



Leveraging generative AI for intent-based networking operations in network slices

Downloaded from: <https://research.chalmers.se>, 2026-04-14 12:57 UTC

Citation for the original published paper (version of record):

Adanza, D., Gifre, L., Alemany, P. et al (2025). Leveraging generative AI for intent-based networking operations in network slices. *Computer Networks*, 272.
<http://dx.doi.org/10.1016/j.comnet.2025.111647>

N.B. When citing this work, cite the original published paper.

Leveraging Generative AI for Intent-Based Networking Operations in Network Slices

Daniel Adanza, LLuis Gifre, Pol Alemany, Carlos Natalino, Paolo Monti,
Raul Muñoz, Ricard Vilalta

^aPacket-Optical Networks and Systems Centre Tecnologic de Telecommunications de Catalunya, Avinguda Carl Friedrich Gauss, 7, Casteldefells, 08860, Catalonia, Spain

Abstract

Large Language Models (LLMs) are among the most popular Generative AI models. They bring benefits like a natural language interface and the automation of complex tasks. Despite their potential, few studies have implemented LLMs for network management. This paper addresses that gap, showcasing in a practical scenario how network management can be efficiently enhanced by automating tasks such as network configuration and traffic analysis, thereby reducing downtime and improving efficiency.

This study presents a LLM agent integrated with a cloud-native SDN controller (ETSI TeraFlowSDN) designed with Retrieval-Augmented Generation (RAG) capabilities to operate with intent-based network operations. The LLM agent understands the context and triggers different operations, such as intent creation, query, and explanation. The results demonstrate a system capable of automating network operations with a factual accuracy of 93% with reasonable computation times, demonstrating how the developed LLM agent can enhance network management.

Keywords: large language models, software-defined networking, intent-based networking and network slices.

1. Introduction

In modern network management, achieving efficient and automated network operations is critical. Intent-Based Networking (IBN) transforms user requests into necessary network operations, configurations, and maintenance tasks to meet the specific needs of the request. IBN address several challenges

in traditional networks, including robust security, flexibility, and enhanced efficiency [22]. Additionally, Software-Defined Networking (SDN) and IBN are reshaping how networks are managed, offering programmability and automation to traditionally static infrastructures. However, current methods cannot effectively interpret user intents written in natural language and manage complex configurations. This leads to inefficiencies, potential increased downtime, and a high level of human intervention needed for operations.

Natural Language Processing (NLP) can simplify network configuration by enabling administrators to use commands simply using natural language, reducing the need for specialized coding skills. This results in error reduction, increased efficiency, and automation, as configurations can be specified more intuitively [4]. Among the noteworthy NLP techniques, several approaches stand out. Large Language Models (LLMs) have emerged as a transformative technology in the field of NLP. Their ability to understand and generate human language has revolutionized applications across industries, including text generation, sentiment analysis, and machine translation [9, 10, 23]. The vast training on diverse datasets enables LLMs to perform robustly across various languages and domains, offering the potential for automation and efficiency in complex tasks. One example of such potential is the integration of agents, which expands the capabilities of LLMs. Their adaptability also facilitates fine-tuning for specific tasks, making them versatile tools for numerous applications in both academic and commercial settings.

To mention a few studies, the authors in [24] combine text mining and a LLM, which results in a question-and-answer system able to understand the topic of materials used in optical networks and to answer appropriately using natural language. Other studies like [21] have also implemented a LLM for the same purpose, covering the topic of alarm compression and fault analysis.

However, LLMs are a very powerful tool and can be useful in many scenarios. Using them as an expert system and creating a question-and-answer tool is only one of the multiple functionalities that they can offer. LLMs can handle complex tasks like context understanding, text analysis and even the generation of complex data structures on demand.

To contribute to the field, a LLM agent is presented, which is capable of understanding the context, inferring the desired operation of the user and acting accordingly. The LLM agent also performs text analysis to validate the user's inputs to finally trigger the necessary network operations according to the user's needs. The LLM agent has proven to be quite robust by reaching 93.3% of factual accuracy.

The presented paper extends the previous work in [17], where a minimum viable product permits the users to generate intents implementing NLP techniques. These NLP techniques are based on a keyword search and one-hot encoding to generate different intents based on the user's requests. Despite presenting a promising use case, the paper does not integrate a LLM in itself. Instead, it takes advantage of a simpler NLP to trigger a single operation, which is the intent generation.

The next step of the research has evolved into an interactive chatbot able to trigger different network operations and partially implemented in an SDN architecture [18, 19]. This work has been published on [2]. The presented paper consists of a natural step forward from the previously presented studies [17, 2], showcasing a LLM which is not merely used as a question-and-answer system as it is generally implemented [5, 14].

In a nutshell, the presented LLM agent leverages novel technologies such as autonomous agents and LLMs, resulting in a practical implementation for IBN, reaching robust accuracy results when tested with real data. The main findings of this study can be summarized in the following points:

- The study showcases a practical implementation of a LLM agent powered with Retrieval-Augmented Generation (RAG) and assembled in a method composed of four different steps to automate network management.
- The implementation of RAG capabilities permits an enhancement of the LLM capabilities by providing a knowledge base adapted to every situation, reaching an average of 96.6% factual accuracy.
- Instead of using LLMs as a mere expert system, the presented solution leverages the capabilities of the LLMs to assist in a wide variety of tasks, such as intent creation, querying and intent explanation. This has been demonstrated in an environment with a cloud-native SDN controller.

The paper is organized as follows. Section II overviews the state of the art, concluding with the main novelty of the presented LLM agent in contrast with the reviewed literature. The proposed solution is then described, illustrating its architecture, its data sources and the respective processes that compose the solution. The experimental evaluation is then described with three different use cases and their corresponding outputs provided by the

LLM, the significance of the results and the execution time results. The paper then concludes with the limitations of the study, future works and the main extracted conclusions.

2. State of the art

A variety of approaches have been proposed in the field of IBN. For instance, [13] integrates intents to perform reconfiguration tasks at a router level using P4 language. This approach has proven effective in real-world scenarios. However, the studies focus on a particular use case, making it challenging to directly compare them with the presented approach. In contrast, studies like [16] present a proof of concept for implementing End-to-End (E2E) IBN within a 5G architecture. This approach supports ubiquity and enhances network efficiency through machine learning (ML) algorithms. While this proof of concept aligns with IBN trends such as ubiquity, AI integration, and increased flexibility, it remains theoretical and lacks practical implementation. Alternatively, other studies such as [1] showcase the ongoing evolution of the IBN technologies by integrating an E2E network slicing system and automating the configuration process and managing orchestration and network slices.

NLP techniques are a perfect complement for IBN systems. They cover the gap between human expressions and machine-executable policies, which results in more accessible and efficient network management. Within the studies that integrated NLP with IBN, it is worth highlighting [12], where a hybrid system is presented combining NLP and neural networks to generate intents within a SDN architecture. This approach has demonstrated practical effectiveness, but it relies on traditional NLP techniques, such as sequence-to-sequence learning models, rather than exploring more advanced technologies like LLMs as it happens in the presented approach.

Despite LLMs being a novel concept, their potential impact on the telecommunications sector is significant. Although not yet widely implemented, there is considerable interest in leveraging these models to revolutionize telecommunications projects and open new opportunities [25].

Current research implementing LLMs in telecommunications can be categorized into two groups. The first group focuses on simplifying network configuration and enhancing human interaction. For instance, [20] demonstrates how LLMs translate high-level policies and requirements into low-level APIs. It has a different scope than the presented paper, which is, however,

useful in its context. Alternatively, some studies emphasize using LLMs as a question-and-answer system. For example, [8] presents various approaches for implementing LLMs to understand telecommunications standards. However, LLMs are a very powerful tool to cope with many different tasks, such as text categorization and context understanding, as it has been demonstrated in the presented paper.

The existing literature on IBN and NLP in network management demonstrates significant advancements in enhancing network automation and user interaction [7]. Various approaches, such as reconfiguration tasks using P4 language and integration of ML in 5G architectures, have shown promising results in improving network efficiency and flexibility [13]. However, these methods often lack adaptability to broader use cases. Similarly, NLP approaches have made strides in simplifying network configurations, but many rely on traditional techniques that do not fully exploit the capabilities of more advanced technologies like LLMs that are able to perform context reasoning and bring some flexibility regarding the user's inputs. In addition to that, the few studies that explored LLM [5, 14]. They merely use it as a question-and-answer system or for translating high-level policies into APIs, and their integration into comprehensive network management solutions remains limited.

Retrieval Augmented Generation (RAG) plays a crucial role in enhancing the capabilities of the LLM. It permits access to up-to-date, factual, and domain-specific information beyond their static training data. LLMs are trained on vast, but fixed, corpora, which can limit their relevance for recent or niche topics. By retrieving real-time or specialized data during generation, RAG enhances the accuracy of LLM, and it allows for more scalable and adaptable solutions [6].

