DMO-GPT: An Intent-driven Framework for Distributed 6G Management and Orchestration

Abdelkader Mekrache, *Member, IEEE,* Adlen Ksentini, *Senior Member, IEEE.* Christos Verikoukis, *Senior Member, IEEE.*

***Abstract*—With the increasing demands for high quality of service (QoS) in 6G networks, managing these complex systems requires intelligent Operations Support Systems (OSSs). Standardization institutions such as ETSI and 3GPP mandate that OSSs enable end-to-end, cross-domain management across all 6G components. To this end, ongoing initiatives within these institutions are actively defining API frameworks to simplify interactions with OSSs, focusing on Intent-Based Networking (IBN) and Zero-touch network and Service Management (ZSM) APIs. However, the diversity of API standards across different OSSs presents significant challenges when managing 6G use cases across multiple Mobile Network Operators (MNOs). To this end, we propose in this paper a novel framework that leverages Large Language Models (LLMs) to enable natural language-based interactions with OSSs across multiple MNOs. To address the complexity of fulfilling user intents, which may involve multiple low-level API calls across heterogeneous OSSs, the proposed solution integrates multi-agent LLMs with an hierarchical planning mechanism. The framework offers two key advantages: a simple natural language-based interaction; and an adaptive system capable of autonomously accommodating new OSS (API) features. Real-world experiments validate the efficacy of the proposed framework, demonstrating its ability to efficiently manage diverse 6G OSSs and enhance the accessibility, interoperability, and automation of 6G network management.**

***Index Terms*—6G, OSS, AI, IBN, ZSM, LLMs.**

I. Introduction

The 6G era is expected to meet stringent Quality of Service (QoS) and Quality of Experience (QoE) requirements for advanced use cases such Augmented Reality (AR). These applications drive the need for infrastructure sharing among Mobile Network Operators (MNOs) to efficiently provide networking services [1]. However, managing these complex, multi-operator networks presents significant challenges. To address this, Artificial Intelligence (AI)-driven automation is being explored by standardization initiatives in institutions like ETSI and 3GPP, with a focus on Intent-Based Networking (IBN) and Zero-touch network and Service Management (ZSM). These institutions are actively working to standardize IBN and ZSM protocols and Application Programming Interfaces (APIs) within Operations Support Systems (OSS) to enable seamless and autonomous network management.

The IBN concept [2] defines networking goals and constraints at a high level using declarative intent. For instance, ETSI's Network Service Descriptor (NSD), enables the specification of networking services in an abstract manner, which is then implemented by the OSS [3]. Other MNOs may adopt different IBN standards (e.g., TMForum and 3GPP). Within OSS, intent is translated into low-level configurations, deployed, and continuously assured. This intent assurance mechanism verifies the validity of the intent and autonomously resolves anomalies without human intervention, enabling ZSM. However, in future 6G networks, where the infrastructure can be shared across multiple MNOs (each potentially following different IBN standards), learning all these different structures is both challenging and time-consuming for OSS users (e.g., vertical users, infrastructure providers). Therefore, natural language may be leveraged as the simplest form of intents, i.e., users can define their goals and constraints using natural language, and the IBN system will fulfill and ensure (ZSM) these intents, regardless of the standards used by the MNOs. This requires an AI approach that can understand human language, such as Large Language Models (LLMs).

LLMs are AI models that can tackle a variety of generic Natural Language Processing (NLP) tasks such as question answering, sentiment analysis, and code completion. These models are also increasingly used to build intelligent systems, known as agents, that can make important decisions based on natural language input (prompts), referred to as agentic systems. Some research has explored using LLMs for IBN, such as [4, 5, 6], but these typically address only one task within a single management system of one MNO using a simple intent, such as translation [4, 5] or assurance [6], corresponding to a single API call. These works cannot meet the demands of 6G management, where fully autonomous networking across different MNOs is required. This highlights the need for an intelligent system capable of performing all IBN tasks, spanning multiple API calls across a set of MNOs that follow different IBN standards.

In this paper, we present an intent-driven Distributed Management and Orchestration framework, DMO-GPT, the first LLM-powered solution capable of: (i) take a high-level natural language intent as input; (ii) select the MNOs required to fulfill the intent; (iii) plan the set of API calls for each MNO, (iv) execute them; and (v) report back to the users using natural language. This transforms the complexities of managing multiple MNO APIs into a single chatbot from the users' perspective, enabling them to request and interact using either natural language or voice, which can be translated into natural language using a Speech-to-Text (STT) model. The main contributions of the paper are:

- We proposed DMO-GPT, a chatbot for cross-domain 6G

A. Mekrache is with EURECOM, France (e-mail: abdelkader.mekrache@eurecom.fr).

A.Ksentini is with EURECOM, France (e-mail: adlen.ksentini@eurecom.fr).

