Towards End-to-End Network Intent Management with Large Language Models

Lam Dinh, Sihem Cherrared, Xiaofeng Huang, and Fabrice Guillemin
Orange Innovation, France

***Abstract*—Large Language Models (LLMs) are likely to play a key role in Intent-Based Networking (IBN) as they show remarkable performance in interpreting human language as well as code generation, enabling the translation of high-level intents expressed by humans into low-level network configurations. In this paper, we leverage closed-source language models (i.e., Google Gemini 1.5 pro, ChatGPT-4) and open-source models (i.e., LLama, Mistral) to investigate their capacity to generate E2E network configurations for radio access networks (RANs) and core networks in 5G/6G mobile networks. We introduce a novel performance metrics, known as FEACI, to quantitatively assess the format (F), explainability (E), accuracy (A), cost (C), and inference time (I) of the generated answer; existing general metrics are unable to capture these features. The results of our study demonstrate that open-source models can achieve comparable or even superior translation performance compared with the closed-source models requiring costly hardware setup and not accessible to all users.**

***Index Terms*—Intent-Based Networking (IBN), End-to-End (E2E) network, Large Language Models (LLMs), LLM metrics.**

I. Introduction

The complexity of network services and applications has continually grown with the advent of novel technologies, especially in the recent few years with the emergence of virtualization technologies applied to 5G networks (e.g., containerized network functions, specialized or shared network slices, etc.). While integrating new network services is necessary to enhance network performance and facilitate a greater variety of applications, it inevitably creates difficulties for network management because network configurations, which contain a large number of technical parameters in different network domains, are no longer sustainable for human operation [1]. This requires new network management strategies, notably Zero-Touch Service Management (ZSM), in which the network operations are fully automated through *self-healing*, *self-configuration*, and *self-optimization* [2].

Policy-driven and intent-based mechanisms are among the key concepts that enable autonomous networks, as highlighted in [3]. Specifically, ZSM utilizes the notions of Intent-Based Networking (IBN) [4] for enabling end-users to specify their high-level network operational goals by defining intents through various network interfaces. This method fills the technical gap between novice users and network management, as IBN allows users with minimal network expertise to focus on the desired network operational goals, rather than on technical specifications.

This work is supported by French government funding within the France 2030 framework through the INFLUENCE project.
ISBN 978-3-903176-72-0 ©2025 IFIP

Once the user has well defined intents, IBN plays an important role in translating them into technical policy configurations and actions, that can be consumed by the available set of controllers and orchestrators for further deployment. For instance, any non-expert user can interact with network to establish both 5G eMBB slice and URLLC slice for a particular region, using natural language descriptions as follows: *"I need to setup eMBB and URLLC slices for the Paris region. For the eMBB slice, our service covers up to 10,000 users at the same time, with a minimum throughput of 100 Mbps. For the URLLC slice, we require to meet a maximum end-to-end latency of 10ms in at least 99,999 % of time for a minimum of 1000 users."*. To translate this description into a deployable network configuration, IBN first tries to extract network intents (i.e., eMBB, 10000 users, 100 Mbps; URLLC 1000 users, 10ms of delay, etc.) before creating detailed technical configurations of network functions (i.e., RAN and Core Network functions) for the building and running phases. The entire process is carried-out via intent life-cycle management procedures, which involves five stages: *profiling*, *translation*, *resolution*, *activation*, and *assurance* [5].

However, there are multiple challenges associated with the use of IBN for End-to-End (E2E) network management: *(i)* High-level network intents must be understandable to both the user and the network. For this reason, they are typically documented in human-readable languages (i.e., JSON, YAML, etc.), and are formularized by certain standards (i.e., 3GPP, TMForum, etc.) with a large number of technical requirements, subject to errors. Therefore, users must master such description methods to harness the full potential of the intent concept. Sometimes, it is challenging for non-expert users to manually describe all the network objectives in simple natural language, which implies additional complexities for network management. In addition, when the technical intents are fully extracted, it is difficult to map intents onto specific network functions during the network activation phase.

All those difficulties related to intent life-cycle management can potentially be mitigated with Large Language Models (LLMs) thanks to their capabilities of understanding, generating, and interpreting human language and facilitating network management for users without extensive technical expertise [6], [7]. However, practical applications of LLM for end-to-end network management is still limited, because it lacks to some extent performance indicators that measure how good/bad the obtained answers can be used for service deployments. Furthermore, most of the studies focused on closed-source GPT models (e.g., GPT-4), for which model specifications (e.g., number of parameters, activation functions, etc.) are not disclosed. This limitation not only hampers our ability to customize the models via fine-tuning and instruction prompts, but also inhibits our understanding of what occurs within these hidden models.

In the objective of developing an LLM based IBN framework with verifiable responses, our key contributions are as follows:

- First, we define an IBN framework, in which business intent resolver and service intent resolver are introduced in the intent layer for translating high level user expressions into network resource configurations. This architecture is compatible with the Open Digital Architecture (ODA) specified by the TM Forum.
- Second, we improve the responses of LLM with regard to intent translation. In this study, we apply prompting on both closed-source and open-source LLM models to study the generated network configurations.
- Finally, we propose a new evaluation benchmark, referred to as FEACI, to objectively check the generated responses from LLM models against the expectation of the network providers in 5 factors: Format (F), Explainability (E), Accuracy (A), Cost (C), and Inference time (I).

The paper is organized as follows: Related work is discussed in Section II. Section III presents the general system model and our contribution, whereas Section IV provides numerical results. Finally, Section V presents concluding remarks.

II. Related works

In the literature, Natural Language Processing (NLP) interfaces between users and the network to facilitate network management through the use of natural language has been considered in various studies, see for instance [8], [9]. This approach has however several shortcomings when accomplishing complex tasks that require higher language understanding and generation capabilities. Alternatively, Large Language Models (LLMs) have been proposed to interpret and transform user intents into network policies and/or generate network configurations to match user descriptions [10]–[12]. The use of NLP/LLMs in intent-based networking can be categorized into several topics as detailed in the following sections.

