Sorbonne University

Doctoral School of Informatics, Telecommunications and Electronics of Paris

EURECOM

Generative AI for Next-Generation Intent-Based Networking in 6G

Presented by **Abdelkader Mekrache**

Dissertation for Doctor of Philosophy in Information and Communication Engineering

Directed by **Prof. Adlen Ksentini**
Co-directed by **Prof. Ulrich Finger**

The Jury committee is composed of:

Prof.	Barbara Martini	Universitas Mercatorum	Reviewer
Prof.	Jérôme François	University of Luxembourg	Reviewer
Prof.	Hakima Chaouchi	Télécom Sud Paris	Examiner
Dr.	Philippe Bertin	Orange	Examiner
Prof.	Jérôme Harri	Eurecom	Jury President
Prof.	Adlen Ksentini	Eurecom	Thesis Director
Prof.	Ulrich Finger	Eurecom	Thesis Co-director

Sophia Antipolis, January 19th, 2026

Sorbonne Université

Ecole doctorale Informatique, télécommunications et électronique de Paris

EURECOM

IA générative pour les réseaux à base d'intentions de prochaine génération dans les réseaux 6G

Présenté par **Abdelkader Mekrache**

Thèse de doctorat en Sciences de l'Information et de la Communication

Dirigée par	**Prof. Adlen Ksentini**
Co-dirigée par	**Prof. Ulrich Finger**

Le jury est composé de :

Prof.	Barbara Martini	Universitas Mercatorum	Rapporteure
Prof.	Jérôme François	Université du Luxembourg	Rapporteur
Prof.	Hakima Chaouchi	Télécom Sud Paris	Examinatrice
Dr.	Philippe Bertin	Orange	Examinateur
Prof.	Jérôme Harri	Eurecom	Président du Jury
Prof.	Adlen Ksentini	Eurecom	Directeur de thèse
Prof.	Ulrich Finger	Eurecom	Co-directeur de thèse

Sophia Antipolis, 19 Janvier, 2026

Abstract

The advent of beyond 5th Generation (5G) and the upcoming 6th Generation (6G) era introduces unprecedented complexity in managing heterogeneous, large-scale, and dynamic network infrastructures. To cope with this complexity, Intent-Based Networking (IBN) has emerged as a paradigm where vertical users express high-level objectives, or intents, which are then automatically translated and enforced by the network [1]. Despite its promise, current IBN solutions remain constrained by their reliance on rigid, structured specifications (e.g., JavaScript Object Notation (JSON), Yet Another Markup Language (YAML)) and limited autonomous assurance mechanisms. As a result, intents are difficult for non-expert users to express correctly, and existing systems struggle to continuously ensure alignment between user goals and network behavior.

This thesis explores how Generative AI (GenAI), and in particular Large Language Models (LLMs), can serve as key enablers of next-generation IBN frameworks. GenAI models have recently demonstrated remarkable capabilities in natural language understanding, reasoning, and adaptability, making them highly suitable for bridging the gap between human-centric intent specification and machine-level execution [2]. Our research addresses three major challenges in this direction: (i) intent translation, (ii) intent assurance, and (iii) GenAI operations in IBN.

For intent translation, we propose LLM-enabled frameworks that allow intents to be expressed directly in natural language and automatically translated into deployable network configurations. We first focus on the configuration of network services across multiple technological domains, namely the Radio Access Network (RAN), Core Network (CN), and Edge/Cloud. Our approach demonstrates that LLMs can effectively convert user-friendly intents into service descriptors, thereby lowering the entry barrier for non-expert users. Achieving this requires decomposing complex intents into coordinated, domain-specific tasks and managing the full intent life cycle, including negotiation, configuration, monitoring, and assurance. We then generalize the translation problem to encompass Operations Support System (OSS)-level interactions, where a chatbot-like system plans and executes multiple Application Programming Interface (API) calls to fulfill high-level intents across heterogeneous domains.

For intent assurance, we investigate how GenAI can ensure that deployed services continuously meet their intended objectives. We design a modular Network Data Analytics Function (NWDAF)-based architecture that integrates Machine Learning (ML) models for anomaly detection in User Equipment (UE) traffic, validated on real-world testbeds. Building upon this foundation, we propose a trustworthy closed-loop assurance pipeline that integrates Artificial Intelligence (AI), eXplainable AI (XAI), and LLMs. This framework not only detects and resolves anomalies autonomously but also explains root causes and corrective actions in natural language, thus enhancing transparency and operator trust which are critical for the vision of Zero-touch network and Service Management (ZSM).

Finally, we address the challenge of GenAI operations in IBN, recognizing that LLMs must be both domain-specialized and efficiently orchestrated. To this end, we introduce 5G INSTRUCT Forge, a data engineering pipeline that systematically extracts knowledge from 3rd Generation Partnership Project (3GPP) Technical Specifications (TSs) to build Telecom-aware LLMs. These models significantly outperform general-purpose LLMs on Telecom-specific tasks, confirming the importance of domain adaptation. Furthermore, we propose a Deep Reinforcement

Learning (DRL)-based scheduler that dynamically allocates LLM tasks under strict Service Level Objectives (SLOs), ensuring scalability and efficiency when multiple applications compete for limited AI resources.

The frameworks developed in this thesis were validated through extensive prototypes and experimental deployments in real 5G/6G environments, including the EURECOM 5G testbed [3]. Results demonstrate significant improvements in intent usability, translation accuracy, assurance reliability, interpretability, and resource efficiency compared to state-of-the-art approaches.

In summary, this thesis establishes GenAI as a cornerstone technology for 6G IBN. By enabling natural language intent specification, autonomous and trustworthy assurance, and scalable AI operations, our contributions bring IBN closer to large-scale, real-world adoption. The work paves the way for future 6G networks that are more user-centric, autonomous, and resilient, while also opening new research directions in federated intent management, robust AI trustworthiness, and multi-operator collaboration.

Résumé

L'avènement des réseaux de cinquième génération (5G) et l'émergence prochaine de la sixième génération (6G) introduisent une complexité sans précédent dans la gestion d'infrastructures réseau hétérogènes, à grande échelle et dynamique. Pour faire face à cette complexité, le paradigme de réseaux pilotés par intention (IBN) a été proposé. Il permet aux utilisateurs verticaux d'exprimer des objectifs de haut niveau, ou intents, qui sont ensuite automatiquement traduits et appliqués par le réseau [1]. Malgré son potentiel, les solutions IBN actuelles restent limitées par leur dépendance à des spécifications rigides et structurées (par exemple JSON, YAML), ainsi que par des mécanismes d'assurance autonome restreints. En conséquence, les intents sont difficiles à formuler correctement pour des utilisateurs non experts, et les systèmes existants peinent à garantir en continu l'alignement entre les objectifs exprimés et le comportement réel du réseau.

Cette thèse explore comment l'intelligence artificielle générative (GenAI), et plus particulièrement les modèles de langue de grande taille (LLMs), peut constituer un levier clé pour les architectures IBN de nouvelle génération. Les modèles de GenAI ont récemment démontré des capacités remarquables en compréhension du langage naturel, en raisonnement et en adaptabilité, les rendant particulièrement adaptés pour combler le fossé entre la spécification centrée sur l'humain et l'exécution au niveau machine [2]. Nos recherches abordent trois défis majeurs : (i) la traduction d'intents, (ii) l'assurance d'intents, et (iii) l'exploitation opérationnelle de la GenAI dans l'IBN.