C. Verikoukis is with ISI/ATH, University of Patras, Greece (e-mail: chverik@gmail.com).

management across different MNOs (as illustrated in Fig.1). Users can define intents using natural language, and the system fulfills them across multiple MNOs.
- The system is designed to leverage multi-agent LLMs that work collaboratively to: (i) accept natural language intents; (ii) select the appropriate MNOs; (iii) plan all necessary API calls for each MNO; (iv) execute these API calls sequentially; and (v) report the results back to the user using natural language.
- Most of the agents are designed to be trained using the API specifications of different MNOs through In-Context Learning (ICL). This also enables the system to adapt to new API calls and new MNOs at inference time.

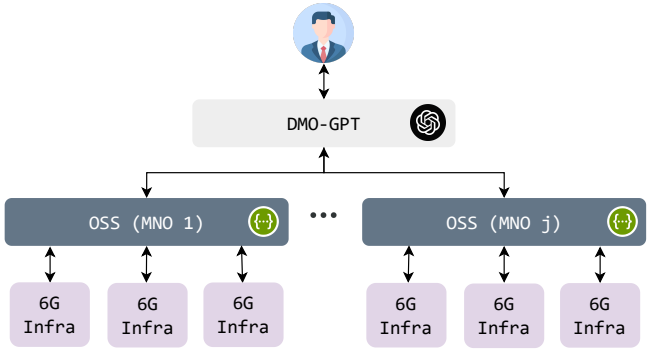

Fig. 1: Vision of DMO-GPT.

The remaining sections of this paper are structured as follows: Section II describes related works. In Section III, we illustrate the DMO-GPT design. Section IV showcases the experimentation setup and results. Section V presents future research directions. Finally, Section VI concludes the paper.

II. Related works and Background

In this section, we provide an overview of related works on API-driven network management and IBN. We also present a brief overview of LLMs. Finally, we identify the existing research gaps and position DMO-GPT within this context.

A. API-driven network management & IBN

Next-generation network management within the OSS layer spans multiple technological domains, including Radio Access Network (RAN), Transport Network (TN), Core Network (CN), and Edge/Cloud. These functionalities are accessible to various users, such as service owners and infrastructure providers, enabling them to manage services and networking resources efficiently. For instance, the Open RAN (O-RAN) Service Management and Orchestration (SMO) oversees the RAN domain, while the Network Function Virtualization Orchestrator (NFVO) handles the Edge/Cloud domain [5]. These systems rely heavily on programmable interfaces through APIs to communicate seamlessly with the infrastructure layer and each other. Additionally, ZSM is integrated to autonomously detect and resolve anomalies within services and infrastructure, eliminating the need for human intervention.

To simplify interactions with standardized APIs from organizations like ETSI, TMF, and 3GPP, IBN has emerged, allowing users to define intents using declarative structures like JSON or YAML. However, these formats are complex to learn. Recent research emphasizes using natural language for intent profiling, leveraging LLMs to translate intents into deployable configurations such as NSDs. For instance, the work in [4] proposed an LLM-based framework that translates natural language intents into NSDs for deployment on the Edge/Cloud domain. This approach was expanded in [5] to handle both RAN and Edge/Cloud domains, while others, such as [7], focused on generating TN configurations. Meanwhile, [6] focused only on intent assurance, enabling ZSM. Although promising, these approaches are limited to specific domains or simple tasks, leaving a gap in addressing complex intents across all 6G technological domains for multiple MNOs.

B. Large language models

LLMs are transformer-based models designed for general NLP tasks like question answering, sentiment analysis, and code completion. However, their effectiveness depends heavily on their training data, making general-purpose LLMs struggle with specialized tasks like network management. Adapting LLMs to such domains is an active research area. While supervised fine-tuning on domain-specific data is a common approach, it is computationally expensive and inaccessible to most users. To mitigate this, resource-efficient techniques such as ICL and Parameter-Efficient Fine-Tuning (PEFT) have emerged. ICL leverages context provided during inference without modifying model weights, e.g., zero-shot and few-shot learning, while PEFT adds small trainable parameters like LoRA [8], reducing training costs.

ICL is advantageous for creating agentic LLMs and multi-agent systems by dynamically defining agents' roles in context, making it suitable for remote or closed-source LLMs like GPT-4. These multi-agent LLMs have gained popularity in both research and industry for their efficiency in tackling complex tasks effectively [9]. However, they can increase inference time due to long contexts. In contrast, PEFT techniques like LoRA modify only a small portion of the model, offering faster inference (since they do not require context training and, hence, long contexts). In this paper, ICL is used for most agents, but LoRA is applied to create an LLM that generates the standardized structures, such as the NSDs from ETSI. This choice avoids the inefficiency of extended context lengths (when using ICL), as demonstrated in prior work [5].

C. Research gap

Existing solutions for natural language-based IBN typically address only simple tasks within a single MNO environment. As a result, they face significant limitations and challenges when instantiated in multi-OSS environments, particularly in addressing the interoperability required for diverse OSSs adhering to different standards. While methods like ReAct [10] and RestGPT [11] propose agentic LLMs for generating API calls, they primarily focus on single API specifications or isolated domains, making them insufficient for multi-domain 6G network management. A unified system capable of dynamically adapting to various API standards and fulfilling high-level intents across domains remains an unsolved challenge.

