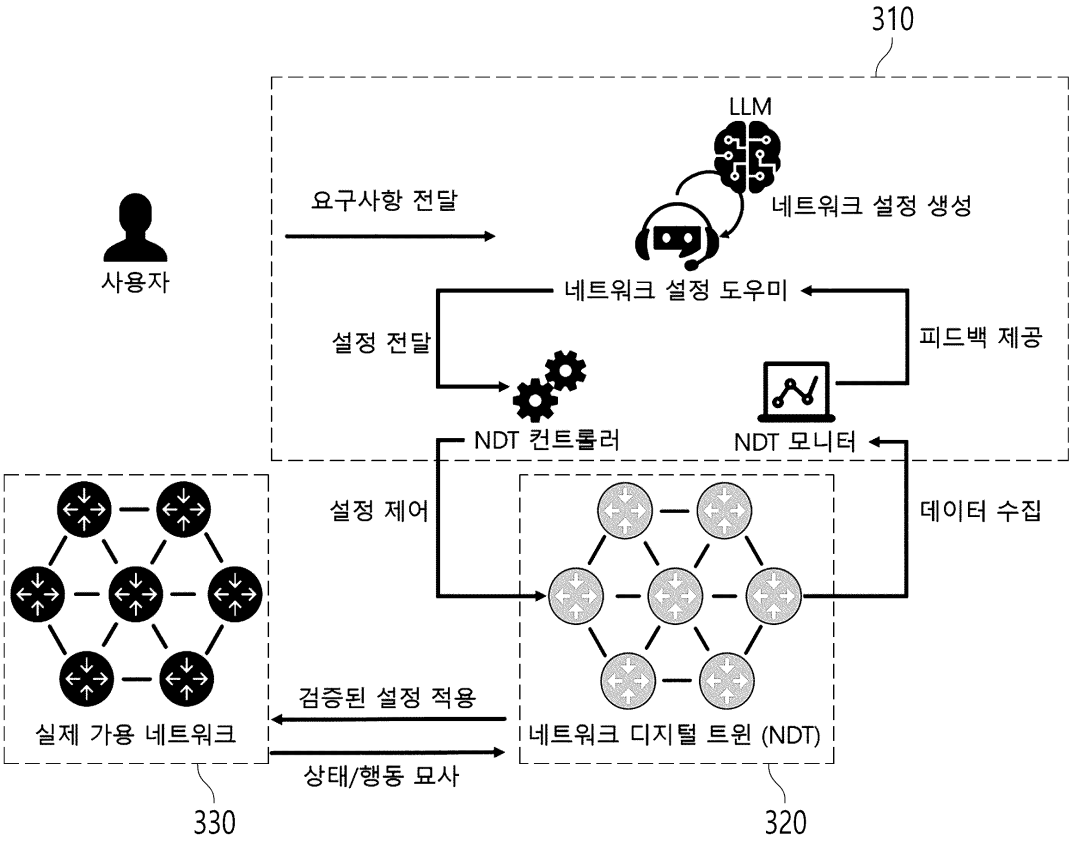
【도 1】


네트워크 설정 자동화 방법 (S100)
문장을 입력으로 하여 외부의 네트워크 기기들에 관한 설정 정보를 출력하는 대규모 언어 모델 저장
S101
네트워크 기기 설정 관련 입력 수신
S103
대규모 언어 모델 기반하여 문장을 이용한 네트워크 기기 설정 수행
S105
네트워크 기기 설정 결과 및 검증 결과를 네트워크 디지털 트윈 통해 구현하여 표시
S107
네트워크 디지털 트윈 통해 수집되는 데이터를 대규모 언어 모델에 반영
S109
수집 데이터 기반하여 대규모 언어 모델 재학습
S111
재학습된 대규모 언어 모델 이용하여 네트워크 기기 설정 최종 수행 및 표시
S113