A. Intent extraction

Empowered by NLP, Mcnamara et al. [8] and Orlandi et al. [11] embed NLP interfaces into the intent engine to facilitate interaction between humans and network management systems for slice provisioning and network service deployment. Although NLP has been demonstrated to perform several rudimentary tasks, such as detecting low-level context absent from user intents [8], or proposing relevant network services from natural language descriptions [13], its capabilities remain insufficient to translate user intents into low-level network configurations. This limitation can be attributed to their inefficiency and lack of customizability in translation tasks, which make them inadequate for the E2E network intent management. Taking this into account, Cesila et al. [14] introduce a customizable, open-source RASA Chatbot into the Intent (Translator) Engine, helping users configure the optical network through high-level language communication. Despite the high level of customization, human-in-the-loop is imperative to validate the chatbot's response, thereby constraining the scalability of the solution. These papers [12], [15], [16] demonstrate the potential use of LLMs for extracting network intents from user descriptions or from network log data. Based on prompts that include task description and background contexts, closed-source LLMs (e.g., GPT-4) are used in these works to provide the extraction of network intents in the response together with explainability. However, this study does not address the question of whether LLM responses can be used directly in intent management when evaluation is not taken into account.

B. Intent translation/assurance

The studies [17], [18] propose intent-based infrastructure management systems that leverage few-shot prompting in closed-source GPT-based models to translate natural language intents into computer programs for multi-domain infrastructure. With respect to the use of open-source LLMs for intent management, Mekrache et al. [19] deploy Mistral, LLama models to configure and manage E2E network services. Particularly, they are capable of collecting Cloud/Edge intents and RAN intents (e.g., slicing, latency, throughput, etc.) from user description to create network configurations in JSON format, and those configurations can be passed to the Network Function Virtualization Orchestrator (NFVO) for network service deployment. However, there are several drawbacks encountered in these studies: *(i)* the efficiency of intent translation are not fully examined, when it is not clear if the LLMs are able to produce efficient translations and it may lead to the conflicts in the intent activation, *(ii)* Human Feedback is involved in the process, and it can be time-consuming. Once the intents are fully acquired from user descriptions , it is also crucial for the network orchestrator to deploy adequate policies to meet the user's requirement. In this context, Collet et al. [20] propose an automated network management framework model, so called *LossLeap*, designed to proactively manage network configurations in order to guarantee the fulfillment of complex network objectives represented by user intents. This study deals with intent assurance and will not be considered in our work.

C. Our approach

In this work, we apply both closed-source and open-source LLMs to deal with (1) the translation of high-level intents from users into low-level network configurations, and (2) the mapping between technical intents and network configurations, via prompting [21]. Furthermore, we introduce a novel performance metrics, namely FEACI (see the Introduction).

III. End-to-End Network Intent Management

A. Network Intent Architecture

IBN introduces an additional layer in addition to the business and the network layers to manage the life-cycle of high level intents from their initiation at the business layer to practical realization at the network layer. The intent layer generally achieves five main functional blocks: [5]: *(i)* **Intent Profiling** is the communication gateway between IBN system and users. It is designed so that users can express their meaningful intents in a human-friendly way (i.e., through natural language expression, drop down menu, etc.). *(ii)* **Intent Translation** is responsible for translating those abstracted intents into network policy that will be seen as low-level network configuration for network infrastructure. *(iii)* **Intent Resolution** is used to resolve the conflicting intents that are infeasible in current network states. *(iv)* **Intent Activation** deploys feasible intents so that the networks are configured as intended by the user, and *(iv)* **Intent Assurance** ensures that the network complies with the user intents throughout its lifetime.

In this paper, we primarily focus on the the life-cycle management of an intent from the moment it is initiated till the low-level configuration of network function (i.e., Intent Profiling, Translation and Resolution.). The IBN architecture, which highlights our contributions by ignoring the not relevant functional blocks, is shown in Figure 1.

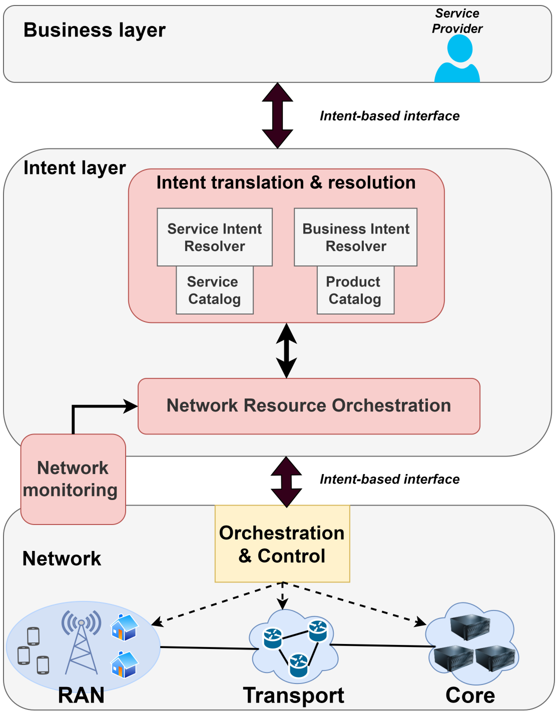

Fig. 1. Network architecture with Intent layer

In the intent layer, the block responsible for intent translation is near the user interface to facilitate interactions with users and to gather user intents through natural language communications. There are two primary ways for users to convey their high-level objectives for network management:

1) Users **explicitly** describe network goals via natural language, for instance *"I need to set up URLLC slices for the Paris region. We aim to meet a maximum end-to-end latency of 10ms in at least 99,999 % of time for a minimum of 1000 users."*. The intent translation and resolution block then extracts technical intents from user inputs (e.g., end-to-end latency of 10 ms, reliability: 99,999 %, supported users: 1000 etc.), prior to parameterizing network functions to meet these goals.
2) Users specify **which/where/when** service they want to set up. This demand will be mapped into a product catalog that contains numerous services, each of which has been configured with technical parameters. For example, user demands following this method can be *"I need to establish URLLC slices for the Paris region"*. Afterwards, both "URLLC service" (**which**) and "Paris region" (**where**) from user demand are used to match the URLLC product, which is one of the products in the catalog, and contains network parameters such as end-to-end latency of 10 ms, reliability: 99,999 %, supported users (1000), and so on.

In this paper, we target the second method within the intent translation and resolution block, as it enables non-expert users to express their needs without being network experts. **Business Intent Resolver** and **Service Intent Resolver** are then used [11] to handle the service orders. As illustrated in Figure 2, the business resolver translates the requirements from business users (i.e., product intent) into *service order* specifying which specific SLAs are used to describe the user demand. Subsequently, service resolver transforms the service order into *resource order* that can be orchestrated by the network resource orchestrator.