Some examples of studies that implemented RAG in the field of telecommunications are Telco-RAG[5], which has been designed as an expert system to deal with problems related to telecom standards, a handy tool to deal with these particular problems. In a similar way, [14] also integrated a LLM with RAG to enhance a question-and-answer system.

The presented method goes one step further by fully implementing LLMs powered with RAG to carry out different tasks like understanding the context, validating the users' inputs and triggering the desired network operations. As a result, network management becomes more intuitive, less complex and more efficient in terms of time.

3. Proposed solution

The proposed solution is designed to simplify and automate complex network configuration operations. The system follows a series of processes depicted in Figure 1.



Figure 1: Main processes composing the presented solution.

The workflow is triggered by a user query, followed by the LLM understanding the context and inferring the operation desired by the user. Secondly, the LLM asks for the necessary information to carry out the operation for the user. Thirdly, it triggers the necessary network operation and presents the results to the user to proceed with its confirmation. Finally, the desired network intent would be deployed in the network.

3.1. Proposed architecture

The presented solution has the overall goal of IBN management. It can be seen as an independent component capable of interacting with an SDN controller. The main components of the solution are illustrated in Figure 2.

The workflow starts with the user communicating interactively with the chatbot interface. The chatbot interface serves as the initial touchpoint to establish a communication channel with the user, providing an interface that merely uses natural language. All the inputs and the outputs of the conversation are being progressively registered in a session variable, generating an increasingly bigger amount of information referred to as conversational memory. This information is given to the LLM agent to process it and to ensure that the workflow is indeed being accomplished.

The LLM acts as a secondary layer, handling tasks like complex validation of user inputs, text interpretation and text generation. It also copes with most of the validations of the user’s inputs. The capabilities of the LLM are enhanced with a set of sample utterances designed to provide the LLM with a knowledge base. This will be explained in detail in the following section. The LLM agent provides functionalities like adapting the data structures to make them compatible with teraflowSDN, and it provides a straightforward

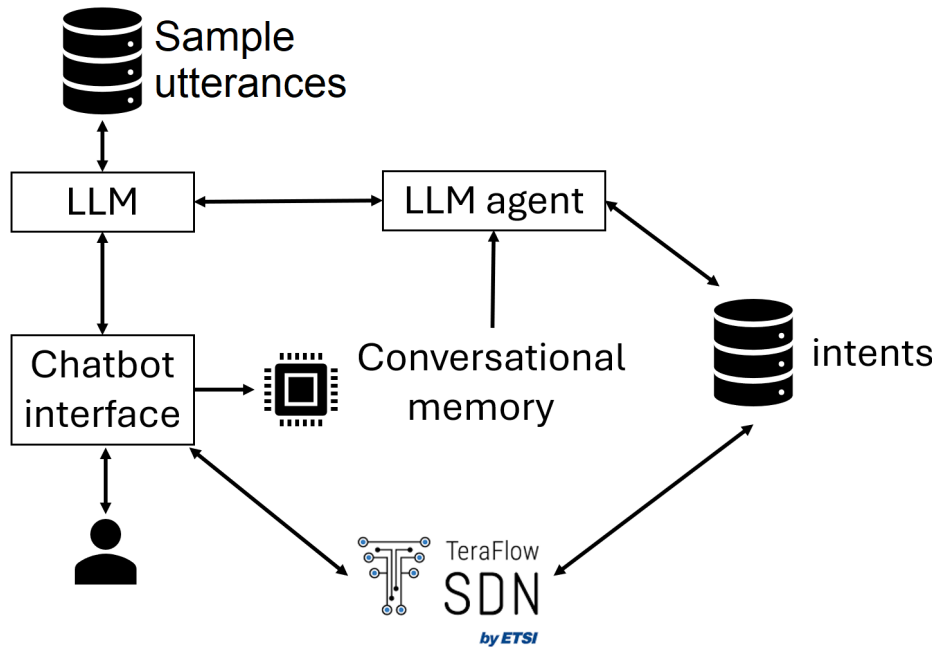


Figure 2: Components architecture of the IBN Generative AI solution.

validation of the user’s inputs. Finally, the intents dataset represented in Figure 2 contains a registry of the generated intents with their corresponding queries for consultancy.

The proposed architecture has been designed to be integrated with TeraFlowSDN. However, its architecture has been designed to be adaptable and to leave the possibility of making it compatible with other SDN controllers in the future. For this purpose, all the components of the architecture would play the same role.

The chatbot interface would interact with the user in the same manner, the LLM would play the same role and would access the same set of sample utterances, and the biggest amount of changes would be in the LLM agent, which would need to ensure that the generated intents fit a different standard. Finally, the generated intent objects would need to be handed over to the new SDN controller to perform the same job as TeraFlowSDN.

3.2. Dataset

The previously presented architecture contains two different datasets with information stored in various formats. One of the datasets includes a set of

sample utterances, with a wide variety of user requests to help the LLM infer the desired operation by the user, whereas the other dataset contains a register of the different generated intents and user queries to make them available in case the user wants to consult them.

3.2.1. Sample utterances

To help the LLM understand the context and infer the desired action of the user, the LLM takes advantage of a set of 240 sample utterances. This dataset is the final result after carrying out a multi-step process, which combines a domain expert validation, the augmentation of paraphrased samples to cope with the problem of linguistic variation and context enrichment.

It is worth remarking that, in some cases, the instances contain words that are considered as “keywords”. For example, given the following input: “Can you describe to me the LOCATIONS of the intent request with the ID 13”. In this case, the user not only indicates that he wants to get an intent described, but he also expresses that he wants to describe only the locations of an intent with a given ID. The words locations and ID, and their respective synonyms and their associated values are particularly remarked for the LLM and by the LLM agent to correctly classify the user request and to better identify the needs of the user.

The small size of the dataset is justified by its uniqueness; each sample utterance was carefully selected, providing different linguistic and contextual variation and ensuring that there are no duplicities. The samples are handpicked with expert knowledge, ensuring high quality and purpose-driven content. The main goal of the generated dataset is to provide efficient knowledge and noise-free information to potentiate accuracy and efficiency.

To avoid bias in classification, the sample sentences were divided equally depending on each action: 80 of them are for creating an intent (Use case described in Sec. 3.4.1), 80 of them are for explaining an intent (described in Sec. 3.4.2) and the remaining 80 are for inspecting the traffic (described in Sec. 3.4.3). It is important to remark that the LLM takes into account the whole dataset to make the classification process, and not only a subset. For clarification, some sample sentences are included below:

- **Sample sentences for creating an intent:** “*can I generate an intent?*”, “*Can I create an intent based on a location?*” and “*I want to describe some requirements and get an intent*”.

- **Sample sentences for explaining an intent:** *“Hello, are you able to explain to me in detail the intents with an id number ...”, “Can you describe to me an intent?”, “Explain to me an intent”.*
- **Sample sentences for inspecting the traffic:** *“Can you retrieve intents from a database?”, “get all the intents”, “I would like to inspect the traffic regarding the historically added intents”.*

After the action of the user has been identified, the text analysis is carried out by two components: The LLM and the LLM agent. The LLM forwards the generated outputs to the LLM agent to retrieve the important input information for each action. The LLM is capable of performing a first layer of validation, and it discards some users’ inputs when the provided reply is not in the desired format or does not contain the required information.

However, some sophisticated validation techniques are addressed by NLP, which is managed by the LLM agent. For example, if the user requests to create an intent with the description of locations, the NLP techniques search for relevant keywords regarding the location such as “origin”, “destination”, “domain”, or “warehouse” to ensure that the user is describing correctly the locations of the expected intent.

3.2.2. Stored intents

All the user requests are stored in a database containing the following attributes:

- **User requests:** Contain the strings included by the user, such as the description of the locations and the bandwidth restriction.
- **Date posted:** A timestamp including the exact date when the user requested that operation.
- **Service Level Agreement reference (SLA):** A reference to the SLA that best matches the user request.
- **Policy reference:** A reference to the policy that matches the user request, some policy examples can be “deploy”, “group”, or “monitor”.

In the case where an intent is deployed in the network, its information is stored in a data structure composed of the following attributes: recognized operations, resources, identified time restrictions and identified locations.

3.3. The workflow of the solution

The proposed solution consists of a linear workflow composed of four sequential steps, as has been previously described in Figure 1. Firstly, the user's requests are being interpreted; secondly, the necessary information for each use case is being extracted and validated; thirdly, the solution requires the user's validation to finally trigger and deploy the necessary network operations. A detailed description of each step is presented below:

- **Task Identification:** The system interprets the user's natural language input to determine the intended network operation. This is a critical step because it sets the foundation for all subsequent actions. The process begins with the user providing a request in natural language, This information is forwarded, in a first step to Amazon lex, and in a second step, to the LLM to analyze the text and identify the desired operation that the user intends to carry out. For this purpose, the LLM leverages a set of sample utterances which were carefully selected to facilitate the task identification. The LLM also harnesses its advanced language understanding capabilities to comprehend the context of the request. This involves parsing the input to detect the user's intent by examining verbs and contextual cues that indicate the desired action.