Our approach introduces a hierarchical planning and execution framework that leverages ICL to create LLM agents capable of transforming high-level intents into API calls spanning different MNOs. These agents handle planning, parameter preparation, API call execution, and natural language reporting. Additionally, the framework can dynamically adapt to new API specifications at inference time without requiring retraining. This enables seamless cross-domain management and fully automated network operations, effectively addressing key limitations in existing IBN methodologies.

III. DMO-GPT

In this section, we first provide a detailed explanation of our solution design, DMO-GPT. Next, we discuss how DMO-GPT manages different standardized APIs through LLM agents. Finally, we present several operational OSS-GPT use cases.

A. System design

The high-level architecture of DMO-GPT is illustrated in Fig. 2, which oversees a set of MNOs. Users send natural language intents to DMO-GPT, which consists of multiple LLM agents working collaboratively to plan and execute the appropriate API calls to the selected OSSs. These LLM agents are trained using ICL with the API specifications of each OSS. To manage context size limitations, all agents access a shared memory in the form of a centralized knowledge base storing API specifications for each OSS. During ICL, each agent retrieves only the relevant subset of this shared memory, enabling efficient prompt construction without overloading the context window. ICL provides a lightweight training strategy, allowing rapid adaptation to new API specifications or OSS updates at inference without additional overhead, and facilitating modular training across heterogeneous systems. In the following sections, we detail each agent's role and how it is trained using information from the shared knowledge base.

1) Assistant: The role of the *Assistant* is to maintain a chatbot-like interaction with the user. It accepts natural language input and responds with natural language output. This agent is trained using ICL with predefined guidelines and handles three key scenarios: (i) If the user asks general questions unrelated to OSS functions, the *Assistant* replies directly to the user's query; (ii) If the user provides an ambiguous or incomplete intent, the *Assistant* prompts the user for clarification or additional details; (iii) If the user provides a valid intent that can be executed by one or more OSSs, the *Assistant* forwards the intent to the *Router* and awaits the response from the *Overall Reporter* to deliver back to the user. This design ensures both fault tolerance and seamless interaction between users and DMO-GPT.

2) Router: The role of the *Router* is to determine the set of MNOs whose OSSs will handle the intent and forward it to the MNO-specific planning and execution LLMs, referred to as OSS-GPT in Fig. 2. OSS-GPT represents a set of LLM agents responsible for fulfilling an intent on the OSS associated with each MNO. The *Router* selects the appropriate ICL APIs specifications for training the OSS-GPT (during inference), ensuring that the correct context and API specifications are utilized. Since the *Router* operates with the descriptions of all MNOs, this remains manageable at moderate scale but may become a bottleneck in large-scale deployments. To address this, the *Router* use a retriever-based mechanism during ICL that filters relevant MNOs based on semantic similarity between the user intent and MNO descriptions. This reduces the prompt's context size and enhances system scalability.

3) Planner: The primary role of the *Planner* is to generate a sequence of API calls that effectively fulfill the user intent for a specific MNO. Upon receiving the intent from the *Assistant*, the *Planner* analyzes the request and identifies the necessary API calls to achieve the desired outcome. This process leverages ICL and a knowledge base containing concise descriptions of available API endpoints for the MNO. Due to the limitations in context size, the full specification of each endpoint is not included; instead, the *Planner* relies on summarized descriptions to select the relevant API calls. The *Planner* operates in an iterative manner, sending one API call at a time to the *Executor*. After the execution, the *Planner* adapts its approach based on the outcome: if successful, it generates the next API call in the sequence; otherwise, the *Planner* dynamically replans to adjust the sequence of API calls, ensuring that the intent is ultimately fulfilled. Once the objective is achieved, the *Planner* invokes the *Reporter* to provide a natural language summary of the execution results. This adaptive mechanism ensures the system remains robust, even under unpredictable conditions or partial failures.

4) Executor: The *Executor* is responsible for executing API calls provided by the *Planner* and returning their results for a specific MNO. It is trained using ICL to utilize detailed descriptions of the available endpoints for execution guidance. Upon receiving an API call, the *Executor* plans the necessary tools for execution. While additional tools may be utilized, it primarily depends on four core tools:

- *Blueprint Generator*: An LLM specialized in generating request bodies for POST operations, such as NSD, based on natural language inputs. The design and training of this LLM are detailed in section III-B.
- *Blueprint Explorer*: A program that retrieves existing data for PUT requests by performing GET operations, ensuring the inclusion of relevant context in PUT requests.
- *API Caller*: A program that executes API calls by sending HTTP requests with the required parameters (headers, body, query, etc.) to one OSS.
- *Human Validation*: A mechanism to obtain user approval for critical operations (e.g., POST, PUT, and DELETE requests). Users can enable or disable this feature, offering a balance between autonomy and human oversight.