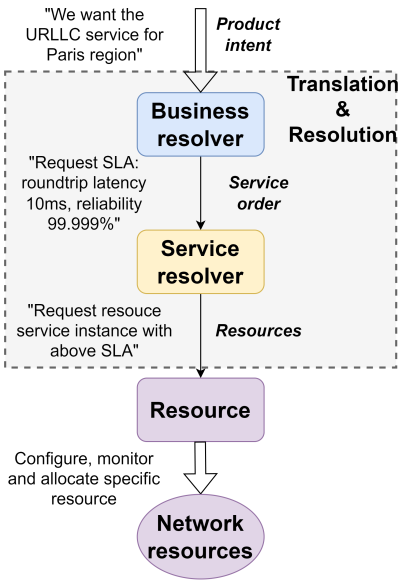

Fig. 2. Translation and resolution block with business and service resolver

The order from business users is mapped onto a single intent-driven product, which contains a subset of (networking) parameters, such as expected E2E latency, supported users, deployment region, etc. For instance, any mission critical product can be characterized with low latency properties (i.e., in the range [1ms;10ms] for industrial use cases), or a high broadband product focus on higher network throughput services (i.e., [100Mpbs:1000Mbps] for consumer use cases). Each product may also be enriched with contextual and meta data information (e.g., ranking of user expertise for data exposure during service implementation) to better adapt to the expert or non-expert user profile.

Once the product order is validated at the business resolver level, the **Service Intent Resolver** begins to verify if the technical intents described in the ordered product can be deployed. Subsequently, those technical intents are converted into technical solutions in vertical network domain (RAN, Transport, Edge/cloud, etc.) to achieve operational goals described in the product specifications. For example, those technical solutions are related to the required cloud resources (e.g., CPU speed, RAM volume, storage, etc) for the core network, topology settings for transport network or radio-related configurations for RAN network. Note that those technical details are not exposed to the users, and are specified in the service catalog.

Finally, the technical solutions, which are produced by the Service Intent Resolver, are handled by the Network Resource Orchestrator (NRO) for their implementation. To ensure user satisfaction over the life-cycle of the intents, the NRO should implement proactive resource provisioning to accommodate the dynamic changes of the underlying network. For, it should continually measures network status and dynamically fine-tune the network configurations to meet user expectation. This point is not addressed in this paper.

In the following, we propose an approach that is based on LLMs to manage business and service resolver for interpreting user intents into RAN and Core configurations.

B. Large Language Models

LLMs such as GPT4 [22], Gemini [23], etc. are capable of understanding contextual data (i.e., texts, images, etc.) across a wide range of technical domain (i.e., telecom-specific, code generation, etc.). The primary factor behind the success of LLMs is largely due to the transformer architecture [24] that enables them to comprehend information context more effectively by selectively paying attention to various parts of the input, and their ability to access a vast corpus of data (e.g., books, internet sources, etc.) during training. Specifically, based on self-attention mechanism, which stands as a core component of transformers, the long-range dependencies and contextual relationships can be inferred from the input data. As a result, the generated output will be generated with high level of coherence and contextual relevance. This approach enhances the results compared to the use of RNN as all relevant information is ephemeral and only represented by the current hidden state.

To demonstrate the relevance of an LLM in the IBN context, Figure 3 presents an LLM architecture, which is composed of several processing blocks: *(i) Pre-processing*, *(ii)* Transformer and *(iii) Post-processing* . At the *pre-processing* stage, human-understandable data is de-serialized into machine-understandable data using *tokenization* and *embedding*. They are responsible for converting human input sequence into corresponding trainable embedding vectors.

Inside the *transformer* block, multi-head attention (i.e., LM head) with trainable weights is introduced to tackle the input sequences that have complex pattern recognition. In this case, each LM head is dedicated to learn different kinds of semantic information of input sequence. Furthermore, it is common in the LLM models that a transformer is placed on top of another to iteratively apply the multi-head self-attention mechanism N times in order to further enhance the efficiency of the input pattern recognition. Finally, the output of each LM head in the final layer of the transformer block is normalized before passing through a Feed-forward Neural Network (FNN) at the *post-processing* step. The goal of this step is to map the LLM output tensors, which are processed within the transformer, with the corresponding tokens to generate the response.

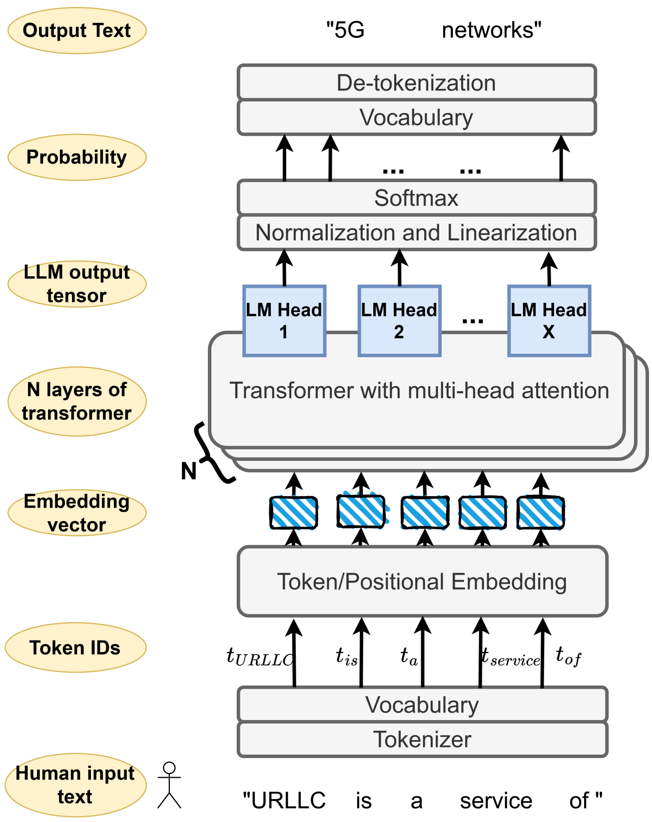

Fig. 3. Illustration of an LLM workflow

C. From natural language to network deployment with LLMs

Figure 4 shows our proposal with the use of LLMs to translate user requests into operational intents, and to activate the corresponding network resources to meet user intents. In this architecture, users first interact with a chatbot based on LLM using natural language to express their needs, for instance *"I would like to create an E2E URLLC slicing for Parisian region."*. Then, given the capability of LLMs to understand human languages and semantic information, they manage to match user demands with *at least* one of the network products that are available.