The LLM is also trained to recognize synonyms and variations in phrasing to ensure robustness against different user expressions. Once the context is established. The presented method classifies the request into one of three predefined operations: creating an intent, explaining an intent, or consulting historical intents. This classification relies heavily on the strong capabilities of the LLM and a set of sample utterances that encompass a wide range of user scenarios and linguistic variations. By accurately identifying the user's intent, the system can proceed to the next step, ensuring that the workflow aligns with the user's objectives.

- **Request and validation of the Key Performance Indicators (KPIs):** Once the task has been identified, the next step is to extract the important information. This step involves an interactive dialogue between the system and the user, facilitated by the chatbot interface. The required information would vary depending on the identified operation. For instance, in the case of intent generation, the user would be

required to introduce information regarding the location parameters (origin and destination), bandwidth requirements, availability levels, and time constraints.

The inputs provided by the user are constantly being registered in a separate session variable and validated to capture all relevant details and ensure that the required information is included and validated correctly. During this interactive query, the system offers guidance to the user by providing sample responses and suggestions, helping to clarify the type of information needed. This approach minimizes user errors and enhances input accuracy. For example, when specifying location requirements, the system might suggest phrases like “From origin A to destination B on port 2000” to guide the user. The system also performs real-time validation of the inputs, ensuring that the data is complete and logically consistent. It checks for valid location identifiers and verifies that bandwidth requirements are feasible given the network’s current state and resources. By validating inputs in real time, the system ensures that the collected KPIs are accurate and actionable, setting the stage for successful intent generation.

- **Confirmation:** After collecting the relevant information, the user verifies the accuracy and completeness of the gathered information before executing the task. Although this step limits somewhat the automation, it is crucial for ensuring that the user’s intent is accurately captured and that any misunderstandings are resolved before proceeding. As a main result, the reliability of the system is increased with a meaningful reduction of critical errors that could otherwise degrade trust in the system. The system compiles the collected KPIs into a concise summary and presents this information to the user through the chatbot interface. The user reviews the summary to confirm that it accurately reflects their request. The system waits for explicit user approval before moving forward, providing an opportunity for the user to make any necessary corrections or adjustments. This confirmation step acts as a safeguard against errors, allowing users to identify and rectify any discrepancies or omissions in the information. If the user detects any issues, they can provide corrections, which the system integrates into the operation. This iterative process ensures that the user’s intent is clearly understood and accurately represented, paving the way for the next steps in the workflow.

- **Trigger Operations:** As a next step, the gathered information is transferred to the LLM agent, which plays the role of triggering the required network operation. In the case where the user requires intent generation, the LLM agent generates the data object based on the user’s restrictions. The data object respects the correct format to connect to the SDNcontroller. This data structure uses the 3rd Generation Partnership Project (3GPP), including attributes such as origin, destination, bandwidth, availability, and time constraints, encapsulating the user’s requirements in a machine-readable format. The system then performs a feasibility check to ensure that the proposed intent can be successfully executed within the network’s current conditions. This involves assessing network capacity, existing loads, and SLA constraints to verify the intent’s viability. By creating a detailed and accurate intent, the system lays the groundwork for successful deployment within the SDN architecture, effectively bridging the gap between user intent and network action. Finally, the approved intent is executed within the network. This step involves applying the necessary configurations and policies to bring the user’s intent to fruition.

Alternatively, when the user requests an intent to be explained. The LLM agent would communicate with the intent database to retrieve the identified intent. This process is enhanced by the RAG. The user is also able to choose to get everything explained or only a part of the intent. Finally, the LLM agent would communicate with the LLM and the chatbot interface to show the requested explanation to the user.

Finally, if the user requests to inspect the available intents, the LLM agent will communicate with the intent database to perform the necessary filters, process the obtained information to make it more user-friendly, and show the information to the user through the chatbot interface.

3.4. Proposed Use Cases

This paper focuses on the following three use cases.

- **Creating an Intent:** When the user aims to generate an intent and deploy it within the network based on a verbal set of requirements.
- **Explaining an Intent:** When the user requests a detailed explanation of an intent using natural language.

- **Consulting Historically Added Intents:** When the user wants to consult a single intent or a group of historically added intents using flexible filters.

3.4.1. Use Case: Creating an Intent

The presented use case describes the set of processes executed by the solution in a scenario where the user wants to generate an intent and deploy it in a network based on a verbal set of requirements.

Task Identification: This process aims to understand the context and to identify the action that the user wants to perform. For that purpose, and as described in section 3.2, the LLM has been trained with a subset of thirty different sentences, including different variations of the same instruction with other structures and synonyms. As a main result, the LLM understands the context and identifies the intended action with robust accuracy.

Request and validation of the KPIs: The interactive chatbot asks the user for location and bandwidth requirements. It also provides sample sentences to facilitate the process. Some sample sentences where they explain the location could be: “*From the origin A on port 2000 until destination B*” or “*on port 2001 in the domain W12*”. Some samples of bandwidth restrictions could be “medium availability and high capacity during 7200 seconds” or “medium bandwidth limitation during 2 hours”. To make the autonomous agent support types of intents, it would be necessary to define a different set of KPIs, but the functionality of the process would occur similarly

Confirmation: The chatbot presents the user with the already gathered information, and it waits for its approval.

Trigger Operations: Firstly, the presented method implements different NLP techniques to translate the user request into numerical information predefined by the network operator. For example, “medium bandwidth limitation” would be interpreted as 20 Gbps or a “low availability” would be considered as 30%. Secondly, the algorithm constructs the data structure in JSON format based on the extracted information, and it shows it to the user to wait for its approval to finally deploy the intent in the network. To make the solution support other types of intents, this step would need to provide the same functionality, but in a different manner, generating different hierarchical objects and taking into account different KPIs. The result of this step would be the same: the generation of a data structure and the approval of the user.

3.4.2. Use Case: Explaining an Intent

The presented use case describes the processes and actions taken when the user wants the chatbot to describe an intent using natural language.

Task identification: To identify what the user wants, the LLM model was trained with a set of 80 different sentences containing different variations of the same request. Although this is explained in detail in Sec. 3.2, for clarification, we can mention that some of these sentences could be: “Can you explain to me what’s inside a network request?” or “Describe to me an intent”.

Request and validation of the KPIs: The chatbot asks in real time the user for which intent and which part of the intent needs an explanation, distinguishing three options: If the user only wants to explain the location contents of the intent, if the user only wants to get explained the network restrictions such as bandwidth, availability and time restrictions or if the user wants all the contents of the intent to be explained to him.

Confirmation: For robustness purposes, the provided solution gathers the included information for the user and waits for its confirmation.

Trigger Operations: Firstly, the LLM agent communicates with the intent database to retrieve the corresponding intent information. Secondly, the LLM agent processes the information stored in the intent and extracts relevant information: In the case of the location attributes, the method extracts the following attributes: The unique universal global identifier (UUID) of the intent, the switch and the port where it is placed for all the location points.

In the case of the other constraint values, the following attributes are being extracted: constraint values in terms of required availability, requested capacity, bandwidth constraints and time restrictions.

As a main result, the most important information is extracted and processed to finally write the intent explanation in natural language. To compose the final message, a big subset of templates containing different ways of saying the same message will be used to create the feeling of a more organic language inside the LLM agent, and this information is forwarded to the LLM.

3.4.3. Use Case: Consulting historical intents

The user can also request the chatbot to consult a single intent or a group of them by applying flexible filters. To achieve that, the system would need to go through the following processes:

Task identification: The LLM understands context based on a subset of 80 sample sentences that permits building the underpinning for task identification whilst being flexible in the language used. Some examples of these sentences are: “Can I retrieve the information from the intent X?” or “Can I retrieve all those intents with low availability and medium capacity?”

Request and validation of the KPIs: The chatbot interactively asks if the user wants to apply a filter and retrieve more than one intent. In that case, the user has to specify the attribute name to filter and its value. The user can also request a specific ID number to receive a single object.

Confirmation: The gathered information is also shown to the user for his confirmation.

Trigger Operations: The LLM agent connects with the intent database, triggers the necessary query, retrieves the necessary information and processes it to make it more user-friendly. The results are shown to the user as a reply from the chatbot.

4. Experimental evaluation

4.1. Implementation details

Regarding the architecture implementation, it is worth mentioning that the chatbot interface is constructed using Amazon Lex version 2, which serves as the user interface for human-machine interaction. Amazon Lex is an interactive chatbot that allows users to input commands and receive responses in natural language, enabling intuitive communication with the system. This component directly interfaces with the Web UI within the TeraFlowSDN architecture, allowing users to initiate network operations seamlessly. By leveraging natural language understanding, Amazon Lex interprets user intents accurately, facilitating streamlined management of network configurations and operations.