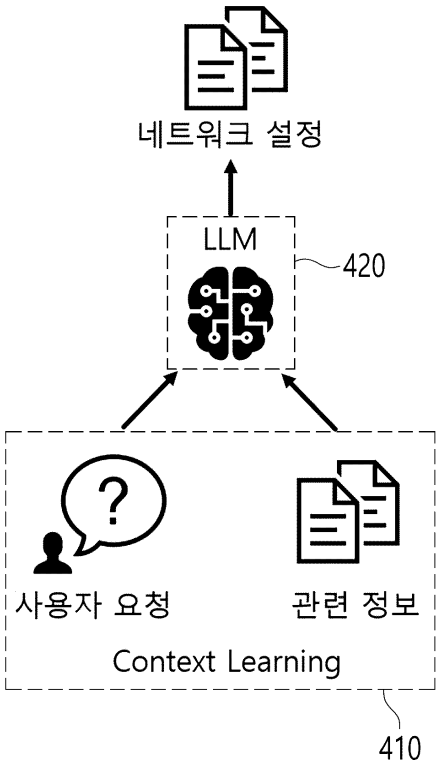
【도 2】


입력
데이터
설정/검증
결과
네트워크 설정
자동화 장치
(200)
입출력장치
210
230
메모리
대규모
언어 모델
231
프로세서
220
250
240
통신장치
네트워크
디지털 트윈

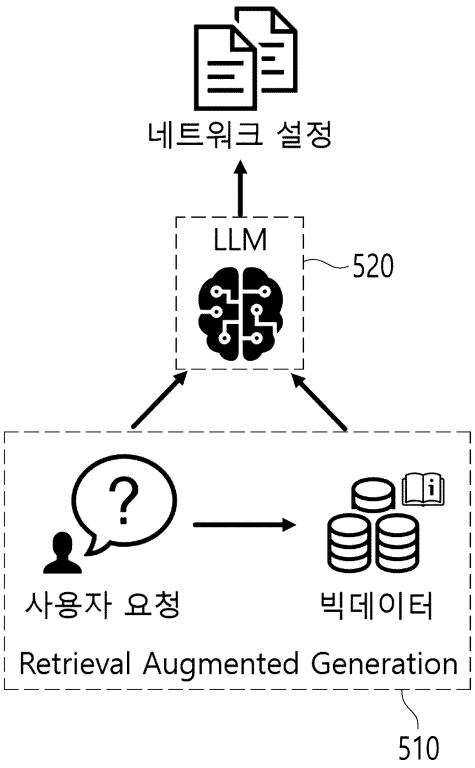
【도 3】


310
LLM
요구사항 전달
네트워크 설정 생성
사용자
네트워크 설정 도우미
설정 전달
피드백 제공
NDT 컨트롤러
NDT 모니터
설정 제어
데이터 수집
검증된 설정 적용
실제 가용 네트워크
네트워크 디지털 트윈 (NDT)
상태/행동 묘사
330
320