Pour la traduction d'intents, nous proposons des architectures basées sur les LLMs permettant d'exprimer les intents directement en langage naturel et de les traduire automatiquement en configurations réseau déployables. Nous nous concentrons d'abord sur la configuration des services réseau à travers plusieurs domaines technologiques, à savoir les réseaux d'accès radio (RANs), les cœurs de réseau (CNs) et l'Edge/Cloud. Notre approche montre que les LLMs peuvent convertir efficacement des intents formulés de manière conviviale en descripteurs de service, réduisant ainsi les barrières à l'entrée pour les utilisateurs non experts. Cela nécessite de décomposer des intents complexes en tâches coordonnées et spécifiques à chaque domaine, tout en gérant le cycle de vie complet de l'intent, incluant la négociation, la configuration, la supervision et l'assurance. Enfin, nous généralisons le problème de la traduction au niveau des interactions avec les systèmes de support aux opérations (OSSs), où un système de type agent conversationnel planifie et exécute plusieurs appels d'interface de programmation applicative (API) afin de satisfaire des intents de haut niveau dans des environnements hétérogènes.

Pour l'assurance d'intents, nous étudions comment la GenAI peut garantir que les services déployés respectent en permanence les objectifs définis. Nous concevons une architecture modulaire basée sur la fonction d'analyse des données réseau (NWDAF), intégrant des modèles d'apprentissage automatique (ML) pour la détection d'anomalies dans le trafic des équipements utilisateurs (UEs), validée sur des environnements expérimentaux réels. Sur cette base, nous proposons un pipeline d'assurance en boucle fermée intégrant AI, AI explicable (XAI) et LLMs. Ce cadre permet non seulement de détecter et résoudre les anomalies de manière autonome, mais également d'expliquer les causes racines et les actions correctives en langage naturel, renforçant ainsi la transparence et la confiance des opérateurs, essentielles pour la vision de la gestion autonome et sans intervention des réseaux et services (ZSM).

Enfin, nous abordons le défi des opérations de la GenAI dans l'IBN, en reconnaissant que les LLMs doivent être à la fois spécialisés dans le domaine et orchestrés efficacement. À cet effet, nous présentons 5G INSTRUCT Forge, un pipeline de traitement de données qui extrait systématiquement les connaissances des spécifications techniques (TSs) du projet de partenariat de 3ème génération (3GPP) afin de créer des LLMs adaptés aux télécommunications. Ces modèles surpassent significativement les LLMs généralistes sur des tâches spécifiques aux télécommunications, confirmant l'importance de l'adaptation au domaine. De plus, nous proposons un ordonnanceur basé sur l'apprentissage par renforcement profond (DRL), qui alloue dynamiquement les tâches aux LLMs sous des contraintes strictes des objectifs de niveau de service (SLOs), garantissant ainsi scalabilité et efficacité lorsque plusieurs applications partagent des ressources AI limitées.

Les architectures développées dans cette thèse ont été validées à travers des prototypes et des déploiements expérimentaux étendus dans des environnements 5G/6G réels, y compris sur l'l'infrastructure 5G d'EURECOM [3]. Les résultats démontrent des améliorations significatives en termes de facilité d'usage des intents, de précision de traduction, de fiabilité de l'assurance, d'interprétabilité et d'efficacité des ressources par rapport aux approches existantes.

En résumé, cette thèse établit la GenAI comme une technologie clé pour l'IBN en 6G. En permettant la spécification des intents en langage naturel, une assurance autonome et fiable, et des opérations AI évolutives, nos contributions rapprochent l'IBN d'un déploiement à grande échelle dans des environnements réels. Ce travail ouvre la voie à des réseaux 6G plus centrés sur l'utilisateur, autonomes et résilients, tout en proposant de nouvelles perspectives de recherche sur la gestion fédérée des intents, l'AI responsable et la collaboration multi-opérateurs.

Acknowledgements

"Whoever seeks high goals must endure sleepless nights."

Since my earliest days, I have aspired to grow in knowledge and understanding. Advancing research has been a dream come true. From a young age, I admired the brilliance of people from my country who excelled abroad, and I found it deeply inspiring to dedicate one's life to the pursuit of knowledge. This thesis is the result of the research I have conducted over the past three years, and achieving this milestone would not have been possible without the support of many remarkable people in my life.

I owe my deepest gratitude to my parents, who have supported me unconditionally since the very beginning of this journey. They provided me with more support than I could ever have imagined, always putting my needs before their own. Throughout this journey, they have made countless sacrifices for us, showing patience, strength, and love at every step, and I would not be here today without them. I am equally grateful to my brothers, Amine and Redouane, who constantly pushed me forward and never hesitated to stand by me whenever I was in need.

I extend my heartfelt thanks to Professor Adlen Ksentini, who played a decisive role in my path to research. Without his encouragement, support, and guidance, I would never have reached this stage. I deeply admire the way he creates opportunities for his students to learn and to bring out the best version of themselves. He is, to me, a true example of a successful professor, combining strong academic depth with remarkable technical expertise. I would also like to warmly thank Professor Christos Verikoukis, for his invaluable support and for sharing with us his strategic insights throughout this journey.

I am sincerely grateful to my friends, who made this journey not only fruitful but also enjoyable. Special thanks go to Karim Boutiba, whose warm welcome made my first days at EURECOM so much easier. I would also like to thank Mohammed, Sofiane, Giulio, Akram, Ayoub, Salim, Mazene, Franck, and Abdelghani for their friendship and support.

Finally, I would like to thank the entire EURECOM family, as well as everyone who, from near or far, has contributed to shaping this work.

Remerciements

"Celui qui vise de hauts objectifs doit endurer des nuits sans sommeil."

Depuis mes premiers jours, j'ai aspiré à grandir en savoir et en compréhension. Avancer dans la recherche a été la réalisation d'un rêve. Depuis mon jeune âge, j'ai admiré l'excellence des personnes de mon pays qui ont réussi à l'étranger, et j'ai trouvé profondément inspirant de consacrer sa vie à la quête du savoir. Cette thèse est le fruit des recherches que j'ai menées au cours des trois dernières années, et atteindre ce jalon n'aurait pas été possible sans le soutien de nombreuses personnes remarquables dans ma vie.

J'exprime ma plus profonde gratitude à mes parents, qui m'ont soutenu inconditionnellement depuis le tout début de ce parcours. Ils m'ont apporté plus de soutien que je n'aurais jamais pu l'imaginer, plaçant toujours mes besoins avant les leurs. Tout au long de ce chemin, ils ont fait d'innombrables sacrifices pour nous, faisant preuve de patience, de force et d'amour à chaque étape, et je ne serais pas ici aujourd'hui sans eux. Je suis également reconnaissant envers mes frères, Amine et Redouane, qui m'ont constamment encouragé et n'ont jamais hésité à me soutenir dans les moments difficiles.

Je remercie chaleureusement le Professeur Adlen Ksentini, qui a joué un rôle décisif dans mon parcours de recherche. Sans son encouragement, son soutien et ses conseils, je n'aurais jamais atteint ce stade. J'admire profondément sa manière de créer des opportunités pour ses étudiants afin qu'ils apprennent et donnent le meilleur d'eux-mêmes. Il est, pour moi, un véritable exemple de professeur accompli, alliant une grande profondeur académique à une expertise technique remarquable. Je tiens également à remercier chaleureusement le Professeur Christos Verikoukis pour son soutien précieux et pour avoir partagé avec nous ses visions stratégiques tout au long de ce parcours.

Je suis sincèrement reconnaissant envers mes amis, qui ont rendu ce chemin non seulement fructueux mais aussi agréable. Mes remerciements particuliers vont à Karim Boutiba, dont l'accueil chaleureux a rendu mes premiers jours à EURECOM beaucoup plus faciles. Je remercie également Mohammed, Sofiane, Giulio, Akram, Ayoub, Salim, Mazene, Franck et Abdelghani pour leur amitié et leur soutien.

Enfin, je souhaite remercier toute la famille EURECOM, ainsi que toutes les personnes qui, de près ou de loin, ont contribué à façonner ce travail.