The chatbot interface integrates Amazon Bedrock, with a ready-to-use LLM which has been trained with a huge amount of data. However, to enhance the accuracy of the LLM in this particular scenario, a dataset of sample utterances has been included.

The LLM agent is developed using Python version 3.11, and it runs on Amazon Lambda to trigger events and implement the logic coded in Python, whereas the information would be stored in two different DynamoDB [15] datasets.

The two different datasets, which have been described in the architecture section 3.1, are in DynamoDB version 2019.11.21 [15], a NoSQL database that is responsible for maintaining up-to-date information on network configurations, conditions, and existing intents. The use of a NoSQL database like DynamoDB offers scalability and flexibility, making it well-suited for handling the large volumes of data typically associated with network management.

Although the solution is integrated into the TeraFlowSDN architecture, it is designed to be adaptable and can be modified to work with other Software-Defined Networking (SDN) controllers. This flexibility ensures that the system can be deployed across various network environments, offering a versatile solution for addressing network management challenges.

4.2. Implementation of RAG.

The RAG techniques significantly improve the performance and the accuracy of the LLM by allowing it to leverage external domain-specific knowledge. In this particular scenario, the RAG mechanism amplifies the capabilities of the LLM by retrieving relevant information from the set of sample utterances and the conversational memory. This retrieved information is dynamically injected into the input information of the LLM, allowing it to generate responses that are more aligned with the user’s needs.

The set of components and their interactions are illustrated in Figure 3. The user triggers the workflow by directly communicating with the Chat UI. In a first layer, Amazon Lex applies NLP techniques to enhance the communicational system, whereas the LLM carries out more complex text processing tasks. The LLM is enriched by both the conversational memory accumulated throughout the session and the retrieval of relevant data instances. The Amazon Lambda function, corresponding to the “LLM agent” described in Section 3.2, provides an additional validation and interpretation of the user’s inputs with the integration of NLP techniques. For example, if the user intended to create an intent with “low bandwidth requirements”, this information would be converted into “50 Mbps”.

Hence, the RAG permits the system to get a combination of static language generation capabilities with dynamic information which results in an improvement of contextual awareness, efficiency and higher accuracy.

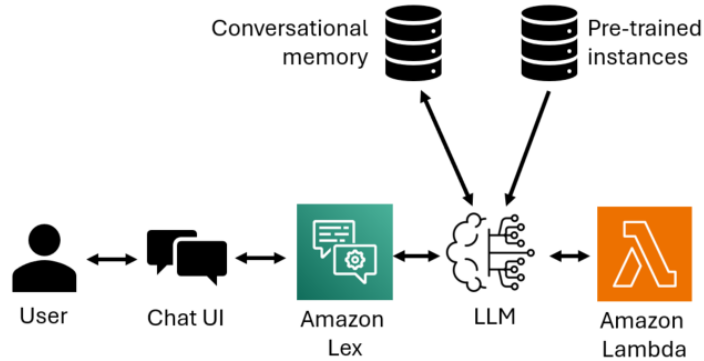


Figure 3: Description of the workflow and the different components to carry out the RAG techniques.

4.3. Security considerations.

The suggested paper presents a proof of concept for an autonomous agent working in a real-world scenario. It has been acknowledged that LLMs are vulnerable to adversarially crafted inputs designed to bypass safety mechanisms. To mitigate this threat, the following security actions have been taken.

First of all, the chatbot interface would always await the confirmation of the user to the proposed system before triggering the network operation. This action is helpful not only to ensure the adequacy of the intents but also to mitigate some malicious attacks.

Second of all, the used TeraflowSDN controller ensures that the generated intent requests comply with one of the available Service Layer Agreement (SLA) policies and the intent passes through a set of verifications before being deployed on the network.

4.4. Use case examples

Even though there are a large number of potential use cases for implementing LLM into intent management. The presented solution focuses on three main use cases corresponding to common operations that occur rarely in network configuration. These use cases are explained through walkthrough examples to demonstrate the system's capabilities in creating, explaining, and querying intents. Each example includes the inputs provided by the

user, the responses received by the chatbot, and the operations carried out in the network.

4.4.1. Use Case: Creating an Intent

Firstly, an example presuming that the user wants to create an intent is being provided, presuming that the user included the following input: *“Can I create an intent based on a description?”* The chatbot was able to understand the context and offer two messages as an automatic reply:

“I can generate an intent based on a set of requirements using natural language!”, *“Can you tell me the action of your intent? create, delete or inspect”*.

Right after understanding the context, the interactive chatbot asks for the relevant information to fulfil the request. The identified action in this example would be *“create”*.

The chatbot also asks the user to include the location requirements for the intent and provides the user with a sample response with a text like this: *“Can you tell me the location requirements?”* For example: *“From origin A to destination B on port 2000”*. The user would then indicate the following information regarding the location requirements: *“from origin C to destination D”*.

Finally, the chatbot asks for other bandwidth requirements: *“Can you tell me the restriction requirements?”* For example: *“medium availability, high capacity during 7200 seconds”* receiving the following response from the user: *“with medium availability, low capacity and medium bandwidth requirements during 2 hours”*.

After receiving the confirmation, the chatbot triggers NLP functionalities to construct the intent based on the user’s specifications. The interpreted information in this particular example has been summarized in Figure 4.

Figure 4 shows that the user indicated that it wants medium availability, which was translated into 70% of availability. It has also requested low capacity, which has been interpreted as 40 Gbps.

The algorithm has automatically matched these requirements with two different internal SLAs. In this case, the algorithm would select the most restrictive SLA to fulfil all requirements.

In Figure 4, it can also be appreciated that two locations were identified where the origin would be C and the destination D. In this particular case, since no ports were specified, it would automatically assign them. The user has also requested a lifetime of 2 hours, which has been converted to 7200

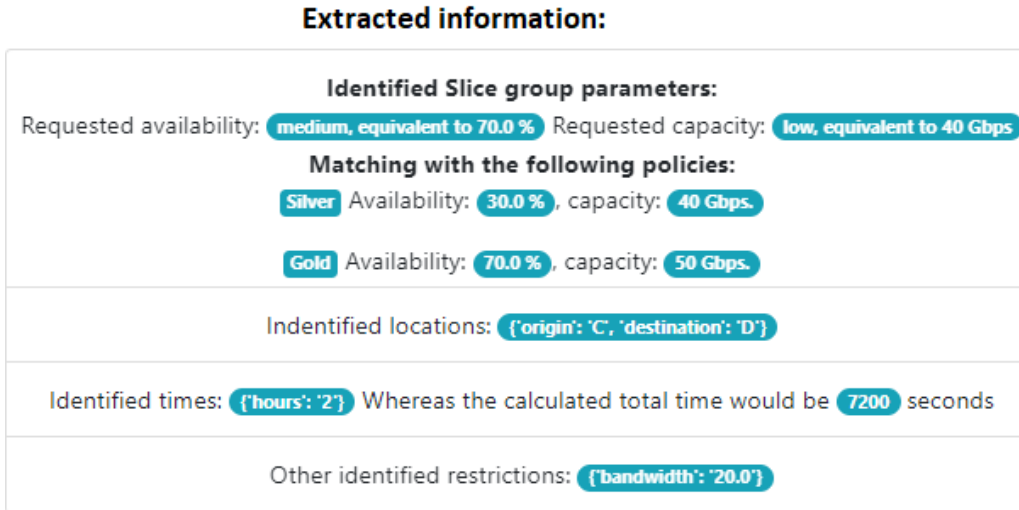


Figure 4: Extracted information to generate an intent based on the information included by the user.

seconds and medium bandwidth requirements, which, in this case, has been converted to 20 Gbps.

Finally, the developed solution would take this extracted information as input to construct the 3GPP data structure of the intent, and it would perform a simplified feasibility check to ensure that it can be successfully deployed into the SDN architecture.

4.4.2. Use Case: Explaining an Intent

For clarification, in this use case, it has been presumed that the input would be the generated intent in the previous case with the same characteristics and as an output. The user would expect to receive a description in natural language based on the contents of the intent.

The use case starts with the following input from the user: *“Can I get the description of an intent?”*. The chatbot was capable of understanding the context and inferring what the user wanted. The chatbot also asks for the identifier of the intent in the databases and the part of the intent that has to be explained, leaving three options: The locations, the network limitations and both of them.

“Can you provide me the intent ID?” where the user replied with the correct number and received the following response requesting the second

KPI: *“Can you tell me which part of the intent you would like me to explain? Locations, network limitations or all.”*

In this case, the user would request to explain all the contents of the intent, and the interactive chatbot would request a confirmation with the following response: *“Ok. I am going to explain to you the contents of the intent with the number XX. Is that correct?”*

Generated description:

There are 2 indicated locations points on the specified request: First, The location labeled as C is placed on the port 2/5. Second, The location labeled as D is placed on the port 2/35. There are 2 limitations on the specified request: First, The request will be valid during 7200 seconds which is 2:00:00. Second, A medium quality bandwidth limitation is included (20.0 Mbps.)