It should be highlighted that each product contains multiple parameters of the desired services. In the business resolver, the product descriptions, which are represented as the operational intents, are usually formatted according to the TM Forum ODA definitions. It takes into account the general parameters of the creation order (e.g., required latency value, etc.) as well as their metadata (e.g., *id* in the product catalog, date of creation, category, etc.). This format is also known as Custom-Facing Service (CFS) specification.

The second translation is also carried out by the LLMs at the level of service resolver. Given the constraints of the service prior to the deployment, the available services in the catalog, and the TM Forum ODA format, a set of technical solutions, also known as Resource-Facing Service (RFS), is produced from the CFS description that is available in the first step. Those RFS are formatted and ready to be fed onto the Network Resource Orchestrator (NRO) for the imminent deployment.

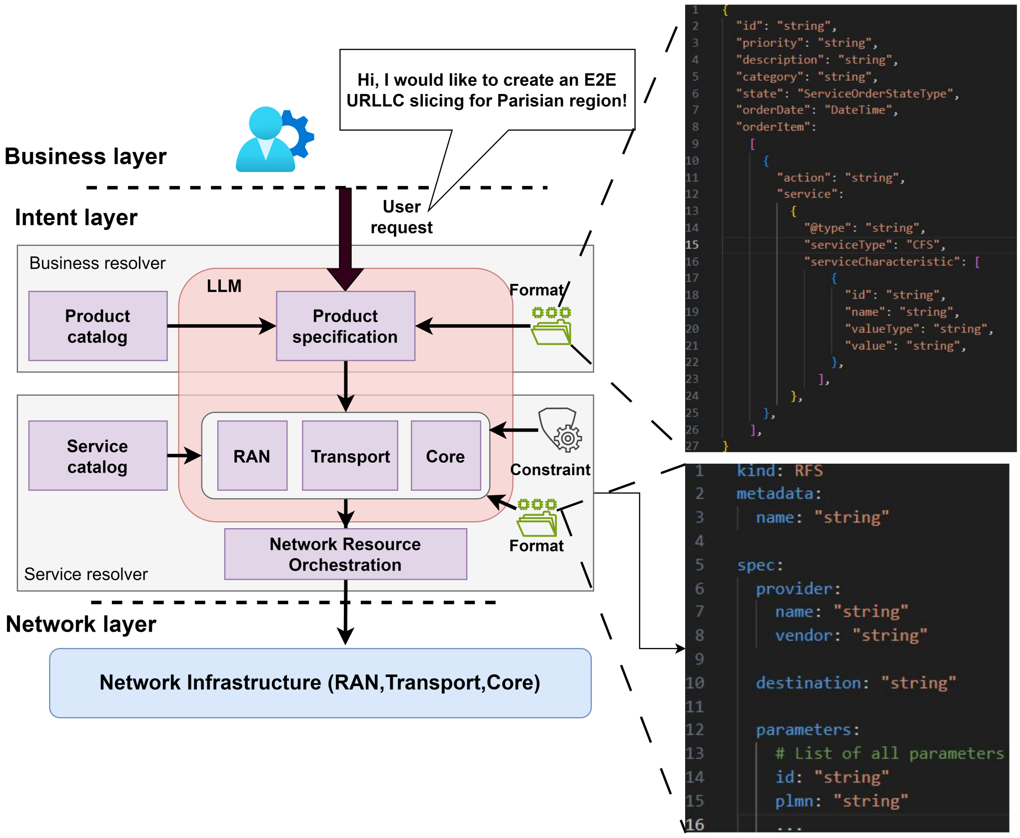

Fig. 4. Intent translation and resolution with Large Language Models

D. Prompting

To better customize the output of the LLM models to match the user expectation, Chain-of-Thought (CoT) prompting is considered as an efficient approach [21]. By providing the models with additional information via prompting, it improves the ability of the language models to understand similar tasks and efficiently respond to them. Practically, a CoT prompt includes a tuple $< Q_i, CoT_i, A_i >$, where Q_i is a description of a task, CoT_i is a series of reasoning steps that lead to the result A_i. In this study, we consider zero-shot prompting (ZERO), one-shot prompting (ONE), and few-shot CoT prompting (FEW), which respectively provide zero example, one example of expected format output, and few examples of how to generate final output.

Without providing any additional context to the LLM models, we examine in ZERO how they can generate technical intents in the "standard" format (i.e., 3GPP and TMF), of which they might not know during training. The generated answer from LLM models will be investigated in: (1) how good the format is, (2) how relevant of each technical intent is and (3) how feasible the models can do reasoning. In this regard, we create a target configuration, which serves as a reference for the performance comparison.

In one-shot and few-shot prompting, we supply additional examples to the LLM models, specifying the service order given in JSON file and the expected results in YAML format. In particular, the one-shot prompt only contains the service order and the expected response, while few-shot CoT prompting additionally shows how to calculate certain fields in the expected response, given the technical intents in the service order as shown in Figure 6 and Figure 7, respectively. Then, we pass the target service order to the LLM models and receive the generated results. Those obtained results are compared with a reference result, which is pre-designed by telecom expert, to evaluate how good/bad the translations are.