Contents

1 General Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 2
- 1.3 Thesis Challenges and Contributions . . . 3
- 1.4 Thesis Structure . . . 9

2 State of The Art **10**
- 2.1 Introduction . . . 10
- 2.2 6G Networks . . . 10
  - 2.2.1 Overview . . . 10
  - 2.2.2 Key Enablers . . . 12
  - 2.2.3 Architecture . . . 12
  - 2.2.4 Challenges . . . 16
- 2.3 Intent-Based Networking . . . 16
  - 2.3.1 Overview . . . 16
  - 2.3.2 IBN Stages . . . 17
  - 2.3.3 Standards and Architectures . . . 18
  - 2.3.4 Challenges . . . 19
- 2.4 Artificial Intelligence . . . 20
  - 2.4.1 Overview . . . 20
  - 2.4.2 A Taxonomy of AI Methods . . . 20
  - 2.4.3 Explainability in AI . . . 24
- 2.5 Generative AI . . . 25
  - 2.5.1 Overview . . . 25
  - 2.5.2 A Taxonomy of GenAI Methods . . . 25
  - 2.5.3 Large Language Models . . . 26
- 2.6 Conclusion . . . 32

I Generative AI for Intent Translation **33**

3 Intent Translation to Service Configuration **34**
- 3.1 Introduction . . . 34
- 3.2 Related Works . . . 35
- 3.3 System Design . . . 36
  - 3.3.1 Proposed Architecture . . . 36
  - 3.3.2 Challenges and Open Issues . . . 38
  - 3.3.3 Our Intent Decomposition and Translation framework . . . 39

3.4 Performance Evaluation . . . 42
3.4.1 Experimentation Setup . . . 42
3.4.2 Experimentation Results . . . 42
3.5 Conclusion . . . 44

4 Intent Translation to Service Planning 45
4.1 Introduction . . . 45
4.2 Related Works . . . 46
4.3 System Design . . . 47
4.3.1 Problem Definition . . . 47
4.3.2 Solution Design . . . 48
4.4 EURECOM's OSS-GPT . . . 50
4.4.1 EURECOM's OSS . . . 50
4.4.2 Blueprint Generator - The NSD-expert . . . 51
4.5 Performance Evaluation . . . 53
4.5.1 Experimentation Setup . . . 53
4.5.2 Experimentation Results . . . 54
4.5.3 Discussions . . . 58
4.6 Conclusion . . . 58

II Generative AI for Intent Assurance 59

5 Anomaly Detection with NWDAF 60
5.1 Introduction . . . 60
5.2 NWDAF Realization . . . 61
5.2.1 NWDAF Architecture Aspects . . . 61
5.2.2 NWDAF . . . 62
5.2.3 NWDAF Interoperability Aspects . . . 63
5.3 NWDAF Use cases . . . 63
5.3.1 Core services . . . 63
5.3.2 ML-based services . . . 64
5.4 Performance Evaluation . . . 65
5.4.1 Experimentation Setup . . . 65
5.4.2 Experimentation Results . . . 65
5.4.3 Discussions . . . 66
5.5 Conclusion . . . 67

6 Anomaly Mitigation with XAI and LLMs 68
6.1 Introduction . . . 68
6.2 Related Works . . . 69
6.3 System Design . . . 70
6.3.1 High-level architecture . . . 70
6.3.2 Dynamic scaling use case . . . 71
6.4 Performance Evaluation . . . 72
6.4.1 Experimentation Setup . . . 72
6.4.2 Experimentation Results - Dynamic scaling framework . . . 73
6.4.3 Experimentation Results - LLM reasoning & decision-making . . . 75
6.4.4 Experimentation Results - LLM's trustworthiness . . . 76

6.4.5 Discussions . . . 76
6.5 Conclusion . . . 77

III Generative AI Operations in IBN **78**

7 Building Telecom-aware LLMs **79**
7.1 Introduction . . . 79
7.2 Related Works . . . 80
7.3 System Design . . . 82
7.3.1 Specifications gathering . . . 82
7.3.2 Cleaning and processing . . . 83
7.3.3 Data generation using LLMs . . . 85
7.3.4 Post-processing . . . 87
7.4 Performance Evaluation . . . 87
7.4.1 Experimentation Setup . . . 87
7.4.2 Experimentation Results . . . 88
7.4.3 Discussions . . . 95
7.5 Conclusion . . . 96

8 Optimizing Telecom-aware LLMs **97**
8.1 Introduction . . . 97
8.2 Related Works . . . 98
8.3 System Model . . . 98
8.4 System Design . . . 100
8.4.1 RL key parts . . . 100
8.4.2 DRL Approach . . . 101
8.5 Performance Evaluation . . . 102
8.5.1 Experimentation Setup . . . 102
8.5.2 Experimentation Results . . . 102
8.5.3 Discussions . . . 104
8.6 Conclusion . . . 105

IV Conclusion and Perspectives **106**

9 Conclusion **107**

10 Future Perspectives **109**
10.1 Introduction . . . 109
10.2 Future Directions in IBN . . . 109
10.2.1 LCM Enhancements . . . 109
10.2.2 Federation and Collaboration . . . 110
10.2.3 Security and Resilience . . . 110
10.3 Future Directions in GenAI . . . 111
10.3.1 Sustainability and Energy Efficiency . . . 111
10.3.2 Inference and Decision-Making Optimization . . . 111
10.3.3 Next-Generation Agentic AI Towards AGI . . . 112

11 Résumé en français **113**
11.1 Contexte de la Thèse . . . 113
11.2 Motivation . . . 114
11.3 Défis de la Thèse . . . 115
11.4 Contributions pour la traduction des intentions . . . 116
11.4.1 Traduction de l'intention en configuration de service . . . 116
11.4.2 Traduction de l'intention en planification de service . . . 117
11.5 Contributions pour l'assurance des intentions . . . 118
11.5.1 Détection des anomalies avec NWDAF . . . 118
11.5.2 Atténuation des anomalies avec XAI et LLMs . . . 119
11.6 Contributions pour les opérations GenAI dans l'IBN . . . 121
11.6.1 Construction des LLMs spécialisés télécom . . . 121
11.6.2 Optimisation des LLMs spécialisés télécom . . . 122
11.7 Conclusion . . . 123

List of Figures

1.1 PhD journey. 3

2.1 The evolution of connectivity [33]. 11
2.2 6G infrastructure and management layers. 13
2.3 3GPP CN architecture [38]. 15
2.4 Main IBN stages [42]. 17
2.5 A taxonomy of AI approaches. [50]. 21
2.6 Deep Reinforcement Learning. 23
2.7 The transformer - model architecture [61]. 27
2.8 Building LLMs pipeline [87]. 29
2.9 Fine tuning and In-Context Learning. 30

3.1 High-level architecture design to handle natural language-based intents life-cycle. 37
3.2 Natural language-based intent LCM in the EURECOM 5G facility [3]. 40
3.3 LLM-based intent decomposition and translation system. 41
3.4 Mean rating score throughout time. 43
3.5 Impact of the number of applications on decomposition and translation time. . . 43

4.1 OSS-GPT design. 49
4.2 EURECOM's OSS components. 51
4.3 LoRA [88]. 53
4.4 Experimentation setup. 54
4.5 Training loss. 55
4.6 Metrics. 55
4.7 Chatbot example when trust is activated. 56
4.8 Accuracy vs golden path. 57
4.9 OSS-GPT cost assessment. 57

5.1 Microservices-based NWDAF architecture. 61
5.2 LSTM Auto-encoder architecture. 64
5.3 Training and validation loss over the number of epochs. 66
5.4 A comparison between train volume and generated volume. 66
5.5 Anomaly probabilities for one week. 67
5.6 The anomaly detection for one week based on the Auto-encoder. 67

6.1 LLM-enabled trustworthy ZSM architecture design and use case. 71
6.2 Evaluation setup. 73
6.3 Dynamic CPU and RAM scaling in response to microservice load. 74
6.4 CPU and RAM allocation comparison. 74

6.5 LLMs scores and *E_output-D_output* generation times for the dynamic scaling use case. . . . 75
6.6 LLMs metrics score. . . . 76