Figure 5: Chatbot response when the user wants to get an intent explained.

After acceptance, the LLM communicates with the LLM agent to implement different NLP processing techniques, generating text based on a set of predefined templates to generate the feeling of a more organic language. These results will be forwarded to the LLM and shown to the user through the chatbot interface. As the main result, the chatbot presented the response gathered in Figure 5. Where it is possible to appreciate that the chatbot found two different locations that were automatically placed on different ports, regarding the medium restrictions, it did not merely mention its values stored in the database but also interpreted them. For example, it specified 7200 seconds, which corresponds to 2 hours or a medium bandwidth restriction.

4.4.3. Use Case: Consulting historical intents

The inputs of the conversation in the case where the user wants to consult the existing intents have been gathered in Figure 6, where it is possible to see that the chatbot identified the desired action for the user to later ask for the relevant information in this specific operation.

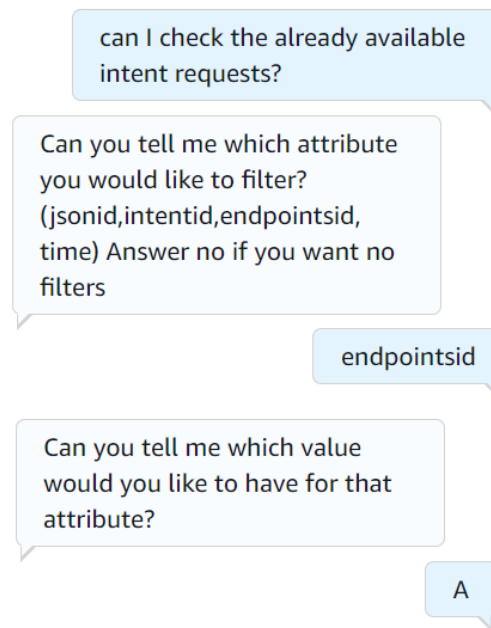


Figure 6: Chatbot response when the user wants to get an intent explained.

After the input parameters have been included and validated, this use case implements RAG. The LLM forwards the information to the LLM agent, which communicates directly with the database to retrieve the information from the intent database. This information will be filtered to retrieve only the required information by the user and processed to make it more user-friendly.

In this particular case, the user wants to retrieve all the intents that include location A and hence, the algorithm would reply with a list of intents that refer to location A, regardless of whether they are from the origin or destination. However, it is also important to point out that the user is also able to retrieve a single intent by specifying the attribute “intentid” and providing the corresponding number identifier. For clarification, an example response from the chatbot containing only one intent in the A location is

provided in 7.

```
{'jsonid': Decimal('1'),
'jsonObject': {'services':
[{'service_id': {'context_id':
{'context_uuid': {'uuid':
"admin"}}, "service_uuid":
{'uuid': "intent-1"}},
"service_type": 1,
"service_status":
{'service_status': 1},
"service_endpoint_ids":
[{'device_id': {'device_uuid':
{'uuid': "A"}}, "endpoint_uuid":
{'uuid': "2/5"}]},
"service_constraints":
[{'custom': {"constraint_type":
"time[sec]", "constraint_value":
"9000"}]}, "service_config":
{'config_rules': [{"action": 1,
"custom": {"resource_key":
"/settings", "resource_value":
{}}]}]}
```

Figure 7: Chatbot response when the user wants to get an intent explained.

4.5. Accuracy results

The presented walkthrough examples illustrate the behaviour and steps the solution takes under typical circumstances, demonstrating its functionality across various scenarios. The results indicate the LLM's robustness in understanding user intent and performing accurate operations.

To assess the efficacy of the LLM, 180 sample cases were generated; 60 of them would be intended for the create operation, 60 of them should trigger the explain operation, and finally, the remaining 60 should trigger the query intents operation. In each case, the intent JSON-generated object provided

by the LLM will be compared with the JSON object with the expected information. This concept of validity is known as factual accuracy.

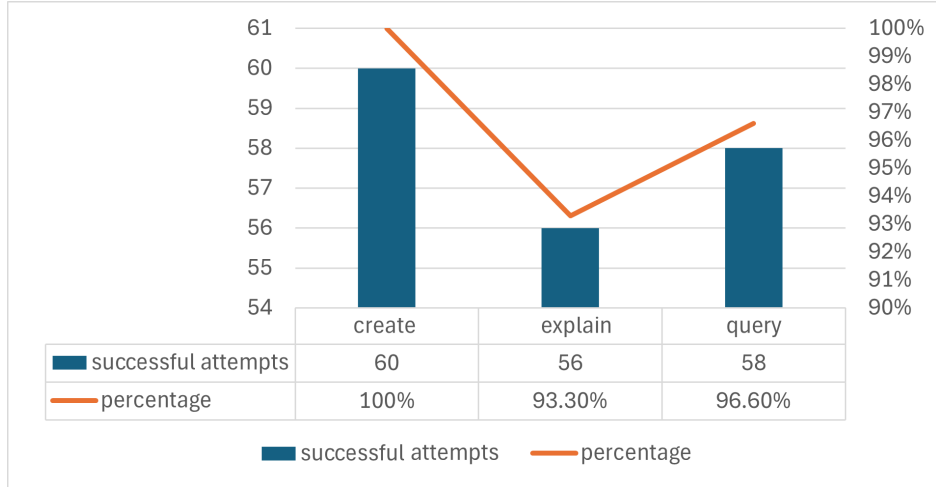


Figure 8: Success rates of the LLM identifying and triggering operations.

The breakdown results for factual accuracy have been gathered in Figure 8. The Figure shows that all the operations obtained a robust success rate. The create operation reached 100% of factual accuracy, being successful 60 out of 60 times. Otherwise, the LLM was able to correctly identify the explained operation 56 out of 60 times, achieving an accuracy of 93.30%. That is because the LLM misinterpreted the explain operation with the create operation since the user can use some words in common with the other operation, like “intent” or “JSON object”. However, with a high success rate in all the operations, it is possible to ensure that the LLM performed very robustly by correctly identifying this operation in the vast majority of the cases. Finally, the query operation was successfully identified 58 times out of 30, achieving 96.6% of accuracy.

To complement the factual accuracy results, the confusion matrix for all the operations has been extracted, and the results are commented on. For instance, the factual accuracy results indicate a 100% of factual accuracy for the create operation, but Table 1 indicates that there were 5 instances with false positives, which means that the LLM considered a user request as a create operation 5 times and it was not in that category, which suggests that perhaps the classification mechanism of the LLM is slightly biased towards the create operation. This corresponds to a precision of 0.92 and a recall of

1, resulting in an F1 score of 0.48

Table 1: Confusion matrix for the create operation

| Real Values — Predicted values | True | False |
|--------------------------------|------|-------|
| True | 60 | 5 |
| False | 0 | 0 |

However, the confusion matrix from Table 2 indicates that there were 56 instances correctly classified, whereas there were four user requests that were wrongly classified. For this operation, there was only 1 case where the LLM wrongly considered a query operation, which shows some strengths regarding the false positives. This implies a precision of 0.98 and a recall of 1, which results in an F1 score of 0.49.

Table 2: Confusion matrix for the query operation

| Real Values — Predicted values | True | False |
|--------------------------------|------|-------|
| True | 56 | 1 |
| False | 0 | 4 |

Finally, the confusion matrix from Table 3 indicates that 58 instances were classified correctly, whereas the remaining two were wrongly classified; the absence of false positives in this operation reassures the fact that the LLM was slightly biased towards the create operation. These results imply a precision of 1 and a recall of 1, which results in an F1 score of 0.5.

Table 3: Confusion matrix for the query operation

| Real Values — Predicted values | True | False |
|--------------------------------|------|-------|
| True | 58 | 0 |
| False | 0 | 2 |

To clarify this aspect, a case where the LLM misled the desired operation by the user is provided in Figure 9, with the input “Can I get explained an already generated intent?”. The user has used important keywords like “generated” and “intent” which made the LLM believe that the user wanted to generate an intent. However, in this particular case, what the user wants is to get an intent explained.

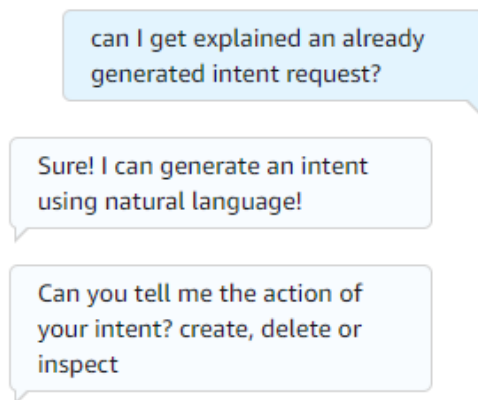


Figure 9: A sample case where the LLM misinterpreted the desired operation by the user.