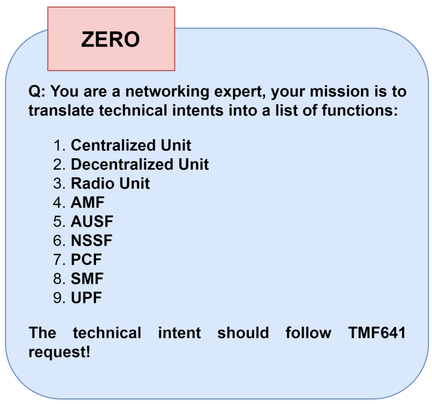

Fig. 5. Zero-shot prompting (ZERO)

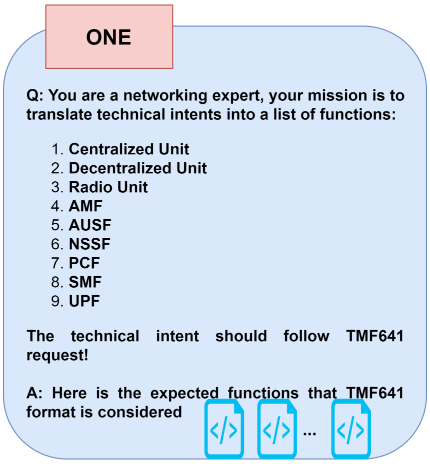

Fig. 6. One-shot prompting (ONE)

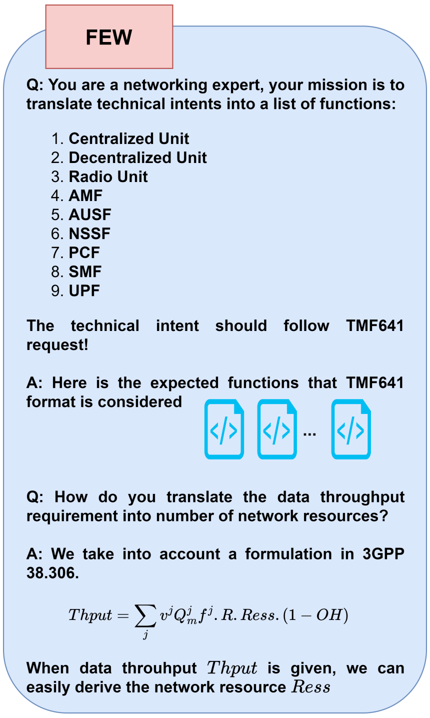

Fig. 7. Few-shot prompting (FEW)

E. LLM evaluation for intent translation/resolution - FEACI

In order to evaluate how the answers generated from the LLM model, general metrics such as BLEU [25] and ROUGE [26] are commonly used in the literature. However, those metrics are mainly used to numerically examine the similarities between machine-translated text and references provided by telco experts. They have been shown to be inefficient to capture the semantic and context understanding of telecom-related responses where different words with analogous meaning may result in overall low scores. As a consequence, it is not appropriate to use them to evaluate the responses generated from LLM models in this work. To deal with this issue, we propose a novel evaluation metric, referred to as FEACI, to respectively evaluate the Format, Explanation, Accuracy, Cost, and Inference time of the generated responses as follows:

- Format: *F-score* measures the correctness of the structure of the output proposed by the LLM compared to the reference structure of the RFS description (e.g., in JSON/YML, etc.). For each translation of the CFS-RFS pair, we consider a Boolean value: $F = 1$ indicates that the generated network configuration is extractable by the resource orchestrator, otherwise $F = 0$. Given a CFS catalog $\mathcal{D}$, the generated configuration of LLM i is recorded in the set $\mathcal{A}$. The format score of LLM i (i.e., $0 \leq F_{\mathcal{A}}^{(i)} \leq 1$) is evaluated by the probability $\mathbb{P}$ as follows:
$$F_{\mathcal{D}}^{(i)} = \mathbb{P}\{F(o_a^i) = 1 | o_a \in \mathcal{A}\} \tag{1}$$
- Explanation: *E-score* indicates how good is an explanation that the LLM proposes for its output RFS. An explanation is not mandatory in our use case, and sometimes an explanation increases the number of tokens. However, for evaluation purposes, we consider the quality of the explanation to understand the result and improve the prompt, especially in cases where the LLM performs multiple calculations before providing the output (as described in the prompt of Figure 7). $E = 1$ means LLM models can produce a satisfactory explanation for those responses generated, otherwise $E = 0$. In this work, we applied human evaluation to assess the quality of the explanation. The explanation score of the generated set $\mathcal{A}$ of the LLM i (i.e., $0 \leq E_{\mathcal{A}}^{(i)} \leq 1$), given CFS catalog $\mathcal{D}$ is as follows:
$$E_{\mathcal{D}}^{(i)} = \mathbb{P}\{E(o_a^i) = 1 | o_a \in \mathcal{A}\} \tag{2}$$
- Accuracy: *A-score* assesses the configuration values of LLM responses by comparing them to the respective reference values. Together with confirming the format of LLM responses, this metric aims to quantify the quality of LLM generation by statistically measuring the percentage of matching values between the response and the target reference. In this sense, each output o_a^i of LLM i receives a score $A^{(i)} \in [0, 1]$. The accuracy score of the generated set $\mathcal{A}$ of the LLM i (i.e., $0 \leq A_{\mathcal{A}}^{(i)} \leq 1$), given CFS catalog $\mathcal{D}$ is measured by the following expectation $\mathbb{E}$:
$$A_{\mathcal{D}}^{(i)} = \mathbb{E}\{A^{(i)}(o_a^i) | o_a \in \mathcal{A}\} \tag{3}$$
- Cost: *C-score* measures how much we should pay (in terms of *USD*) for the prompt tokens used. For the close-source LLMs (i.e., Gemini 1.5 pro or GPT-4), the amount of input tokens (i.e., N_{in}) and output tokens (i.e., N_{out}) are used to determined the cost (i.e., c_i and c_o, respectively) as
$$C = c_i(N_{in}) + c_o(N_{out}) \tag{4}$$
With open-source LLMs (i.e., LLama, Mistral, etc), there is no additional cost associated with the number of tokens generated (i.e., $C = 0$). The normalized cost ($0 \leq C^{(n)} \leq 1$) is computed based on a threshold C_0 as follows:
$$C^{(n)} = \begin{cases} \frac{C}{C_0} & \text{if } C \leq 10C_0 \\ 1 & \text{if } C > 10C_0 \end{cases} \tag{5}$$

In this work, we consider the reference $C_o = 0.1$ [USD]. Given CFS catalog $\mathcal{D}$, C-score is computed as follows:

$$C_{\mathcal{D}}^{(i)} = \mathbb{E}\{C^{(n)}(o_a^i)|o_a \in \mathcal{A}\} \tag{6}$$

- Inference time: *I-score* measures the delay of generating the answer from the moment users query the LLM models until the responses are shown. Generally, this metric is heavily dependent on the processing time (i.e., t_{proc}) of input tokens (i.e.,$X_{inToken}$) in the transformer layer and time (i.e., t_{gen}) required for generating output tokens (i.e.,$Y_{outToken}$).