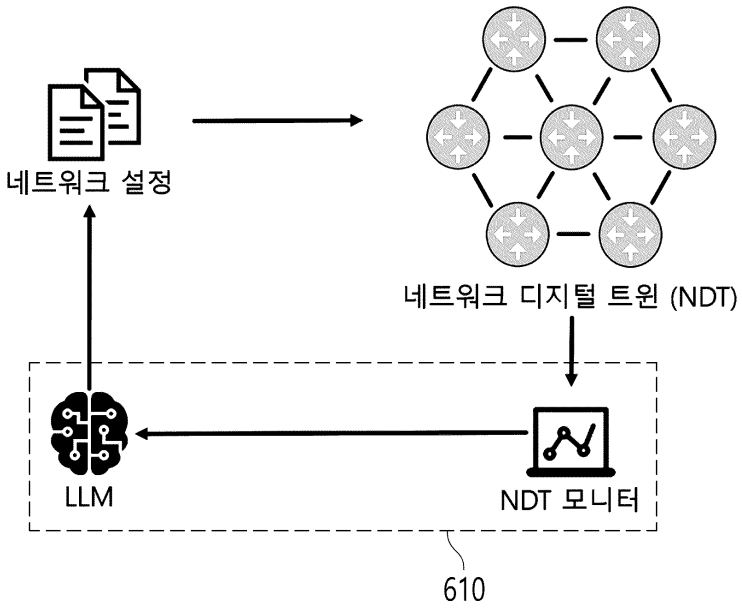
【도 4】



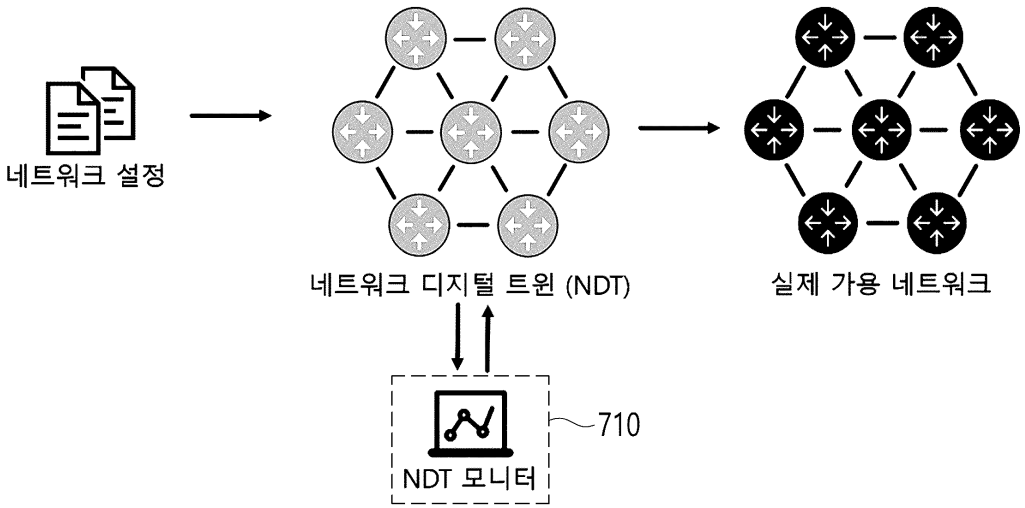
【도 5】


네트워크 설정
LLM
520
사용자 요청
빅데이터
Retrieval Augmented Generation
510

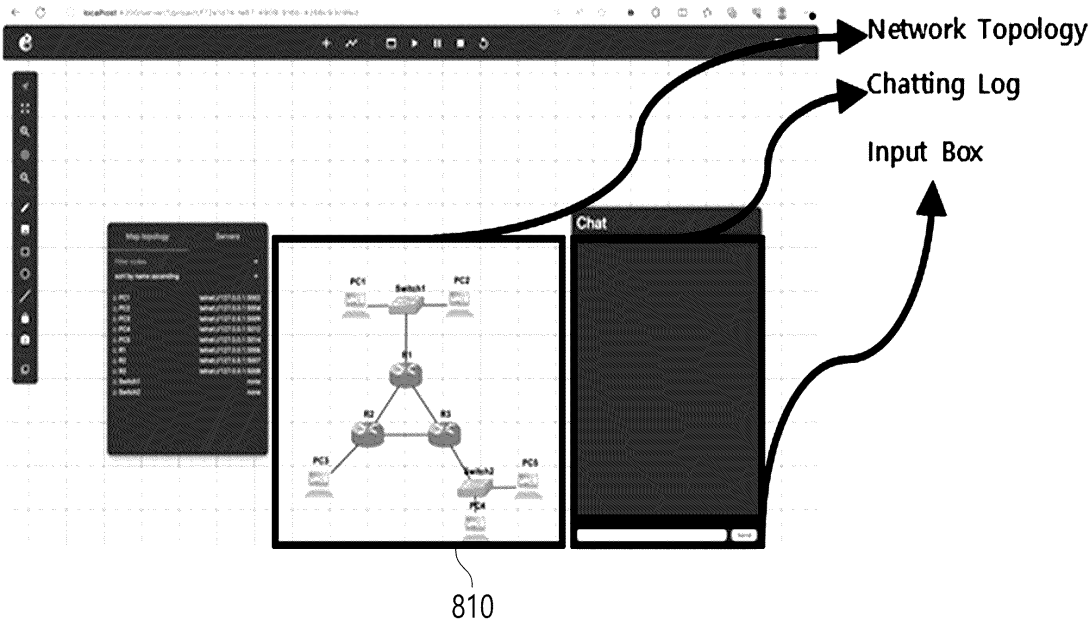
【도 6】


네트워크 설정
네트워크 디지털 트윈 (NDT)
LLM
NDT 모니터
610

【도 7】


네트워크 설정
네트워크 디지털 트윈 (NDT)
실제 가용 네트워크
710
NDT 모니터

【도 8】


Network Topology
Chatting Log
Input Box
Chat
810