Regarding the usage of the NLP techniques. It is difficult to infer whether the user wanted to generate an intent with those specific values or to receive certain information from that intent. The inner way of working on the NLP techniques consists of keyword identification with one-hot encoding. The identified values are automatically converted into numeric values. For instance, “low bandwidth” is identified as 40 Gbps.

In this case, the way to cope with these misinterpretations and ensure 100% accuracy is to show the user the results before the operation is triggered in the network. For example, the algorithm would show the intent generated based on the user’s information and await its confirmation before sending that intent to the network.

4.6. Comparison with other LLMs

To complement the results for factual accuracy, the same scenario has been recreated with the same sample of 180 different instances equally distributed among the three types of actions. Three different options were considered: The suggested LLM implementing AWS suite (integrating RAG capabilities in the same manner as the previous experiment) and the latest version of ChatGPT with and without the implementation of the RAG capabilities. For the sake of this experiment, the LLMs were only instructed to identify the desired operation by the user with the main aim of assessing the differences between different LLMs and the impact of the implemented RAG capabilities.

Figure 10 summarizes the number of successful attempts for each of the operations, whereas the create operation managed to acquire a 100% of accuracy with all the options, the explain operation seems to be a bit more challenging for all the LLM, the usage of ChatGPT 4.0 without RAG reached only 54 successful classifications out of 60, the remaining 60 instances were wrongly classified as a query. In contrast, the Amazon bedrock classified correctly 56 out of 60 instances. Finally, the usage of ChatGPT implementing the RAG capabilities seems to outperform its comparables, with only one misclassified instance out of 60. Finally, the query operation seems to be only a bit misleading for Amazon Bedrock, classifying correctly only 58 out of 60 instances.

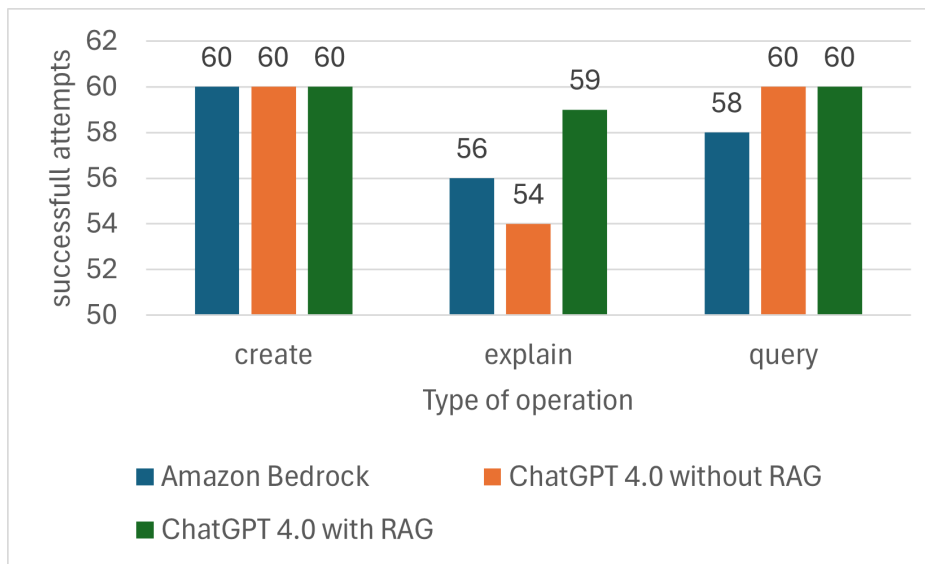


Figure 10: Successful attempts when implementing the AWS suite and ChatGPT with and without RAG capabilities.

The aforementioned results contribute to quantifying and demonstrating the contribution of the RAG, increasing the accuracy by 8.3% and to assess and quantify the impact of implementing different LLM algorithms.

4.7. Execution time results

The efficiency of the solution has also been assessed. The method has been tested 24 times, having 8 samples for each of the use cases, and the

time taken to carry out the different processes has been measured, providing the following results.

Within the use case for generating a new intent, the following four processes have been distinguished: The time taken for the LLM to identify the desired operation by the user, the total time taken to request and validate the KPIs, the time taking constructing the intent and performing a simplistic validation and feasibility check and the time taking sending the request to the network.

The time results have been gathered in Figures 11 and 12, where the Y axis represents the time taken in seconds and the X axis expresses the CDF values of each operation. The graphs have been broken down into two different figures for clarification, since the scale of the processes varies greatly between them.

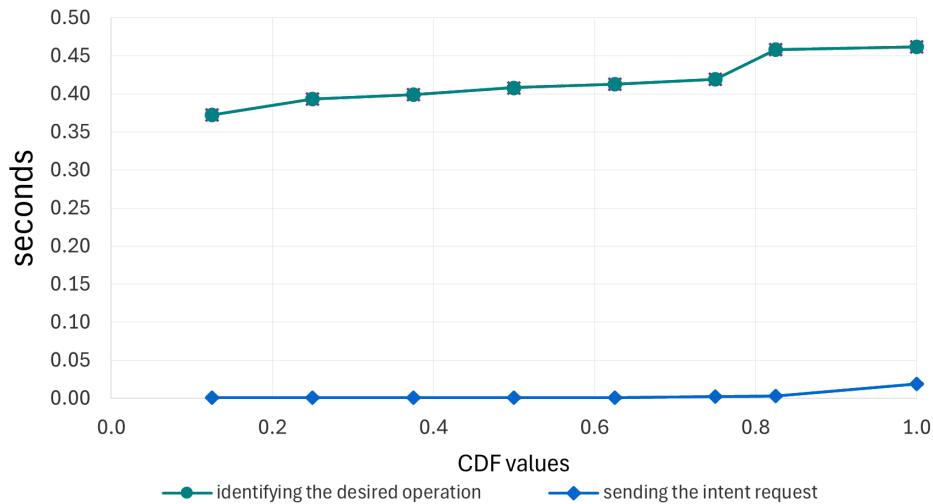


Figure 11: Cumulative distribution values for identifying an operation and sending a request.

Figure 11 shows that the algorithm took a very short time to identify the desired operation by the user, with all the results taking between 0.37 and 0.46 seconds. Regarding the process of constructing the intent represented in 12, it is possible to see a clear difference between the samples. That is because the algorithm has to take more time generating the intent in those cases where more restrictions enumerated by the user result in more child objects appended to the intent object. Additionally, the feasibility check for

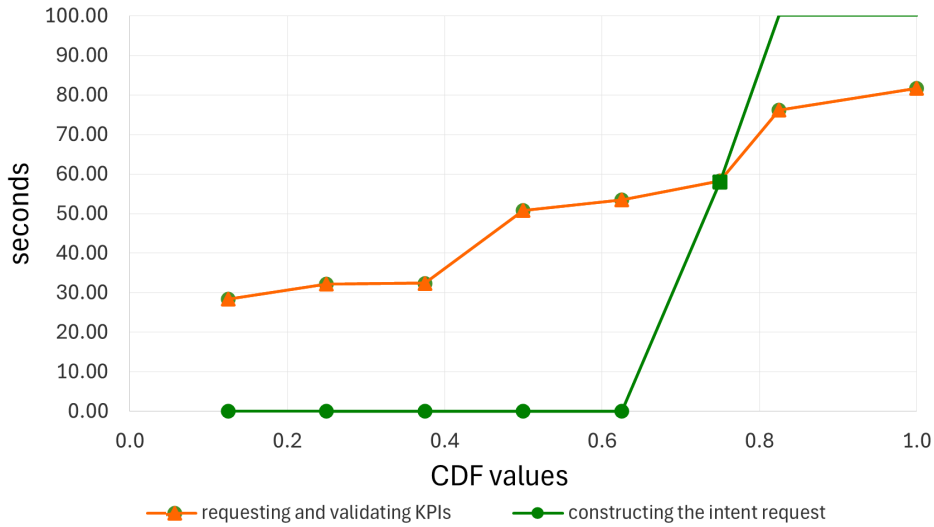


Figure 12: Cumulative distribution values for validating KPIs and constructing the request.

ensuring that the intent can be deployed in the network might have different performances depending on the traffic and the type of intent.

Finally, Figure 11 also shows the time taken for the E2E process when requesting and validating the KPIs, which varies from 30 to 80 seconds. That is because this process relies on the user’s input, which might take longer or not, depending on the situation. Finally, the Figure also shows the very short time taken to deploy the intent in the network with a set of values ranging from 0.001 to 0.012.

Within the use case for searching between the existing intents, four different processes can be distinguished: To identify the desired operation, to request and validate the required KPIs, to carry out the DB search and to send the request to show the results to the user.