$$I = t_{proc}(X_{inToken}) + t_{gen}(Y_{outToken}) \tag{7}$$

The normalized version of this metric ($I^{(n)}$) is computed by defining a threshold I_0 ($I_0 > 0$) as follows:

$$I^{(n)} = \begin{cases} \frac{I}{I_0} & \text{if } I \leq I_0 \\ 1 & \text{if } I > I_0 \end{cases} \tag{8}$$

In this work, we consider the threshold $I_o = 60$ [seconds]. Given CFS catalog $\mathcal{D}$, I-score is computed as follows:

$$I_{\mathcal{D}}^{(i)} = \mathbb{E}\{I^{(n)}(o_a^i)|o_a \in \mathcal{A}\} \tag{9}$$

The following evaluation score is proposed to examine the LLM models i for the translation of service resolver from the data set $\mathcal{D}$:

$$\mathcal{E}_{serv}^{(i)} = \omega_1 \times F_{\mathcal{D}}^{(i)} + \omega_2 \times E_{\mathcal{D}}^{(i)} + \omega_3 \times A_{\mathcal{D}}^{(i)} - \omega_4 \times C_{\mathcal{D}}^{(i)} - \omega_5 \times I_{\mathcal{D}}^{(i)} \tag{10}$$

where $0 \leq \omega_i \leq 1$, $i = 1, \ldots, 5$, is the corresponding weight of each elementary score, representing the weight of each individual contribution to the total score.

IV. Results and Discussion

A. Scenario description

For the sake of simplicity, our E2E network omits the transport network and comprises only two entities: *(i)* a RAN and *(ii)* a Core network. To establish an E2E network service from user intents, the corresponding RAN and Core network functions must be appropriately configured, prior to the deployment. Figure 8 illustrates our target scenario in which a Radio Unit (RU), a Distributed Unit (DU) and a Control Unit (CU) are involved as the RAN functions, while User Plane Function (UPF), Access and Mobility Management Function (AMF), Policy Control Function (PCF), Session Manager Function (SMF), Authentication Server Function (AUSF) and Network Slice Selection Function (NSSF) are considered in the Core network. Customers express their demands through a User-friendly chatbot containing technical parameters (i.e., CFS) such as service slicing types, throughput requirements, maximum latency, numbers of supported users, etc. and wait for the deployment of their demands that are translated into a list of configurations for each network function (i.e., RFS).

B. Simulation setup

Our experiment setup for testing open-source LLM models consists of a computing server, which is built based on 12xIntel(R) Xeon(R) E-2236 CPU of speed 3.4GHz and 2xNvidia H100 GPUs with 80GB of vRAM. We regard Gemini version

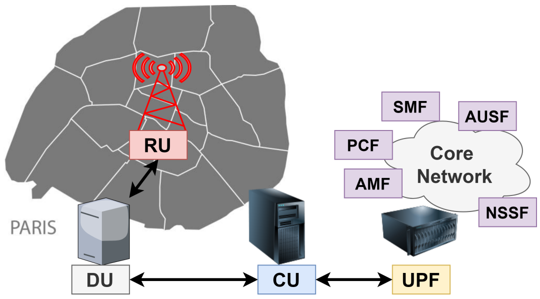

Fig. 8. Network deployment for Parisien region

1.5 Pro (GEM 1.5) and ChatGPT version 4 (GPT-4) as closed-source models, since their underlying model specifications are inaccessible to the public. In this case, we use API to send the request to the closed-source model. On the other hand, the open models such as Llama-3 with 8B parameters (LLama I), 70B parameters (II), and Mistral models with 7B parameters (I), Mixtral 8x7B parameters (II) are also considered in our study.

In general, closed-source models, which are only accessible via APIs, have more trainable parameters than open-source counterparts, which are locally stored in our computing server. We also demonstrate the inference cost of the LLM models by comparing the total cost associated with the number of input/output tokens required to tokenize the prompt and generate answers (i.e., in USD) per 1 million tokens used. In this regard, the costs associated with the open-source models are 0, in contrast to those found in the closed-source models.

Concerning the decoding strategy for the language model to generate the network configuration, temperature sampling and nucleus sampling are used where temperature (i.e., T parameter) and $top - p$ are set at 0.2 and 0.9, respectively. Table I presents the parameters of open-source and closed-source LLM models employed in this work.

TABLE I
(Hyper)-Parameters of LLM models

Parameter(s)	**Model(s)**					
	GEM 1.5	GPT 4	Llama I	LLama II	Mistral I	Mixtral II
N-params	1500B	1760B	8B	70B	7B	45B
Open	N	N	Y	Y	Y	Y
Attention heads	N/A	N/A	32	32	32	32
Token Size	N/A	N/A	128K	128K	32K	32K
Attention layer(s)	N/A	N/A	32	80	32	32
Activation function	N/A	N/A	SILU	SILU	SILU	SILU
Cost In/Out (per 1M TK)	1.25/5	10/30	0/0	0/0	0/0	0/0

C. Numerical results

To obtain the statistical results of intent translation, we consider a catalog containing 10 network service orders that correspond to 10 CFSs. Then, each translation from an CFS into an RFS is iterated 10 times for the average results.

Table II gives the BLEU/ROUGE performance of intent translation when zero-shot (ZERO), one-shot (ONE) and few-shot (FEW) prompting are applied on the closed-source/open-source LLM models. In particular, the generation of network configurations for each model and each prompting approach is compared to a reference configuration, which is pre-designed by our telco experts. As shown in this table, without prompts (i.e., ZERO), the scores of BLEU/ROUGE are 0 which signifies that the LLM models are not able to properly export the network configurations corresponding to the expert's expectation. When one-shot/few-shot prompts are applied to the models, they are capable of producing the correct format of network configuration. This explains why the scores improve when compared to those of zero-shot prompting. However, those numbers do not have any significative meaning as we do not know how effective the translations are. As a consequence, we cannot use this general metric as a benchmark to compare the translation performance of the LLM models.

TABLE II
PERFORMANCE OF INTENT TRANSLATION WITH GENERAL METRICS.