In the case of an intent to create a 6G service, such as a 6G CN, and ensure a specific throughput, the *Executor* first utilizes the *Blueprint Generator* to construct the request body. It then invokes the *API Caller* to send the request to the endpoint for creating the 6G CN. Upon receiving a response, the *Executor* verifies that it adheres to the expected schema. If the response is successful, the *Executor* requests again the *API Caller* to send another request to the endpoint responsible for deploying ZSM components, that will ensure the SLA. If this request is successful, the *Executor* forwards the result to the *Planner*.

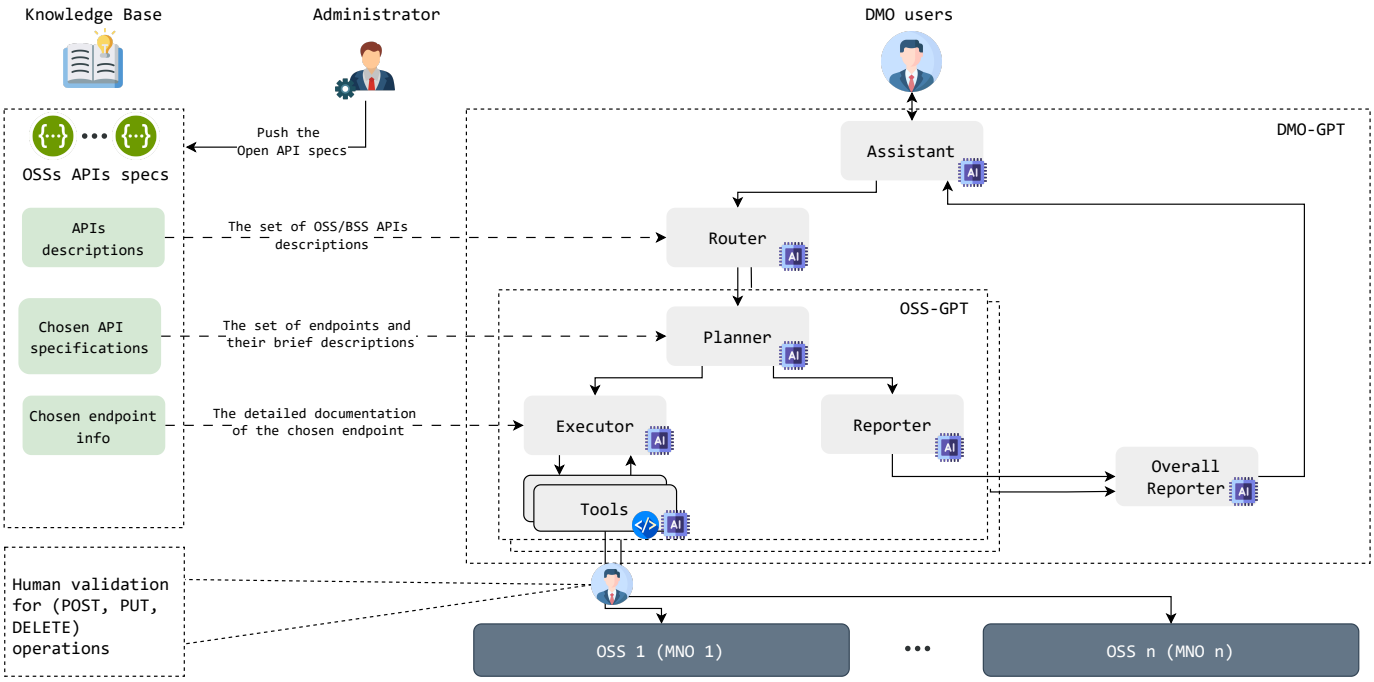

Fig. 2: System design.

5) Reporter: The *Reporter* agent is an LLM designed to generate a natural language report based on the conversation history between the *Planner* and *Executor*. Leveraging ICL, it processes this input and provides the generated response to the *Overall Reporter* agent.

6) Overall Reporter: The *Overall Reporter* agent aggregates reports from individual *Reporters*, providing a unified context. It then generates a comprehensive response, summarizing the outcomes from all OSSs. To support auditability, this agent logs each API request, along with the user identity, timestamp, and response. This ensures full traceability and allows integration with external monitoring tools.

B. Design and training of standard-specific tools

In the context of the *Executor* agent, the *Blueprint Generator* is an LLM-based tool that generates request structures compliant with specific standards. This tool is standards-specific, as each OSS may adhere to different specifications such as ETSI, 3GPP, or TMF. For example, ETSI defines the NSD structure to describe the services to be deployed and managed. Similarly, 3GPP and TMF provide other specifications for service and resource modeling. To support this process, the Blueprint Generator must be trained on datasets that reflect the syntax, structure, and semantics of the relevant standards. In our implementation, we focus on the ETSI NSD standard, and we developed the *NSD-expert*, an LLM fine-tuned using the LoRA technique [12], specifically designed to generate NSDs from natural language inputs. When the *Executor* agent invokes this tool, it processes natural language intents and outputs the corresponding NSD.