Figure 14 shows the number of seconds (sec) on the y-axis and the ordered samples on the x-axis. The algorithm was able to identify the desired operation by the user in a very short time, i.e., in a range of 0.37 and 0.42 sec, the request and the validation of the KPIs shown in Figure 13 are slightly shorter for this use case since less information is required by the user, with the times ranging from 9 until 63 sec.

Alternatively, Figure 13 shows that the process for performing the database search turned out to be pretty stable and fast, although it is not possible to

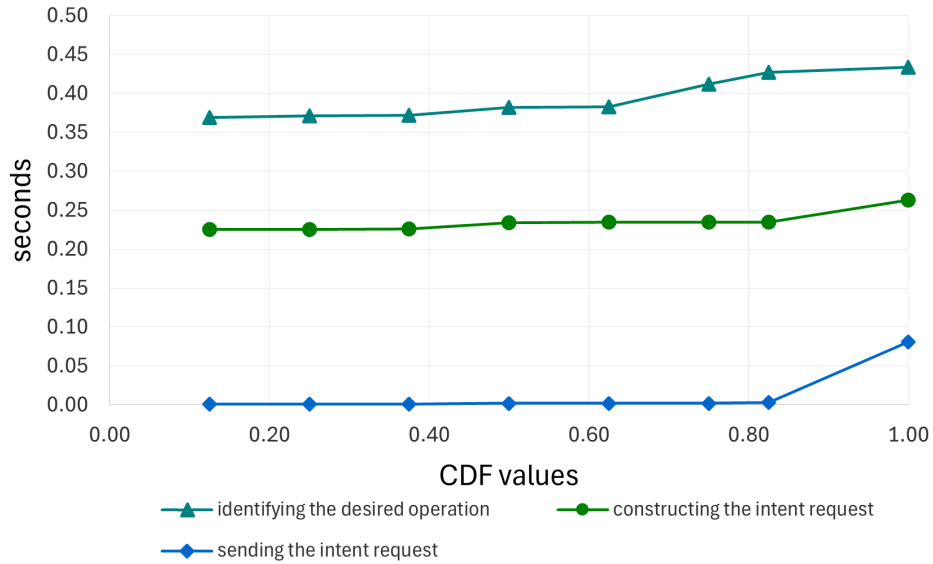


Figure 13: Cumulative distribution values for the time taken on three processes checking the available intents.

appreciate it very well in the graph, the times were distributed between a minimum of 0.225 Sec. and a maximum of 0.263 Sec. Finally, sending the request to the user was done within 0.001 and 0.081 Sec.

Finally, the time for four different tasks has been measured for the case of getting an intent explained: a) To identify the correct operation by the LLM, b) request and the validation of the KPIs, c) to carry out the database search and d) to implement NLP to write the reply to the user.

The CDF values for the time taken on the different processes have been gathered in Figure 15 and Figure 16. The process for the request and validation of the KPIs was done in a range of 17 and 35 sec., since not too much information was required by the user. Whereas identifying the correct operation represented in Figure 16 ranges from 0.31 to 0.43 sec.

Finally, the database search obtained similar results to the previous use case but with slightly lower values, ranging from a minimum of 0.255 to 0.263 sec. As expected, the results for the user were carried out in a short period, varying from 0.001 to 0.081 sec.

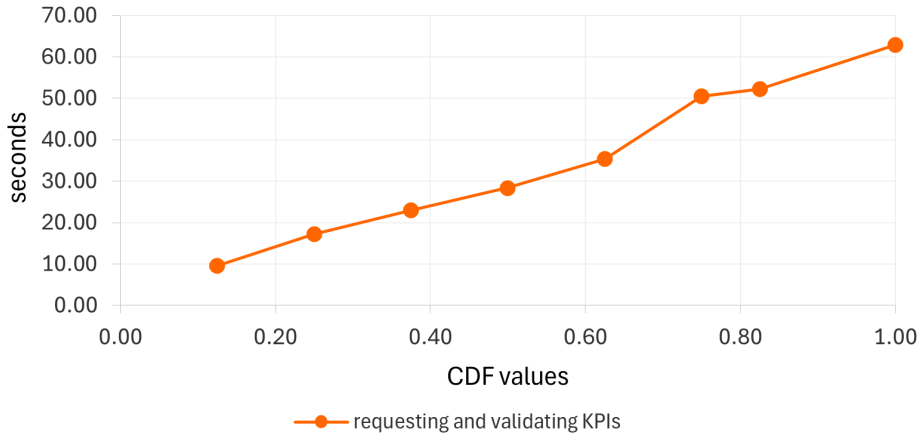


Figure 14: Cumulative distribution values for the time taken requesting the KPIs.

4.8. Comparison with other studies

To establish a comparison between the results shown in the study and its peers, it is worth considering that the context presents a significant innovation, meaning that few studies offer direct comparisons with the presented work. Additionally, it is worth considering that the LLMs are implemented in different scenarios and receive inputs of different kinds. Hence, it is necessary to be cautious when comparing results. However, it is equally fair to say that the presented LLM overperformed its comparables, obtaining an accuracy of at least 93.3% when identifying the desired operations by the user.

The following study [11] presented a LLM that has been assessed depending on the understanding of the context and its factual accuracy, which means to check whether the LLM understood the context appropriately and provided the desired outputs, reaching a factual accuracy of 82% to 86%. Alternatively, other papers on the same topic assessed LLMs’ understanding of the “long-term context”, reporting only 59% accuracy [3].

5. Conclusion

LLMs have proven to be a powerful tool to simplify complex network operations. The intent explanation process that this paper showcased is just a sample of the potential capabilities that the LLM have to make the process of network configuration more understandable. This method harnesses the

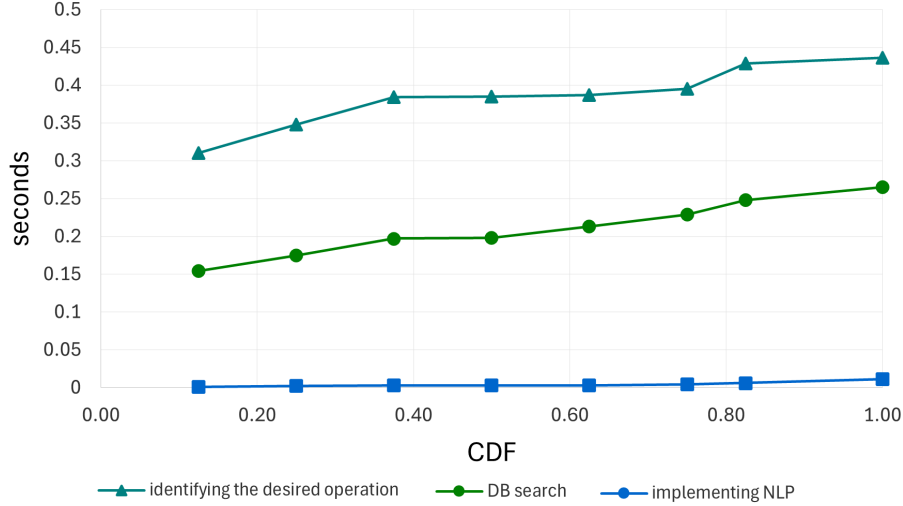


Figure 15: Cumulative distribution values of the time taken on three tasks in the use case for the intent explanation.

combination of three different technologies, which are LLM, NLP techniques, and data science, that are assembled to create a fully automated system.

The LLM agent is flexible enough to understand the context, infer the desired operation by the user and act accordingly. Additionally, using a chatbot provides a user-friendly interface for network operations, which permits interaction with the system by merely using natural language. That simplifies and makes the process of network configuration more understandable. Finally, leveraging AI to interpret and execute network intents reduces human iteration and hence decreases the response time to network changes and demands by increasing automation. This leads to a more dynamic and adaptable network infrastructure.

Implications: This study extends beyond merely using the LLM model as a question-and-answer system, fully integrating its capabilities into a broader system to facilitate network configuration. The solution directly communicates with the SDN architecture, allowing it to understand user intent and autonomously deploy operations after receiving the user’s confirmation. This highlights the potential applications of LLMs in network configuration, showcasing their transformative capabilities.