Prompt	Metrics	GEM	GPT	LLAMA-I	LLAMA-II	Mistral-I	Mistral-II
ZERO	BLEU/ROUGE	0.0	0.0	0.0	0.0	0.0	0.0
ONE	BLEU	0.415	0.382	0.236	0.331	0.332	0.361
	ROUGE-1	0.164	0.177	0.124	0.173	0.145	0.147
	ROUGE-2	0.130	0.145	0.097	0.115	0.111	0.111
	ROUGE-L	0.164	0.177	0.124	0.173	0.145	0.147
FEW	BLEU	0.455	0.291	0.376	0.429	0.383	0.394
	ROUGE-1	0.243	0.117	0.599	0.158	0.145	0.141
	ROUGE-2	0.217	0.085	0.0	0.119	0.111	0.106
	ROUGE-L	0.243	0.117	0.599	0.158	0.145	0.141

Table III compares translation performance when our metrics (i.e., FEACI) are used. They examine the Format (F), Explanation (E), Accuracy (A), normalized Cost (C), and normalized Inference time (I) of each language model to produce technical intents for RAN and Core networks. It shows that one-shot prompts is essential for the models to understand the output format of the network specifications, while few-shot prompting plays a significant role in reasoning why the certain results are generated (i.e., better scores obtained for the explanation (E) and Accuracy (A) metrics).

In terms of accuracy, zero-shot prompting (ZERO) is insufficient for the models to produce the expected results when there is no matching between the configuration file generated from the models and the references. One-shot prompting (ONE) considerably improves the accuracy to 54%-84%, when an example of format and expected results is given to the LLM models. Few-shot prompting (FEW) further enhances accuracy of the intent translation compared to one-shot prompting (ONE). Besides, the numerical values show that open-source LLM models allow us to achieve accuracy performance similar to the one of closed-source models, which are much larger and computationally expensive.

In terms of computational cost, only closed-source models are taken into account as there is a price associated with the number of input/output tokens used to tokenize the user demands and generate the responses. Among them, the most expensive model is GPT-4. Furthermore, more examples we provide to the models, higher number of input tokens are used and turns out in higher cost. It explains why the cost related to the few-shot example is higher than the other prompting schemes.

TABLE III
PERFORMANCE OF INTENT TRANSLATION WITH OUR METRICS

Metrics	Prompt	GEM	GPT	LLAMA -I	LLAMA -II	Mistral -I	Mistral -II
Format	ZERO	0.2	0.1	0.0	0.0	0.0	0.0
	ONE	0.9	0.86	0.65	0.78	0.85	0.87
	FEW	0.94	0.89	0.75	0.82	0.84	0.88
Explain	ZERO	0.0	0.0	0.0	0.0	0.0	0.0
	ONE	0.83	0.64	0.72	0.75	0.71	0.86
	FEW	0.97	0.87	0.82	0.86	0.83	0.89
Accuracy	ZERO	0.0	0.0	0.0	0.0	0.0	0.0
	ONE	0.84	0.54	0.72	0.71	0.78	0.82
	FEW	0.93	0.62	0.82	0.75	0.84	0.86
Normalized Cost	ZERO	0.03	0.87	0.0	0.0	0.0	0.0
	ONE	0.08	1.0	0.0	0.0	0.0	0.0
	FEW	0.1	1.0	0.0	0.0	0.0	0.0
Normalized Inference	ZERO	0.24	0.36	0.62	1.0	0.65	0.84
	ONE	0.27	0.41	0.69	1.0	0.67	1.0
	FEW	0.31	0.45	0.71	1.0	0.73	1.0

Finally, we compare how much (inference) time is needed for the model to generate the responses. Overall, this value depends on how many tokens that the models use to generate the results and is normalized by $I_o = 60$ [seconds]. As we can see from the table, few-shot prompting requires longer inference time to generate the answers because it takes more time to process the additional tokens. In general, inference time of the Google Gemini 1.5 pro model is shortest when compared with the GPT v4 model and open-source model. Besides, inference time of the open-source models with higher number of parameters (i.e., Mistral-II and LLama-II) is longer than with the lighter ones (i.e., Mistral-I and LLama-I).

Table IV details the count of tokens that are consumed by each language model to process the input prompt (i.e., zero-shot, one-shot, and few-shot) and to generate the response (output). In general, providing more examples through few-shot prompting enables the model to gain a deeper understanding of the context and produce more effective responses. As a result, a greater number of tokens are utilized during the inference process, thereby increasing the inference time overall. In contrast, zero-shot promptings do not require relevant examples to process user demands and results in lower number of tokens being used. In this scenario, while the inference time is significantly reduced when compared to few-shot prompting, the generated results are susceptible to *hallucination* when the accuracy, the format, and explainability of the results do not meet user expectation.

Figure 9 illustrates the total evaluation score, which is described in Equation 10, when the weights are evenly distributed (i.e., $\omega_i = 0.2$ for $i = 1, ..., 5$). It should be highlighted that the cost (C) and inference time (I) are normalized in this case. As shown in this figure, the evaluation score of the LLM translation when few-shot prompts are applied is higher than

TABLE IV
NUMBER OF AVERAGE INPUT/OUTPUT TOKENS USED

	GEM	GPT	LLAMA-I	LLAMA-II	Mistral-I	Mistral-II
ZERO	2817/2614	2588/2036	2443/3401	2680/3164	3094/2646	3094/2234
ONE	6511/3299	6858/2659	5697/4396	6723/3523	7328/1813	7328/2413
FEW	8729/2683	8090/1865	7707/3386	8056/2520	9769/2486	9769/1944

that of the one-shot and zero-shot prompts, when both accuracy and format scores are higher. GPT-4 model tends to perform the worst because the cost-related to the token consummation is the highest, whilst the accuracy, format, explanation scores are not better than the other models.

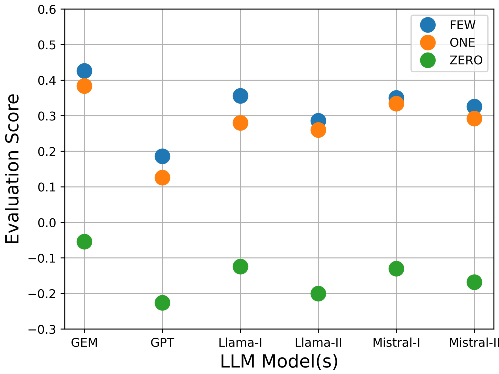

Fig. 9. Evaluation scores of LLM models including the FEACI metrics

V. CONCLUSION

In this study, we have presented an end-to-end network intent management framework that relies on LLMs. Leveraging the capacity of LLMs to comprehend complex tasks via prompt-based instruction, we have addressed the obstacles encountered in the translation and resolution phases of IBN. Benchmarking the performance of both closed-source models (i.e., Google Gemini 1.5 pro, ChatGPT-4) and open-source models (i.e., LLama-8B(I), LLama-70B(II), Mistral-7B(I) and Mixtral-8x7B(II)) with few-shot (FEW), one-shot (ONE) and zero-shot (ZERO) prompts, we have demonstrated that prompting is an efficient method to substantially enhance the performance. Furthermore, we have proposed FEACI, a novel evaluation metric that assesses the performance of LLMs in generating network configurations from user intents. The results reveal that open-source models can achieve similar or even better performance than closed-source models. Future studies should examine the techniques to further improve the reasoning capabilities and evaluation scores of the models.