The training process began with dataset generation using an ICL approach. We leveraged existing examples from [4], where the evaluation involved applying the proposed ICL technique to generate NSDs from natural language intents. The experimenters manually scored the results and selected the correct NSDs, which were then stored to form the base dataset, consisting of 100 intent–NSD pairs. To augment this dataset, we used GPT-4 with few-shot prompting to generate an additional 400 entries. Specifically, we provided the model with a few representative examples (i.e., intents paired with their corresponding NSDs) from the initial dataset, which it used to generate new samples. All generated entries were validated by domain experts before being added to the dataset, resulting in a final training set of 500 examples. For fine-tuning, we employed a PEFT technique known as LoRA [8]. LoRA introduces trainable low-rank matrices into a frozen pre-trained language model, allowing it to adapt to the task of NSD generation while preserving its general language capabilities. This method enabled us to specialize the model in generating valid NSD structures based on natural language intents without retraining the full model from scratch.

Our approach is generalizable and can be extended to other standards beyond ETSI, such as TMF and 3GPP specifications, by training new domain-specific experts on datasets derived from those standards. This would enable the *Blueprint Generator* to support a broader range of OSSs and reinforce its adaptability across heterogeneous management systems.

C. Operational DMO-GPT Use Cases

OSS-GPT is designed to support a wide range of operational scenarios in production environments by enabling users to express high-level intents in natural language. These intents are translated into appropriate API calls exposed by the underlying OSSs. Before executing any action, OSS-GPT enforces strict access control through token-based authorization. Each API request is validated by the OSS-level authorization service,

which verifies whether the user is permitted to perform the action. These permissions are configured by internal MNO administrators who act as governance controllers. This mechanism ensures that users can only perform operations for which they are explicitly authorized. In addition, all interactions are audited by the *Overall Reporter* agent, which logs every request along with user identity and timestamp. This enables full traceability and facilitates integration with external auditing tools. Beyond access control and auditability, DMO-GPT integrates failure recovery and trust management mechanisms to ensure safe autonomous infrastructure changes. The *Planner* monitors the outcome of each API call and dynamically replans in response to errors. If repeated attempts fail, execution is halted and the user is notified with a detailed diagnostic report. To minimize risk, all LLM agents operate in a controlled runtime environment with safeguards such as schema validation, rate limiting, and request verification. Below, we outline representative use cases illustrating how OSS-GPT assists operators in managing service lifecycles, monitoring system health, and handling faults:

- (i) An operator may request, "Update the service template to version 2.3," and OSS-GPT translates this into a version-aware API call (e.g., `PUT /service/id`), ensuring that versioning constraints, and rollback policies are respected via the OSS APIs.
- (ii) A domain expert may ask, "Is the edge cluster deployment running normally?" OSS-GPT queries monitoring endpoints to retrieve health status and operational KPIs, then provides a natural language summary of results.
- (iii) In the event of service degradation, a user may state, "Ensure the latency constraint for service X to be Y ms." OSS-GPT calls the relevant OSS endpoint that triggers deployment of ZSM components [6], leveraging AI techniques for root cause analysis and recommending corrective actions such as resource scaling.

IV. Performance Evaluation

In this section, we first present the evaluation setup, followed by the results. These results include: the evaluation of the *NSD-expert*, the quality assessment of DMO-GPT, and finally, the cost assessment of DMO-GPT.

A. Evaluation setup

Our experimental setup, illustrated in Fig. 3, involves three machines hosting the Kubernetes-based clusters and the OSS-GPT framework. Two machine, equipped with an Intel(R) Xeon(R) Silver 4314 CPU (2.40GHz), manages two single-node Kubernetes clusters that hosts the virtualized 6G services and infrastructure components. The second machine, a MacBook with M1 Pro chip, hosts the infrastructures management components, including the two OSSs and DMO-GPT, running on Docker and bare-metal, respectively. The DMO-GPT is implemented using Langgraph[1], and interacts with two LLMs: OpenAI's GPT-4[2], accessed remotely, and *NSD-expert*, a locally deployed model with 8B parameters running on the Ollama framework[3]. This *NSD-expert* was fine-tuned on a third machine equipped with an NVIDIA A100 GPU, using Llama3.2-3B[4] as the base model with LoRA. For training, we employed the Unsloth framework[5] with a training setup that included 120 steps, and learning rate of 2e-4

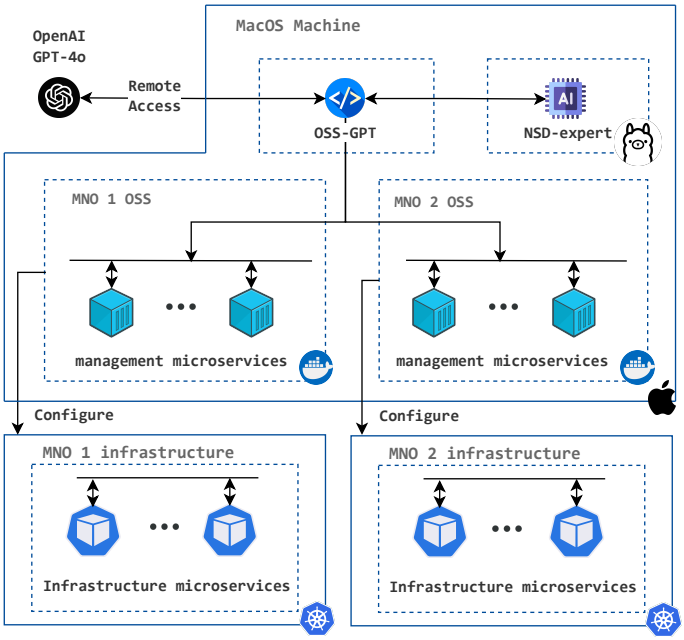

Fig. 3: Evaluation setup.