7.1 5G Instruct Forge pipeline stages. . . . 82
7.2 Page 23 of the 3GPP TS 45.001 V18.0.0 (2024-03) 83
7.3 Figure textual description generation using MiniCPM. . . . 84
7.4 Illustration of the data cleaning and processing stage. . . . 85
7.5 Illustration of the data generation using LLMs stage. . . . 86
7.6 Example entries of OAI Instruct dataset, generated using 5G Instruct Forge. . . . 89
7.7 Impact of the percentage of trained parameters on MMLU [150] and 5G exam scores. . . . 91
7.8 Comparison of LLMs metrics. Bars colored in blue represent the Default versions of the models, while bars colored in orange represent the 5G-aware versions. . . . 93
7.9 Perplexity score. . . . 93
7.10 Mean generation time on the Q/A evaluation dataset. . . . 94
7.11 Exam results of different LLMs. . . . 94

8.1 DRL-enabled task scheduling for LLMs serving in 6G networks design. . . . 100
8.2 Convergence evaluation of DRL-DQN during training. . . . 103
8.3 Performance comparison among DRL-DQN, RR, and Random scheduling methods. 104

11.1 Parcours doctoral. . . . 115

List of Tables

2.1 A classification of 3GPP defined NWDAF Events [12]. 14

7.1 Open-source LLMs used for training. 88
7.2 Fine-tuning hyperparameters for each open-source LLM. 88
7.3 Improvement in Expert Satisfaction. 95

8.1 Setup parameters. 103

Acronyms

2G 2nd Generation.

3GPP 3rd Generation Partnership Project.

4G 4th Generation.

5G 5th Generation.

6G 6th Generation.

ADE Anomaly Detection Engine.

AE Analytics Engine.

AGI Artificial General Intelligence.

AI Artificial Intelligence.

AMF Access and mobility Management Function.

API Application Programming Interface.

AR Augmented Reality.

BSS Business Support System.

CN Core Network.

CoT Chain-of-Thought.

COTS Commercial Off-The-Shelf.

CS Communication Service.

DDOS Distributed Denial Of Service.

DIH Domain Intent Handler.

DL Downlink.

DNN Data Network Name.

DQN Deep Q-Network.

DRL Deep Reinforcement Learning.

E2E End-to-End.

EM Exact Match.

eMBB enhanced Mobile BroadBand.

ETSI European Telecommunications Standards Institute.

GAN Generative Adversarial Network.

GenAI Generative AI.

GPT Generative Pre-trained Transformer.

GUI Graphical User Interface.

HF Human Feedback.

HTTP Hypertext Transfer Protocol.

IBN Intent-Based Networking.

ICL In-Context Learning.

IETF Internet Engineering Task Force.

ILI Infrastructure-Level Intent.

IoT Internet of Things.

JCAS Joint Communication And Sensing.

JSON JavaScript Object Notation.

KB Knowledge Base.

KG Knowledge Graph.

KPI Key Performance Indicator.

KPM Key Performance Metrics.

LCM Life-Cycle Management.

LLM Large Language Model.

LMS Local Management System.

LoRA Low-Rank Adaptation.

LSTM Long Short-Term Memory.

MAE Mean Absolute Error.

MANO Management and Orchestration.

MDIH Multi-Domain Intent Handler.

MEC Multi-access Edge Computing.

MILP Mixed-Integer Linear Programming.

ML Machine Learning.

mMTC massive Machine-Type Communication.

MNO Mobile Network Operator.

MS Monitoring System.

NBI NorthBound Interface.

NER Named-Entity Recognition.

NF Network Function.

NFV Network Function Virtualization.

NFVO Network Function Virtualization Orchestrator.

NLP Natural Language Processing.

NMS Network Management System.

NSD Network Service Descriptor.

NWDAF Network Data Analytics Function.

O-RAN Open RAN.

OAI OpenAirInterface.

OS Operating System.

OSS Operations Support System.

PDU Packet Data Unit.

PEFT Parameter-Efficient Fine-Tuning.

QoE Quality of Experience.

QoS Quality of Service.

RAG Retrieval Augmented Generation.

RAN Radio Access Network.

RAND RAN Descriptor.

RCA Root Cause Analysis.

RIC RAN Intelligent Controller.

RIS Reconfigurable Intelligent Surface.

RL Reinforcement Learning.

RLHF Reinforcement Learning from Human Feedback.

RR Round-Robin.

SBI SouthBound Interface.

SBMA Service-Based Management Architecture.

SDN Software-Defined Network.

SDWAN Software-Defined WAN.

SHAP SHapley Additive exPlanations.

SLA Service Level Agreement.

SLM Small Language Model.

SLO Service Level Objective.

SM Service Model.

SMF Session Management Function.

SMO Service Management Orchestration.

TMF TM Forum.

TN Transport Network.

TS Technical Specification.

UE User Equipment.

UL Uplink.

UPF User Plane Function.

URLLC Ultra-Reliable and Low-Latency Communication.

VIM Virtualized Infrastructure Manager.

VR Virtual Reality.

XAI eXplainable AI.

XR eXtended Reality.

YAML Yet Another Markup Language.

ZSM Zero-touch network and Service Management.

Chapter 1

General Introduction

1.1 Context

Cellular networks have continuously evolved over the past decades, driven by the need to support ever-increasing traffic demands, new service types, and stricter performance requirements. At the foundation of these networks lies the physical and virtual infrastructure, composed of RANs, Transport Networks (TNs), and CNs that provide connectivity. On top of this infrastructure, network management frameworks are responsible for orchestrating, allocating, and optimizing resources to ensure service continuity and efficiency. With every new generation, from 2nd Generation (2G) to 5G, the management layer has become progressively more sophisticated, moving from manual configuration and static policies towards automation and software-driven control [4]. This evolution sets the stage for the next step: highly autonomous and intelligent management frameworks required to operate the complex and dynamic infrastructures of beyond 5G and 6G systems.

The evolution from 5G towards 6G networks is not only characterized by higher data rates and lower latency, but also by a paradigm shift towards more intelligent, flexible, and autonomous network infrastructures. While 5G introduced key service categories such as enhanced Mobile BroadBand (eMBB), Ultra-Reliable and Low-Latency Communication (URLLC), and massive Machine-Type Communication (mMTC), 6G is envisioned to extend these capabilities and push them to new extremes [5]. In particular, 6G aims to support immersive communication services such as holographic telepresence and eXtended Reality (XR), integrate Joint Communication And Sensing (JCAS) functionalities, enable large-scale digital twins of physical systems, and embed AI as a native feature of the network [6]. Furthermore, 6G is expected to provide ubiquitous connectivity across diverse environments, from dense urban areas to remote and underserved regions, ensuring that users and devices can seamlessly access high-quality services anytime and anywhere. This evolution underscores the increasing complexity of network operations and highlights the urgent need for advanced management frameworks capable of coping with the unprecedented scale, heterogeneity, and dynamism of 6G infrastructures [7].

Future 6G services, such as holographic communications, XR, autonomous systems, and industrial automation, demand strict performance guarantees, high resilience, and adaptive Quality of Service (QoS) provisioning. Traditional management frameworks, which are primarily reactive and rule-based, are not adequate to sustain the complexity and requirements of such services. Instead, the community is converging towards advanced, proactive, and AI-driven management frameworks that can reason about network states, predict anomalies, and adapt

resources autonomously [6]. In this context, IBN has emerged as a promising paradigm for network management. Rather than manually configuring low-level parameters, operators and end users (including vertical industry) can specify high-level intents, such as SLOs or performance goals [1]. The management system then interprets these intents, translates them into actionable configurations, and enforces them across the underlying infrastructure. Crucially, this process also involves continuous assurance mechanisms that verify whether the network is meeting the expressed intents in real time, providing feedback loops that can trigger corrective actions when deviations occur [8]. This approach not only reduces operational complexity but also guarantees alignment with business-level requirements while ensuring that End-to-End (E2E) performance and reliability commitments are fulfilled.