Limitations: While the results show that the LLM can understand con-

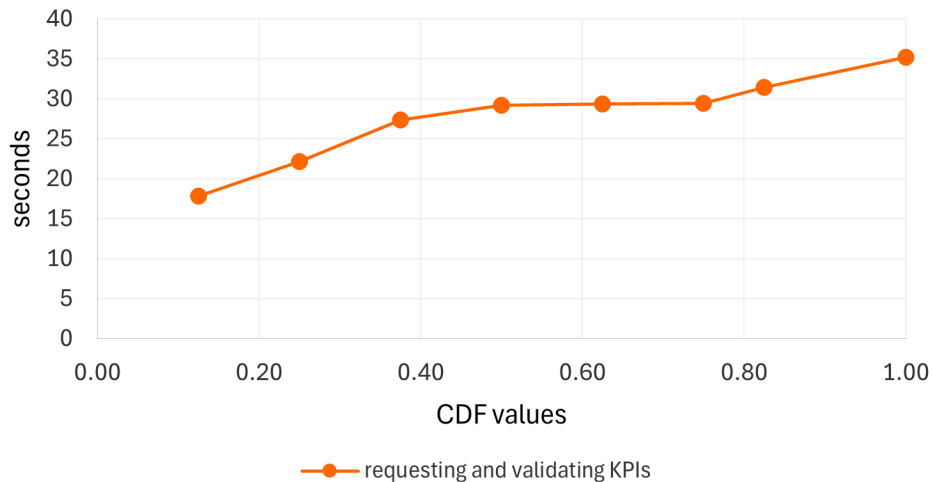


Figure 16: Cumulative distribution values for the time taken requesting and validating the KPIs.

text and trigger user-desired operations, it’s important to recognize that the LLM is currently programmed for three specific actions. This limitation serves as a starting point for aiding network configurations, but there are numerous other potential use cases where this technology could further enhance network management.

Alternatively, the validation provided by the user greatly contributes to enhancing the quality of the inputs to the system. However, it also limits true automation.

Finally, it is worth mentioning that the system automatically converts the user’s inputs into a number. For instance, low bandwidth requirements are converted to 20 Gbps, which boosts automation and simplifies the configuration task for the user, but also limits flexibility in the conversion process.

Future work: Future improvements could focus on increasing the LLM’s flexibility in identifying context and performing various operations related to network configuration. The inclusion of broader functionalities (e.g., deletion, update, anomaly detection) could be implemented easily. The LLM would need to be fed with new cases that identify these operations and that would trigger the new functionalities coded in a similar manner to use cases already included. Scalability analysis is also an important aspect that should be addressed in future work, since the LLMs consume nowadays a big amount of resources to promote the efficiency of the system and carry out a scala-

bility analysis would be a relevant work line. Additionally, integrating novel techniques such as autonomous agents could complement LLM functionality, enabling greater automation and task completeness. Another potential enhancement could be the elimination of the user validation to provide complete automation of the system. Although the user validation acts as a filter, this scope could meaningfully reduce the computational times in the other steps. Finally, another future work could be to carry out a systematic test against adversarial prompts and assess the effectiveness of the intent verification methods or in ensuring a robust authentication authorization, and role-based access control mechanisms to prevent unauthorized use of the suggested agent to ensure it is not being used by a malicious actor.

Acknowledgements

This work was partially supported by the Horizon Europe Hexa-X-II (101095759); Ministerio de Asuntos Económicos y Transformación Digital and the European Union-Next GenerationEU in the frameworks of the Plan de Recuperación, Transformación y Resiliencia and of the Mecanismo de Recuperación y Resiliencia through UNICO-5G I+D 6GMICROSDN project under references TSI-063000-2021-19, TSI-063000-2021-20, TSI-063000-2021-21; MCIN/AEI/10.13039/501100011033/FEDER/UE and by ERDF A way of making Europe through the Spanish.

References

- [1] Khizar Abbas, Talha Ahmed Khan, Muhammad Afaq, and Wang-Cheol Song. Network slice lifecycle management for 5g mobile networks: An intent-based networking approach. *IEEE Access*, 9:80128–80146, 2021.
- [2] Daniel Adanza, Carlos Natalino, Lluís Gifre, Raul Muñoz, Pol Alemany, Paolo Monti, and Ricard Vilalta. Intentllm: An ai chatbot to create, find, and explain slice intents in teraflowsdn, 2024.
- [3] Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, and Jian-Guang Lou. Make your llm fully utilize the context, 2024.
- [4] Zied Ben Houidi and Dario Rossi. Neural language models for network configuration: Opportunities and reality check. *Computer Communications*, 193:118–125, 2022.

- [5] Andrei-Laurentiu Bornea, Fadhel Ayed, Antonio De Domenico, Nicola Piovesan, and Ali Maatouk. Telco-rag: Navigating the challenges of retrieval-augmented language models for telecommunications, 2024.
- [6] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6491–6501, New York, NY, USA, 2024. Association for Computing Machinery.
- [7] Krishna Prakash Kalyanathaya, D Akila, and P Rajesh. Advances in natural language processing—a survey of current research trends, development tools and industry applications. *International Journal of Recent Technology and Engineering*, 7(5C):199–202, 2019.
- [8] Athanasios Karapantelakis, Mukesh Thakur, Alexandros Nikou, Farnaz Moradi, Christian Orlog, Fitsum Gaim, Henrik Holm, Doumitrou Daniil Nimara, and Vincent Huang. Using large language models to understand telecom standards, 2024.
- [9] Junyi Li et al. Pretrained language models for text generation: A survey. *arXiv preprint arXiv:2201.05273*, 2022.
- [10] NOur EL Marref. *Talking Taboo Topics with ChatGPT AI: Bais/Sentiment Analysis study/Personality Analysis*. PhD thesis, 2023.
- [11] Tsukihito Komanaka et al. Miyu Sasaki, Natsumi Watanabe. Enhancing contextual understanding of mistral llm with external knowledge bases. *PREPRINT (Version 1) Research Square*, 2024.
- [12] Manh-Tien-Anh Nguyen, Sondes Bannour Souihi, Hai-Anh Tran, and Sami Souihi. When nlp meets sdn: an application to global internet exchange network. In *ICC 2022 - IEEE International Conference on Communications*, pages 2972–2977, 2022.
- [13] Mohammad Riftadi and Fernando Kuipers. P4i/o: Intent-based networking with p4. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 438–443, 2019.

- [14] Sujoy Roychowdhury, Sumit Soman, H G Ranjani, Neeraj Gunda, Vansh Chhabra, and Sai Krishna Bala. Evaluation of rag metrics for question answering in the telecom domain, 2024.
- [15] Swaminathan Sivasubramanian. Amazon dynamodb: a seamlessly scalable non-relational database service. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, page 729–730, New York, NY, USA, 2012. Association for Computing Machinery.
- [16] Luis Velasco, Marco Signorelli, Oscar González De Dios, Chrysa Papianni, Roberto Bifulco, Juan Jose Vegas Olmos, Simon Pryor, Gino Carrozzo, Julius Schulz-Zander, Mehdi Bennis, Ricardo Martinez, Filippo Cugini, Claudio Salvadori, Vincent Lefebvre, Luca Valcarengi, and Marc Ruiz. End-to-end intent-based networking. *IEEE Communications Magazine*, 59(10):106–112, 2021.
- [17] Ricard Vilalta et al. Demonstration of intent-based networking for end-to-end packet optical cloud-native SDN orchestration. In *European Conference on Optical Communication (ECOC)*, page D3, 2023.
- [18] Ricard Vilalta, Raul Munoz, Ramon Casellas, Ricardo Martínez, Victor López, Oscar González de Dios, Antonio Pastor, Georgios P Katsikas, Felix Klaedtke, Paolo Monti, et al. Teraflow: Secured autonomic traffic management for a tera sdn flows. In *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pages 377–382. IEEE, 2021.
- [19] Ricard Vilalta, Raul Muñoz, Ramon Casellas, Ricardo Martínez, Victor López, Oscar González de Dios, Antonio Pastor, Georgios P. Katsikas, Felix Klaedtke, Paolo Monti, Alberto Mozo, Thomas Zinner, Harald Øverby, Sergio Gonzalez-Diaz, Håkon Lønsethagen, José-Miguel Pulido, and Daniel King. Teraflow: Secured autonomic traffic management for a tera of sdn flows. In *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pages 377–382, 2021.
- [20] Changjie Wang, Mariano Scazzariello, Alireza Farshin, Dejan Kostic, and Marco Chiesa. Making network configuration human friendly, 2023.

- [21] Yidi Wang et al. AlarmGPT: An intelligent operation assistant for optical network alarm analysis using ChatGPT. In *European Conference on Optical Communication (ECOC)*, page Th.A.1.2, 2023.
- [22] Yiming Wei, Mugen Peng, and Yaqiong Liu. Intent-based networks for 6g: Insights and challenges. *Digital Communications and Networks*, 6(3):270–280, 2020.
- [23] Haoyu Xu et al. Monolingual denoising with large language models for low-resource machine translation. In Fei Liu et al., editors, *Natural Language Processing and Chinese Computing*, pages 413–425. Springer, 2023.
- [24] Jiuyang Zhao et al. OpticalBERT and OpticalTable-SQA: Text- and table-based language models for the optical-materials domain. *Journal of Chemical Information and Modeling*, 63(7):1961–1981, 2023. PMID: 36940385.
- [25] Hao Zhou, Chengming Hu, Ye Yuan, Yufei Cui, Yili Jin, Can Chen, Haolun Wu, Dun Yuan, Li Jiang, Di Wu, Xue Liu, Charlie Zhang, Xianbin Wang, and Jiangchuan Liu. Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities, 2024.