B. Evaluation results

To evaluate the quality of the *NSD-expert*, we conducted an extensive benchmark using a test set of 100 natural language intents, and reference NSDs. We evaluated model outputs using four key metrics: (i) Perplexity, which measures the model's confidence in predicting the next token in a sequence (lower values indicate higher confidence); (ii) BERTScore, which assesses semantic similarity between the generated and reference texts (higher scores indicate greater similarity); (iii) Cosine Similarity, computed using Sentence-BERT[6], quantifies similarity between vector representations of texts; and (iv) Exact Match (EM), which computes the percentage of outputs that exactly match the reference, particularly suitable for structured data like JSON. As shown in Fig. 4, the *NSD-expert* achieves a low perplexity, reflecting confident token generation. While general-purpose models like Llama3.2 and GPT-4 also exhibit low perplexity, this does not necessarily correlate with structurally valid output. Indeed, the structure-aware metrics (Cosine Similarity, BERTScore, and EM) offer a more reliable evaluation of NSD correctness. Llama3.2 and GPT-4 scored poorly on these metrics: their outputs had low Cosine Similarity and BERTScore (typically below 0.80) and failed to generate valid NSDs, resulting in a 0% EM score. In contrast, the *NSD-expert* achieved high semantic similarity

[1]https://www.langchain.com/langgraph
[2]gpt-4o-2024-08-06
[3]https://ollama.com
[4]https://ollama.com/library/llama3.2
[5]https://unsloth.ai
[6]all-MiniLM-L6-v2

(Cosine ≈ 0.98, BERTScore ≈ 0.99) and a substantially better EM score of 60%, indicating correct structure in a majority of cases. These results highlight the effectiveness of our domain-specific fine-tuning in producing a reliable model that can generate syntactically and semantically accurate NSDs. The *NSD-expert* can thus be reliably used by the *Executor* agent to autonomously generate NSDs in response to user intents.

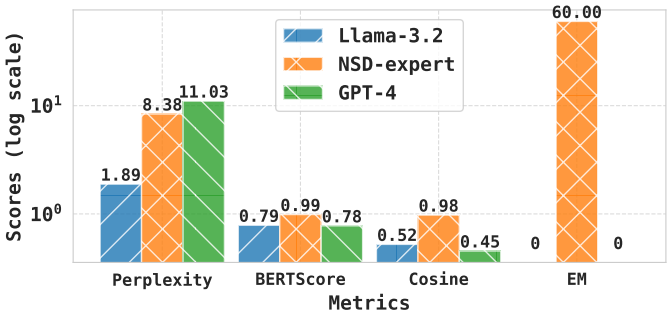

Fig. 4: Quality assessment of the *NSD-expert*.

To evaluate DMO-GPT's performance, we created a dataset of intents ranging from simple to complex, requiring up to 7 API calls (golden path). A golden path of $i = 0$ corresponds to cases where DMO-GPT can generate a response directly using its internal knowledge without invoking any API. Figure 5 presents two evaluation metrics: (i) Deviation (orange), which measures the number of redundant API calls made beyond the golden path; and (ii) Accuracy (blue), defined as the percentage of intents that are ultimately fulfilled correctly. For intents requiring fewer than 4 API calls ($i < 4$), DMO-GPT achieves near-perfect accuracy (close to 1.0). As the complexity of the intent increases ($i > 4$), accuracy decreases slightly, primarily due to occasional errors in selecting the correct API endpoints. This is reflected in the deviation metric, which shows that DMO-GPT sometimes performs more API calls than necessary before reaching the correct execution path. Nevertheless, DMO-GPT's replanning mechanism enables it to recover from such errors, maintaining a strong performance with an accuracy exceeding 0.80 even for intents with a golden path length of 7. This behavior highlights DMO-GPT's robustness and reliability in fulfilling complex, multi-step intents in cross-domain network management scenarios.

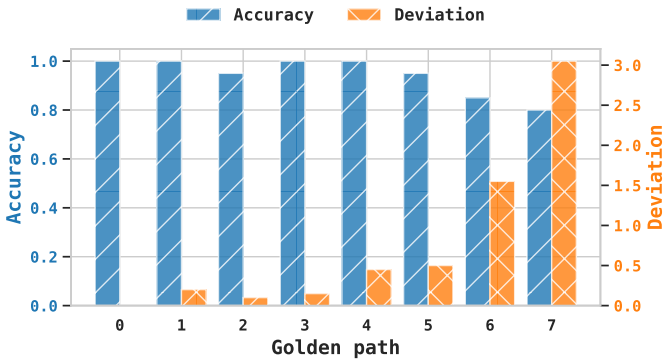

Fig. 5: Quality assessment of DMO-GPT.