1.2 Motivation

At the present time, GenAI technologies, such as LLMs, have matured and are now considered stable, enabling their deployment in complex network management scenarios. Consequently, GenAI can play a central role in supporting autonomous, intent-driven management in beyond 5G and 6G networks. By leveraging GenAI, networks can better interpret high-level goals, adapt dynamically to changing conditions, and ensure that services meet strict performance and reliability requirements. The maturation of GenAI opens the door to practical, large-scale implementations of intent-driven frameworks that were previously only theoretical, making it timely and crucial to explore how these technologies can fundamentally improve IBN [9].

First, intent translation focuses on bridging the gap between high-level intents and the technical configurations required to implement them within the infrastructure [1]. Although standards define intents as human-readable constructs, typically expressed in formats like JSON or YAML, to simplify the communication of objectives, these intents remain complex and often require expert knowledge to craft correctly. Misinterpretations or misconfigurations can lead to service degradations or failures. This motivates our research to propose mechanisms that can simplify the expression of intents and use GenAI techniques to automatically translate them into precise, low-level network configurations. By doing so, we aim to significantly reduce the operational burden on human operators while maintaining accuracy, reliability, and alignment with the intended objectives.

Next, intent assurance focuses on verifying that network behavior consistently meets the expressed intents and on autonomously detecting and resolving anomalies [1]. Implementing a closed-loop system that can monitor, analyze, and correct deviations without human intervention is essential to ensure that high-level objectives are met. Achieving this requires frameworks capable of detecting anomalies, identifying their root causes, and triggering corrective actions, while maintaining transparency and trust for human operators. This motivates our research to explore GenAI-driven approaches that can ensure autonomous decision-making is reliable and interpretable, building confidence in these systems while enhancing operational efficiency.

Finally, these GenAI-driven approaches highlight that GenAI is a key enabler of both translation and assurance, but they also introduce new challenges related to GenAI model management. Deploying GenAI at scale in critical network operations requires careful attention to the lifecycle of GenAI models, including training, updating, monitoring, and integration [10]. Without effective model management, performance can degrade, decisions may become untrustworthy,

and scalability can be compromised. This motivates our research to consider frameworks and strategies that ensure GenAI models remain robust, trustworthy, and scalable, supporting the reliable operation of future 6G networks.

1.3 Thesis Challenges and Contributions

This thesis explores the advancement of IBN frameworks in 6G networks through the use of GenAI. As GenAI has recently gained popularity, it has simplified human-centric applications that involve natural language, including tasks such as language translation, understanding, sentiment analysis, and question answering. Intent translation and intent assurance fall within these tasks, as our goal is to express network intents using natural language and ensure their fulfillment using GenAI models while explaining decisions in a natural language. Fig. 1.1 summarizes my PhD journey.

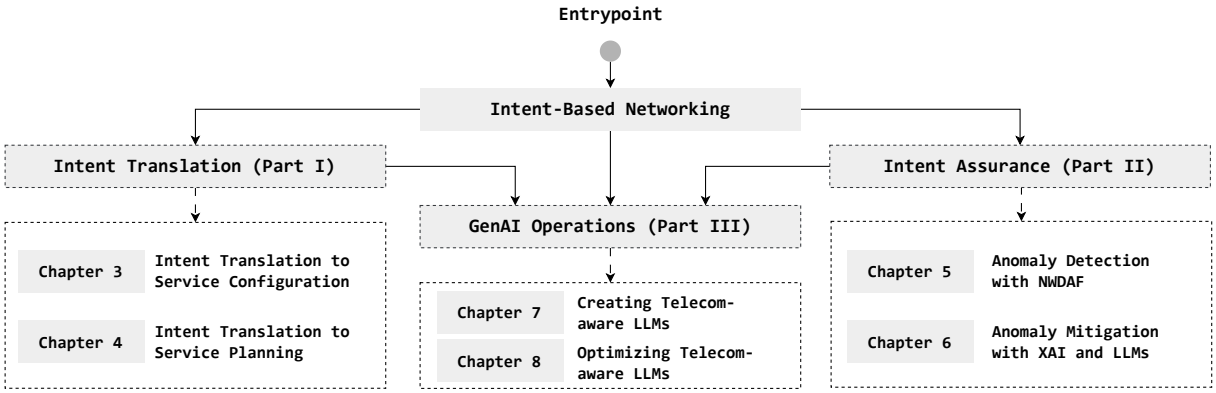


Figure 1.1: PhD journey.

The thesis first addresses the problem of intent translation, which becomes increasingly complex when intents are expressed in natural language. Translating natural language intents into structured, low-level configurations is particularly challenging due to the inherent ambiguity and flexibility of human language. To tackle this, we rely on LLMs, which have proven highly effective at understanding unstructured text across different languages, providing full flexibility in defining intents [11]. We initially focus on the task of service configuration across multiple technological domains, including RANs, CNs, and Edge/Cloud, which adds complexity because a single intent must be translated into configurations spanning multiple domains. We then extend the translation beyond service configuration to encompass other management tasks, such as retrieving intent status, configuring resources, and deleting resources across multiple domains. This requires decomposing and planning high-level intents into OSS-level actionable tasks.

Next, the thesis tackles the intent assurance problem, which involves autonomous detection and resolution of anomalies. We begin with anomaly detection in UE traffic, proposing an architecture that combines GenAI models, such as autoencoders, with a NWDAF to detect abnormal traffic patterns [12]. This detection serves as the input for other functions in the closed-loop system, enabling automated intent assurance. Subsequently, we address the full intent assurance loop, which involves detecting anomalies, identifying their root causes, and autonomously resolving them [13]. This is achieved through a combination of AI, XAI and GenAI models, with a primary focus on explaining decisions and corrective actions to users in natural language

using LLMs.

Finally, we observed that to enhance both translation and assurance quality and efficiency, domain-specific LLMs are required, particularly for Telecom. To this end, we propose a pipeline that extracts and structures 3GPP standards to create Telecom-aware LLMs, which outperform general-purpose models in understanding domain-specific intents. Given the substantial GPU requirements of LLMs, dedicating a separate model to each IBN task is not practical. To address this, we propose a routing mechanism that efficiently directs tasks to a set of LLMs using DRL [14], ensuring optimal resource utilization while maintaining high performance in IBN.

In this thesis, we tackled the following problems:

1. **Generative AI for Intent Translation**

   (a) Intent Translation to Service Configuration

      i. *Problem description:* Current IBN solutions rely on structured formats such as JSON or YAML (e.g., expressed as Network Service Descriptors (NSDs) for the Edge/Cloud domain in the European Telecommunications Standards Institute (ETSI) standards [15]), to configure services. While precise, creating these structures is complex, error-prone, and time-consuming, especially for non-expert users. Enabling intent specification directly in natural language can remove these barriers [11]. In addition, translating natural language intents becomes more challenging when spanning multiple technological domains, including RANs, CNs, and Edge/Cloud. Key challenges include (i) ambiguity and flexibility inherent to human language, (ii) decomposing high-level intents into actionable tasks across heterogeneous domains, and (iii) supporting the full intent Life-Cycle Management (LCM), including negotiation, translation, activation, monitoring, and assurance [16].

      ii. *Proposed solution:* To address these challenges, we propose an LLM-enabled intent translation and LCM system that allows users to express network intents directly in natural language. The system leverages the reasoning and language understanding capabilities of LLMs to interpret high-level intents and convert them into deployable structures, which can be activated across multiple technological domains, including RANs, CNs, and Edge/Cloud. The architecture supports the full intent LCM, including decomposing complex intents into domain-specific tasks, negotiating with users in natural language, translating intents into low-level configurations, activating services, and continuously assuring intent fulfillment. Additionally, a Human Feedback (HF) loop enables the system to iteratively learn from past translations. A prototype deployed at the EURECOM 5G facility [3] demonstrated high accuracy in generating deployable NSDs and reduced complexity for end users, validating the feasibility and effectiveness of LLM-enabled IBN in real-world multi-domain scenarios.

      iii. *Demonstration:* A live demonstration showcasing the intent translation prototype is available online[1].