REFERENCES

[1] K. Abbas, T. A. Khan, M. Afaq, and W.-C. Song, "Network Slice Lifecycle Management for 5G Mobile Networks: An Intent-Based Networking Approach," *IEEE Access*, vol. 9, pp. 80 128–80 146, 2021.
[2] ETSI, "Zero-touch network and Service Management (ZSM); Intent-driven Closed Loops," 2024.
[3] M. Liyanage, Q.-V. Pham, K. Dev, S. Bhattacharya, P. K. R. Maddikunta, T. R. Gadekallu, and G. Yenduri, "A survey on Zero touch network and Service Management (ZSM) for 5G and beyond networks," *Journal of Network and Computer Applications*, vol. 203, p. 103362, Jul. 2022.
[4] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Definitions," Oct. 2022.
[5] A. Leivadeas and M. Falkner, "A Survey on Intent-Based Networking," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, 2023.
[6] C. Liu, X. Xie, X. Zhang, and Y. Cui, "Large Language Models for Networking: Workflow, Advances and Challenges," *Computing Research Repository (CoRR)*, no. arXiv:2404.12901, Apr. 2024.
[7] A. Maatouk, N. Piovesan, F. Ayed, A. D. Domenico, and M. Debbah, "Large Language Models for Telecom: Forthcoming Impact on the Industry," *IEEE Communications Magazine*, pp. 1–7, 2024.
[8] J. Mcnamara, D. Camps-Mur, M. Goodarzi, H. Frank, L. Chinchilla-Romero, F. Cañellas, A. Fernández-Fernández, and S. Yan, "NLP Powered Intent Based Network Management for Private 5G Networks," *IEEE Access*, vol. 11, pp. 36 642–36 657, 2023.
[9] 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.
[10] D. Brodimas, K. Trantzas, B. Agko, G. C. Tziavas, C. Tranoris, S. Denazis, and A. Birbas, "Towards Intent-based Network Management for the 6G System adopting Multimodal Generative AI," in *2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, Jun. 2024, pp. 848–853.
[11] B. Orlandi, S. Lataste, S. Kerboeuf, M. Bouillon, X. Huang, F. Faucheux, A. Shahbazi, and P. Delvallet, "Intent-Based Network Management with User-Friendly Interfaces and Natural Language Processing," in *Clouds, Internet and Networks (ICIN)*, Mar. 2024, pp. 163–170.
[12] D. M. Manias, A. Chouman, and A. Shami, "Towards Intent-Based Network Management: Large Language Models for Intent Extraction in 5G Core Networks," in *2024 20th International Conference on the Design of Reliable Communication Networks (DRCN)*, May 2024.
[13] B. Orlandi, S. Lataste, S. Kerboeuf, M. Bouillon, X. Huang, O. Marcé, F. Faucheux, S. Cherrared, B. Mongazon, and P. Delvallet, "From a Business Intent to a Network Deployment with User-Friendly Interfaces and Natural Language Processing," in *2024 27th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, Mar. 2024, pp. 91–93.
[14] C. H. Cesila, R. P. Pinto, K. S. Mayer, A. F. Escallón-Portilla, D. A. A. Mello, D. S. Arantes, and C. E. Rothenberg, "Chat-IBN-RASA: Building an Intent Translator for Packet-Optical Networks based on RASA," in *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*. Madrid, Spain: IEEE, Jun. 2023, pp. 534–538.
[15] C. Shah, R. W. White, R. Andersen, G. Buscher, S. Counts, S. S. S. Das, A. Montazer, S. Manivannan, J. Neville, X. Ni, N. Rangan, T. Safavi, S. Suri, M. Wan, L. Wang, and L. Yang, "Using Large Language Models to Generate, Validate, and Apply User Intent Taxonomies," May 2024.
[16] B. Carrion, T. Onorati, P. Díaz, and V. Triga, "A taxonomy generation tool for semantic visual analysis of large corpus of documents," *Multimedia Tools and Applications*, vol. 78, no. 23, Dec. 2019.
[17] J. Lin, K. Dzeparoska, A. Tizghadam, and A. Leon-Garcia, "AppleSeed: Intent-Based Multi-Domain Infrastructure Management via Few-Shot Learning," in *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, Jun. 2023, pp. 539–544.
[18] J. Wang, L. Zhang, Y. Yang, Z. Zhuang, Q. Qi, H. Sun, L. Lu, J. Feng, and J. Liao, "Network Meets ChatGPT: Intent Autonomous Management, Control and Operation," *Journal of Communications and Information Networks*, vol. 8, no. 3, pp. 239–255, Sep. 2023.
[19] A. Mekrache, A. Ksentini, and C. Verikoukis, "Intent-Based Management of Next-Generation Networks: An LLM-centric Approach," *IEEE Network*, pp. 29–36, 2024.
[20] A. Collet, A. Banchs, and M. Fiore, "LossLeaP: Learning to Predict for Intent-Based Networking," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, May 2022, pp. 2138–2147.
[21] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Computing Research Repository (CoRR)*, no. arXiv:2201.11903, Jan. 2023.
[22] OpenAI, "GPT-4 Technical Report," Mar. 2024.
[23] G. Team, R. Anil, and S. Borgeaud, "Gemini: A Family of Highly Capable Multimodal Models," *Computing Research Repository (CoRR)*, no. arXiv:2312.11805, Jun. 2024.
[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *Computing Research Repository (CoRR)*, no. arXiv:1706.03762, Aug. 2023.
[25] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Jul. 2002, pp. 311–318.
[26] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, Jul. 2004, pp. 74–81.