The cost evaluation of DMO-GPT highlights its affordability for both training and inference: (i) Training: Training the *NSD-expert* was required only 3.07 minutes and 4.006 GB of vRAM on a single NVIDIA A100 GPU with 40 GB of vRAM; (ii) Inference: The cost was measured by execution time and OpenAI's GPT-4 pricing. As illustrated in Fig. 6, for golden paths up to 7, response times remained under 80 seconds, while costs were \$0.175 per request, making it affordable even for complex intents requiring multiple API calls. Notably, OpenAI's pricing is decreasing, and advancements in open-source models are rapidly narrowing the performance gap with closed-source alternatives, paving the way for cost-effective open-source solutions in the near future. However, both latency and cost increase exponentially beyond a golden path length of 5, confirming that DMO-GPT performs more API calls than necessary. This leads to more errors and triggers additional replanning. These results indicate the existence of a gap of improvement in terms of the planning abilities of LLMs.

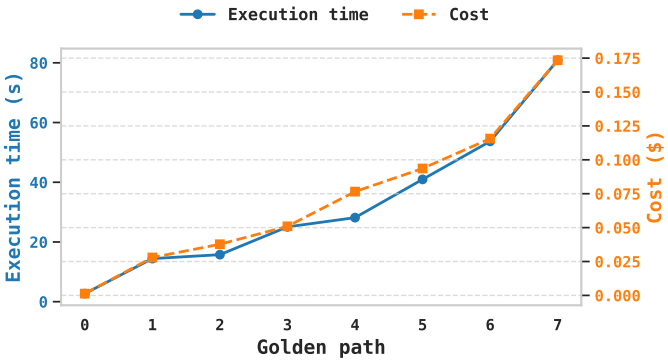

Fig. 6: Cost assessment of DMO-GPT.

V. Limitations and Future Directions

Our work demonstrates the potential of multi-agent LLMs for real-world planning and execution, enabling IBN in 6G network management. However, several challenges arise: (i) LLMs are time-consuming and energy-intensive, emphasizing the need for faster, more energy-efficient solutions; (ii) reliance on OpenAI's closed-source LLMs for structured output highlights the necessity for powerful open-source alternatives to reduce the inference costs; and (iii) planning errors reveal the need for more advanced models that move closer to the capabilities of Artificial General Intelligence (AGI).

A. Fast and green intelligence with SLMs

While our work employed a large LLM, i.e., GPT-4, such models are computationally intensive. Smaller Language Models (SLMs), with fewer parameters (e.g., 1B-4B), offer a more energy-efficient and environmentally friendly alternative, though they currently fall short in handling complex tasks. To evaluate this trade-off, we tested two open-source SLMs: Llama3.2-3B[7], a general-purpose model, and Qwen3-4B[8],

[7]https://ollama.com/library/llama3.2
[8]https://ollama.com/library/qwen3:4b

optimized for reasoning. Both struggled with multi-step intent fulfillment. Llama3.2 achieved 0.82 accuracy for golden path length 0, 0.11 for length 1, and 0 beyond that, often producing incorrect structured outputs. Qwen3 performed slightly better, 1.0 for path 0, 0.857 for path 1, 0.12 for path 2, but also dropped to 0 beyond that. While Qwen3 generated more structured outputs, it suffered from indecisiveness in endpoint selection. These results confirm that current SLMs, though promising for low-power scenarios, still lack the reasoning capabilities needed for multi-step intent fulfillment in IBN compared to GPT-4 (see Fig. 5). Further research is required to improve their accuracy and reduce generation time, especially for applications demanding fast responses like URLLC [13].

B. LLMs structured outputs

ICL enables training LLMs during inference, as demonstrated in DMO-GPT agents. In agentic frameworks, LLMs must generate outputs in a structured format that is understandable by other LLMs, a concept referred to as "structured output," as discussed in [14]. This concept is critical, as any error in the structure generated by an LLM can cause the entire multi-agent process to fail. According to [14], GPT-4 is recognized as the leading model in structured output generation. However, it is a closed-source model and not freely accessible, which limit its adoption for some users. Therefore, the community should focus on developing customized telecom-aware alternatives that are capable of producing high-quality structured outputs to close the gap with GPT-4.

C. Artificial general intelligence

In modern 6G networks, intents are anticipated to become more complex, as autonomous management needs to be as abstracted as possible. As shown in Fig. 5, as the complexity of intents increases, represented by the growing size of the golden path, the deviation from the optimal path tends to increase. This observation suggests that as tasks become more complex, the planner and executor LLMs exhibit a higher frequency of planning errors. These results indicate that the current capabilities of LLM-based planning systems need enhancement, especially for complex and multi-step tasks. Ongoing research is focused on addressing these challenges, with the ultimate goal of advancing toward AGI, where systems can effectively handle a broader spectrum of tasks with minimal errors [15].