[1] https://www.youtube.com/watch?v=SDyBge8WMt0

iv. *Publications:*

LLM-enabled Intent-Driven Service Configuration for Next-Generation Networks
Abdelkader Mekrache; Adlen Ksentini
2024 IEEE 10th International Conference on Network Softwarization (NetSoft)

Intent-Based Management of Next-Generation Networks: an LLM-Centric Approach
Abdelkader Mekrache; Adlen Ksentini; Christos Verikoukis
IEEE Network (Volume: 38, Issue: 5, September 2024)

(b) Intent Translation to Service Planning

i. *Problem description:* Managing 6G networks across multiple technological domains requires intelligent OSSs capable of E2E, cross-domain management. While previous work on LLM-centric intent LCM addressed service configuration, existing OSSs include many other management tasks which remain complex for users with limited technical knowledge [17]. For instance, an OSS may offer a single API call to configure a service, but several others to check infrastructure health, activate monitoring, and so on. Users often need to learn new API endpoints and data structures, and fulfilling high-level intents may require multiple coordinated API calls. Therefore, a more comprehensive solution is needed to simplify user interaction and support the full spectrum of OSS tasks, including configuration, monitoring, assurance, and adaptation to evolving APIs.

ii. *Proposed Solution:* We present OSS-GPT, an LLM-enabled multi-agent architecture [18] that extends our previous LLM-centric intent LCM approach to cover all OSS tasks. Using hierarchical planning, the system interprets natural language inputs, plans and executes multiple API calls across all 6G domains, monitors intent fulfillment, and ensures E2E service assurance. This chatbot-like interface simplifies user interaction while allowing the system to autonomously adapt to new APIs and evolving network conditions. Real-world experiments at EURECOM demonstrated the system's effectiveness in managing all OSS tasks across multiple domains using natural language.

iii. *Demonstration:* A live demonstration showcasing network management with OSS-GPT is available online[2].

iv. *Publications:*

OSS-GPT: An LLM-Powered Intent-Driven Operations Support System for 5G Networks
Abdelkader Mekrache; Adlen Ksentini; Christos Verikoukis
2025 IEEE 11th International Conference on Network Softwarization (NetSoft)

[2]https://www.youtube.com/watch?v=A1tTyHhyT80

Next-Generation 6G Network Management with OSS-GPT
Abdelkader Mekrache; Adlen Ksentini; Christos Verikoukis
Proceedings of the ACM SIGCOMM 2025 Posters and Demos, 158-160

2. **Generative AI for Intent Assurance**

   (a) Anomaly Detection with NWDAF

      i. *Problem description:* Anomaly detection represents the first step toward enabling ZSM in intent assurance. In this regard, the 3GPP [12] defines the NWDAF as a core component of 5G/6G networks, responsible for collecting, processing, and analyzing network data to enhance performance and user experience. Among the various NWDAF use cases is anomaly detection, where intent assurance systems rely on NWDAF-reported anomalies to resolve issues without human intervention. Although the 3GPP specifies these use cases, real-world implementations face challenges related to limited scalability and the need for advanced ML models capable of processing real-time network data. Moreover, the internal architecture of the NWDAF remains insufficiently defined, making standard-compliant implementations even more challenging.

      ii. *Proposed Solution:* We propose a microservices-based NWDAF architecture, allowing each 3GPP-defined application to be implemented as an independent, plug-and-play microservice [19]. Moreover, we tackled one use case which concern abnormal UE traffic detection, we design an Long Short-Term Memory (LSTM) autoencoder ML model trained on real network data from the Milano dataset [20]. This ML model is fully integrated into the NWDAF framework and deployed on an OAI 5G CN and RAN testbed [21]. Experimental validation demonstrates that the system can collect real network data through 3GPP-compliant interfaces and accurately detect abnormal traffic patterns generated by real UEs, highlighting both the flexibility and effectiveness of the proposed NWDAF architecture.

      iii. *Demonstration and Open-Source Artifacts:* A live demonstration showcasing UE traffic anomaly detection with NWDAF is available online[3]. In addition, the complete NWDAF source code has been made publicly available[4].

      iv. *Publication:*

      Combining Network Data Analytics Function and Machine Learning for Abnormal Traffic Detection in Beyond 5G
      Abdelkader Mekrache; Karim Boutiba; Adlen Ksentini
      GLOBECOM 2023 - 2023 IEEE Global Communications Conference

   (b) Anomaly Mitigation with XAI and LLMs

[3]https://www.youtube.com/watch?v=kI9GuJeW0es
[4]https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-nwdaf

i. *Problem description:* Intent assurance in IBN refers to ensuring that intent requirements are satisfied throughout their lifecycle, including autonomous anomaly detection and resolution. This concept, also referred to as ZSM, is a cornerstone of 6G networks [13]. To achieve ZSM, AI/ML models are commonly employed for anomaly detection and resolution. However, these models often function as black boxes, making it difficult for operators to interpret or trust their decisions. This lack of explainability raises concerns regarding reliability and accountability in AI-driven network management [22]. The core challenge, therefore, is to ensure that AI-based ZSM operates autonomously while remaining trustworthy and interpretable to human operators.

ii. *Proposed Solution:* To tackle the aforementioned challenges, we extend the work in [23] by including an LLM after the AI and XAI modules, resulting in a novel pipeline composed of AI|XAI|LLMs to enable trustworthy ZSM. Specifically, AI (XGBoost) detects anomalies [24], XAI (SHAP) identifies their root causes via feature importance [25], and LLMs (Llama2) provide natural language explanations and suggest/apply corrective actions [26]. This ensures both autonomous resolution of anomalies and improved interpretability for operators. A use case targeting Service Level Agreement (SLA) latency violations in cloud-native 6G microservices demonstrates how the system can scale resources while providing understandable explanations.

iii. *Demonstration:* A live demonstration showcasing anomaly detection and resolution with the proposed pipeline is available online[5].

iv. *Publication:*

On Combining XAI and LLMs for Trustworthy Zero-Touch Network and Service Management in 5G
Abdelkader Mekrache; Mohamed Mekki; Adlen Ksentini; Bouziane Brik; Christos Verikoukis
IEEE Communications Magazine (Volume: 63, Issue: 4, April 2025)

3. **Generative AI Operations in IBN**

(a) Building Telecom-aware LLMs

i. *Problem description:* LLMs have demonstrated remarkable capabilities across general domains; however, their performance in specialized fields such as telecommunications remains limited. This limitation poses challenges for telecom-specific tasks such as intent translation and assurance in IBN. The primary reason is that LLMs are typically trained on general-purpose corpora that contain little domain-specific knowledge. Fine-tuning LLMs for 5G/6G expertise requires high-quality, structured datasets derived from complex and unstructured sources such as 3GPP TSs [27]. These documents are dense, highly technical, and often difficult to parse, making it challenging to create datasets that enable LLMs to

[5]https://www.youtube.com/watch?v=1CoDNVWJcqA

understand telecom protocols, network configurations, and SLAs. Without such datasets, LLMs cannot reliably perform telecom-specific tasks or provide accurate and actionable insights for network management.

ii. *Proposed Solution:* We present 5G INSTRUCT Forge, an advanced data engineering pipeline that processes 3GPP TSs into structured formats suitable for training LLMs. The pipeline enables systematic creation of domain-specific datasets for Telecom. As a proof of concept, we generated the OAI-Instruct dataset from relevant 3GPP TSs and used it to fine-tune open-source LLMs. The resulting models, specialized in Telecom, demonstrated superior performance compared to OpenAI's GPT-4 [28] on Telecom-related tasks.

iii. *Open-Source Artifacts:* The complete 5G INSTRUCT Forge source code has been made publicly available[6]. In addition, the OAI-Instruct dataset is available online[7].

iv. *Publication:*

5G INSTRUCT Forge: An Advanced Data Engineering Pipeline for Making LLMs Learn 5G
Azzedine Idir Ait Said; Abdelkader Mekrache; Karim Boutiba; Kostas Ramantas; Adlen Ksentini; Moufida Rahmani
IEEE Transactions on Cognitive Communications and Networking (Volume: 11, Issue: 2, April 2025)

(b) Optimizing Telecom-aware LLMs

i. *Problem description:* LLMs are increasingly integrated into IBN networks due to their advanced capabilities in reasoning, coding, and language understanding, which enable autonomous network decision-making. However, their high computational cost means that a single LLM must often be shared across multiple network applications. This introduces several complex challenges for task scheduling: (i) the arrival times of tasks are unpredictable, making it difficult to plan resource allocation in advance, (ii) each task is subject to strict SLO deadlines, which must be met to ensure network reliability and performance, and (iii) different LLMs exhibit varying levels of performance depending on the type of task, resulting in heterogeneous task scores [2]. These factors make traditional scheduling methods insufficient, as they cannot dynamically optimize task assignments to maximize overall performance while respecting deadlines.

ii. *Proposed Solution:* We propose a DRL-enabled framework [14] for task scheduling that dynamically routes tasks to the most appropriate LLM, aiming to maximize task scores while meeting SLO deadlines. These tasks encompass various 6G operations, including coding, intent translation, intent assurance, and more. The framework continuously learns optimal scheduling policies in real time un-

[6]https://gitlab.eurecom.fr/netsoft/5g-instruct-forge
[7]https://huggingface.co/datasets/Netsoft/oai-instruct

der dynamic traffic and heterogeneous task conditions. Experimental evaluations demonstrate that the DRL-based scheduler outperforms conventional approaches, such as Round-Robin (RR) and random scheduling, in terms of task success rate, deadline adherence, and overall network efficiency.

iii. *Publication:*

DRL-enabled SLO-aware Task Scheduling for Large Language Models in 5G Networks
Abdelkader Mekrache; Adlen Ksentini; Christos Verikoukis
IEEE ICC - 2025 IEEE International Conference on Communications

1.4 Thesis Structure

This thesis is structured as follows: Chapter 2 is dedicated to the presentation of the background and concepts on which our research is based, which can be mainly grouped into four topics: 6G networks, IBN, AI, and GenAI techniques. Then, we divided the thesis into four Parts: In Part I, we introduce the contributions related to GenAI for intent translation. Part I consists of two chapters: In Chapter 3, we present our LLM-based intent translation framework for service configuration, while in Chapter 4, we extend this framework to support multiple management tasks, including service planning and configuration. In Part II, we present the contributions related to GenAI for intent assurance, specifically detecting and resolving anomalies autonomously. Part II consists of two chapters: In Chapter 5, we address anomaly detection for UE traffic using NWDAF, while in Chapter 6, we present a trustworthy framework implementing a full assurance closed-loop with AI, XAI, and LLMs. In Part III, we present GenAI operations for IBN. Part III consists of two chapters: In Chapter 7, we introduce contributions related to creating Telecom-aware LLMs, while in Chapter 8, we present task routing based on DRL to multiple LLMs for networking tasks. Finally, Part IV contains two chapters: Chapter 9 provides a general conclusion, while Chapter 10 discusses the perspectives and future directions of the thesis.

Chapter 2

State of The Art

2.1 Introduction

The evolution of mobile communication networks towards the 6G is expected to support unprecedented levels of connectivity, intelligence, and automation. Unlike previous generations, 6G aims to seamlessly integrate communication, computing, and sensing resources to enable immersive experiences, ultra-reliable services, and AI-native network operations [29]. The growing complexity and heterogeneity of 6G infrastructures pose significant challenges for network management, particularly in translating high-level user or operator requirements into low-level configurations. IBN has emerged as a promising approach to address this challenge. IBN allows stakeholders to express high-level intents, which are automatically translated, deployed, and continuously assured in the network. However, implementing IBN in 6G networks requires advanced reasoning, adaptability, and the ability to understand and act upon complex, context-rich instructions [30]. GenAI, particularly LLMs and autonomous agents, offers unique capabilities to enhance IBN. These models can interpret natural language intents, generate configuration commands, reason over network constraints, and provide human-friendly explanations, enabling a new level of autonomous and intelligent network management [31].

In this chapter, we first provide a clear conceptual foundation of 6G networks, followed by a discussion of IBN, which is the main focus of this thesis in 6G network management. We then introduce key AI concepts and conclude with GenAI, which is the main focus of this thesis in the context of AI.

2.2 6G Networks

In this section, we first provide a brief overview of 6G networks and their key enablers. Next, we present the architectural layers, organized into infrastructure and management. Finally, we discuss the challenges within the management layer, which constitutes the primary focus of this thesis.

2.2.1 Overview

6G network is a wireless communication system envisioned to succeed 5G around 2030, designed to provide ultra-reliable, intelligent, and immersive connectivity by tightly integrating communication, computing, and sensing resources. 6G extends beyond eMBB, URLLC, and mMTC, to incorporate native support for AI, semantic communication, and holographic services. This

evolution is driven by the need to support a wide range of advanced and data-intensive use cases that push the limits of current network technologies [32]. Fig. 2.1 illustrates the evolution of connectivity in cellular networks.

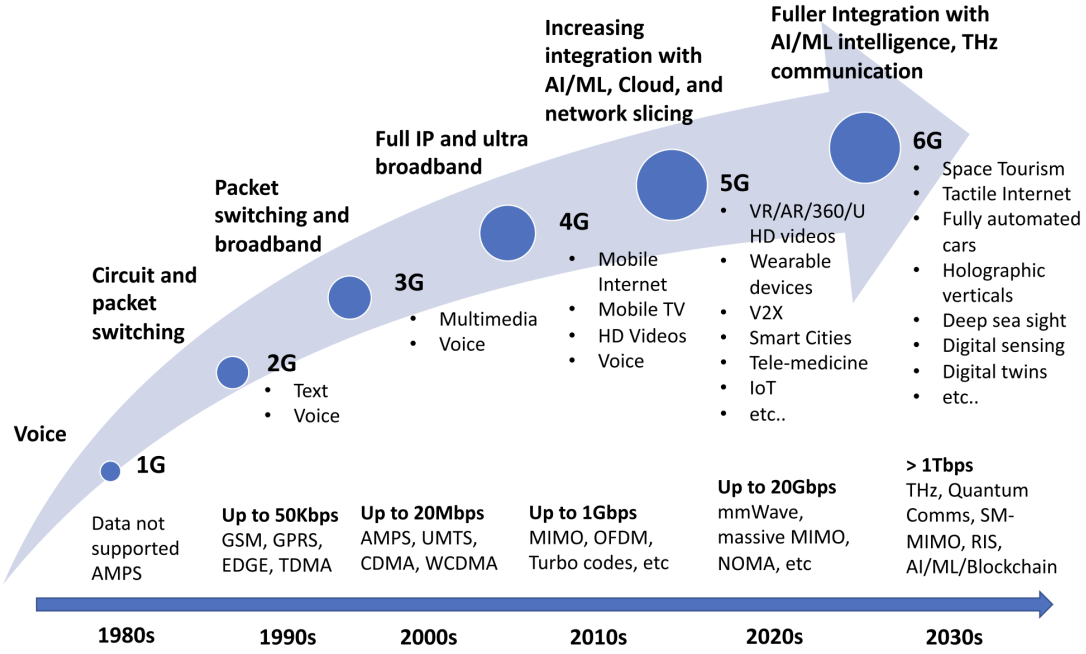


Figure 2.1: The evolution of connectivity [33].