VI. Conclusion

This paper presented a multi-agent LLM-based framework to simplify 6G network management by enabling natural language, intent-driven interactions with diverse OSSs. Through hierarchical planning and execution, the proposed solution efficiently manages cross-domain API calls, enabling an advanced, fully natural language-driven IBN and ZSM in 6G. Real-world experiments validated the approach, demonstrating enhanced accessibility and automation in managing multiple 6G OSSs. Future work should address key challenges, including improving LLM efficiency, enhancing structured output generation in open-source models, and refining planning accuracy, paving the way for more intelligent, scalable, and autonomous 6G management systems powered by AGI.

Acknowledgment

This work is supported by the European Union's Horizon Program under the Sunrise-6G project (Grant No. 101139257).

References

[1] Wei Jiang et al. "The road towards 6G: A comprehensive survey". In: *IEEE Open Journal of the Communications Society* 2 (2021), pp. 334–366.

[2] Aris Leivadeas and Matthias Falkner. "A survey on intent-based networking". In: *IEEE Communications Surveys & Tutorials* 25.1 (2022), pp. 625–655.

[3] Sagar Arora, Adlen Ksentini, and Christian Bonnet. "Lightweight edge slice orchestration framework". In: *ICC 2022-IEEE International Conference on Communications*. IEEE. 2022, pp. 865–870.

[4] Abdelkader Mekrache, Adlen Ksentini, and Christos Verikoukis. "Intent-based management of next-generation networks: An LLM-centric approach". In: *Ieee Network* (2024).

[5] Abdelkader Mekrache and Adlen Ksentini. "LLM-enabled intent-driven service configuration for next generation networks". In: *2024 IEEE 10th International Conference on Network Softwarization (NetSoft)*. IEEE. 2024, pp. 253–257.

[6] Abdelkader Mekrache et al. "On Combining XAI and LLMs for Trustworthy Zero-Touch Network and Service Management in 6G". In: *IEEE Communications Magazine* (2024).

[7] Ahlam Fuad et al. "An intent-based networks framework based on large language models". In: *2024 IEEE 10th International Conference on Network Softwarization (NetSoft)*. IEEE. 2024, pp. 7–12.

[8] Edward J Hu et al. "Lora: Low-rank adaptation of large language models". In: *arXiv preprint arXiv:2106.09685* (2021).

[9] Junyou Li et al. "More agents is all you need". In: *arXiv preprint arXiv:2402.05120* (2024).

[10] Shunyu Yao et al. "React: Synergizing reasoning and acting in language models". In: *arXiv preprint arXiv:2210.03629* (2022).

[11] Yifan Song et al. "RestGPT: Connecting Large Language Models with Real-World RESTful APIs". In: *arXiv preprint arXiv:2306.06624* (2023).

[12] Yuren Mao et al. "A survey on lora of large language models". In: *arXiv preprint arXiv:2407.11046* (2024).

[13] Piotr Nawrot et al. "Dynamic memory compression: Retrofitting llms for accelerated inference". In: *arXiv preprint arXiv:2403.09636* (2024).

[14] Yu Liu et al. "Are LLMs good at structured outputs? A benchmark for evaluating structured output capabilities in LLMs". In: *Information Processing & Management* 61.5 (2024), p. 103809.

[15] Fabrizio Davide, Pietro Torre, and Andrea Gaggioli. "AI Predicts AGI: Leveraging AGI Forecasting and Peer Review to Explore LLMs' Complex Reasoning Capabilities". In: *arXiv preprint arXiv:2412.09385* (2024).

BIOGRAPHIES

Abdelkader Mekrache (Member, IEEE) is a PhD candidate at EURECOM's Communication Systems Department. His primary focus is on advanced network management frameworks in next-generation wireless networks under the supervision of Prof. Adlen Ksentini. He is an active participant in collaborative research and notably contributes to the OAI project, as well as multiple European projects, including 6G-Bricks, 6G-Intense, and Sunrise-6G.

Adlen Ksentini (Senior Member, IEEE) is a professor at EURECOM in the Communication Systems Department, leading the Network Softwarization group. His research focuses on network virtualization, SDN, and edge computing for 5G/6G networks. He has participated in several H2020 and Horizon Europe projects and is the technical manager for 6G-Intense and AC3. Adlen has given tutorials at major IEEE conferences and is a member of the OAI board of directors.

Christos Verikoukis (Senior Member, IEEE) received his Ph.D. degree from the Technical University of Catalonia (UPC), Barcelona, Spain, in 2000. He is currently a Professor with the University of Patras and a Collaborating Faculty member with ISI/ATH, Patras . He has authored 170 journal articles and over 250 conference papers. He has participated in more than 60 competitive projects and he has coordinated 15 of them. He is currently the IEEE ComSoc GITC member and the Editor-in-Chief of the IEEE NETWORKING LETTERS.