For example: (i) Augmented Reality (AR) and Virtual Reality (VR) applications will require per-user data rates in the gigabit-per-second range and sub-millisecond latencies to enable real-time, immersive experiences, going far beyond the capacities of 5G networks. Holographic telepresence will demand even higher performance, with multi-terabit-per-second data rates and ultra-low latency to allow real-time interaction with fully digitized sensory inputs, including vision, touch, and potentially other human senses [34]; (ii) In the healthcare sector (eHealth), 6G will enable remote surgery, real-time monitoring, and workflow optimization by guaranteeing ultra-reliable connectivity ($> 99.99999\%$) and sub-millisecond latencies, overcoming current limitations in tactile feedback and remote intervention. Pervasive connectivity will also be a key feature, supporting an extremely dense network of devices, up to 10^7 devices per km^2 in urban areas, while ensuring energy-efficient, low-cost, and environmentally friendly deployments [34]; (iii) Industry 4.0 and robotics will leverage 6G to fully realize digital manufacturing, integrating cyber-physical systems and Internet of Things (IoT) services for real-time diagnostics, maintenance, and operation. This will require highly reliable, low-latency, and high-throughput communication for AR/VR training, inspection, and direct machine-to-machine coordination [34]; Finally, (iv) unmanned mobility, including autonomous vehicles and aerial drones, will rely on 6G to achieve unprecedented levels of reliability ($> 99.99999\%$) and low latency ($< 1ms$), even in ultra-high-mobility scenarios of up to $1000km/h$. This will ensure passenger safety, efficient traffic management, and high-capacity data exchange for infotainment, sensors, and fleet coordination [34].

To meet the stringent performance requirements of these diverse use cases, multiple 6G enabling technologies are being considered, including advanced spectrum utilization, ultra-dense

networks, integrated AI-driven management, and novel air-interface designs. Conventional network management approaches will be insufficient; therefore, the integration of AI and intelligent orchestration is fundamental to the realization of 6G networks, enabling them to dynamically adapt, optimize, and deliver immersive and reliable services across all envisioned scenarios [6].

2.2.2 Key Enablers

The realization of 6G networks is supported by several transformative technologies, each playing a crucial role in enabling ultra-reliable, intelligent, and immersive connectivity [32]:

- JCAS: Systems that allow devices and base stations to sense the environment while simultaneously transmitting data. JCAS enables applications such as environmental monitoring, autonomous navigation, intelligent transportation systems, and smart city services.
- Reconfigurable Intelligent Surfaces (RISs): Programmable metasurfaces that dynamically control the propagation of radio waves. RIS improve signal quality, extend coverage, and enhance energy efficiency by intelligently reflecting and refracting signals. Their integration with JCAS supports real-time optimization of the wireless environment for applications such as localization and gesture recognition.
- Edge and cloud-native computing: Bringing computation closer to the data source reduces latency and bandwidth usage. These architectures support real-time analytics, decision-making, and low-latency applications, including autonomous vehicles, industrial automation, and AR/VR services. Virtualization and containerization enable dynamic allocation of network and computing resources, supporting multi-tenant environments and flexible service deployment.
- Network softwarization and virtualization: Software-Defined Network (SDN) and Network Function Virtualization (NFV) decouple network services from the underlying hardware, enabling flexible, programmable, and automated network management. Virtualized network slices allow multiple services to coexist on shared infrastructure with guaranteed performance, supporting dynamic resource orchestration.
- Intent-driven orchestration and AI-native management: Combining AI-native management with IBN enables automated decision-making, predictive maintenance, and adaptive resource allocation. The network can self-optimize and enforce high-level service intents across heterogeneous virtualized and physical infrastructures, ensuring reliability, efficiency, and SLA compliance.
- Quantum communication and security: Leveraging quantum mechanics for secure key distribution and unbreakable encryption. These technologies protect sensitive data and ensure confidentiality and integrity, making 6G networks resilient to emerging security threats.

2.2.3 Architecture

6G networks are conceptualized as a multi-layer architecture in which the infrastructure and management layers serve distinct yet interconnected roles. The infrastructure layer provides the essential physical and virtual resources for connectivity, computing, and data transport, while the management layer orchestrates and controls these resources to ensure efficient, reliable, and autonomous network operation. Fig. 2.2 illustrates these layers.

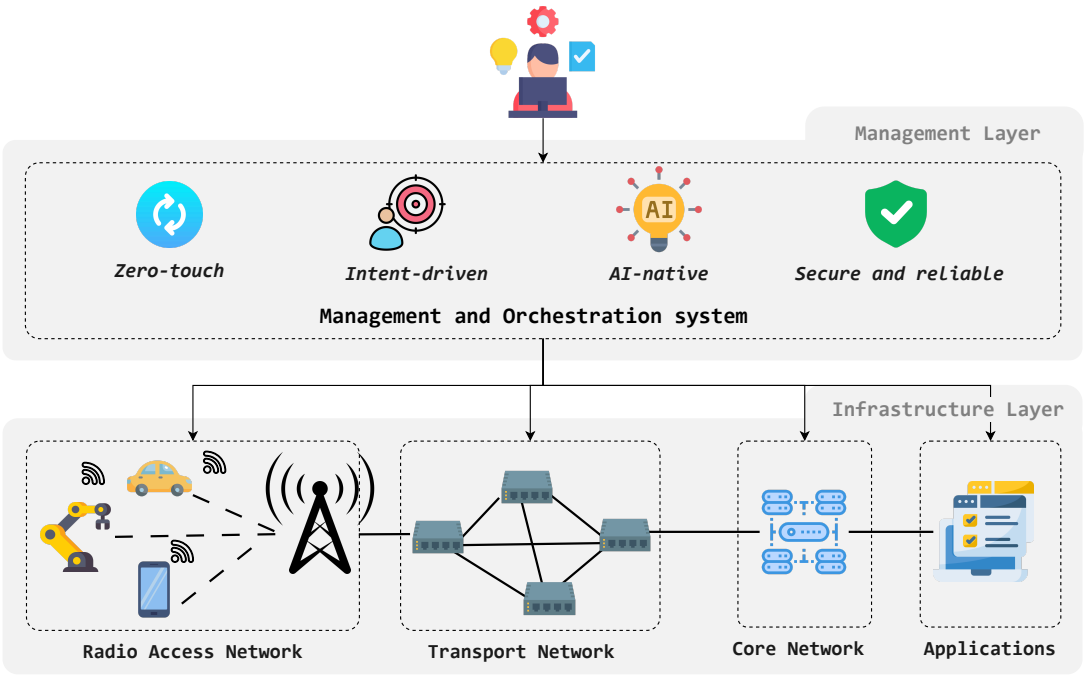


Figure 2.2: 6G infrastructure and management layers.

Infrastructure layer

The infrastructure layer encompasses the physical and virtual resources that form the foundation of next-generation networks. It can be summarized in four main components.

The RAN comprises all the wireless links and devices that enable wireless connectivity in the 5G network. It can include UEs, such as smartphones, vehicles, robots, and IoT devices, together with base stations and advanced radio technologies like terahertz communications and RISs. These technologies collectively provide ultra-high data rates, enhanced spatial multiplexing, and improved coverage, supporting immersive applications and high-mobility scenarios [34]. Multiple standardization bodies are working toward the openness of RAN interfaces, such as O-RAN [35], which is expected to be a key feature in collaborative next-generation networks.

The TN interconnects the RAN and CN through high-capacity, low-latency links that ensure seamless end-to-end connectivity. It relies primarily on optical fiber and microwave backhaul, supported by high-performance routers, switches, and aggregation devices to efficiently manage and forward traffic across the network. These networking components provide resilient and scalable data transport while enabling features such as redundancy, traffic engineering, and QoS enforcement. To meet the dynamic requirements of next-generation networks, SDN and NFV allow flexible and programmable control of transport resources, enabling dynamic bandwidth allocation, traffic prioritization, and network slicing. Furthermore, Software-Defined WAN (SD-WAN) extends these capabilities across wide-area networks, intelligently routing traffic based on real-time network conditions to optimize performance and reliability.

Before reaching the applications deployed within the MEC or Cloud, traffic passes through the CN, which is first responsible for connecting the UEs and providing them access to the internet via the control plane, and then for supporting data transport between the UEs and external networks via the data plane. (i) The control plane comprises functions such as the Access and

mobility Management Function (AMF), responsible for UE authentication and mobility management, and the Session Management Function (SMF), which manages session control, among others [36]. In addition, as recently standardized by 3GPP [12], the NWDAF is an ML-powered Network Function (NF) designed to collect data from other CN NFs and from the orchestration and management system, providing analytics information and predictive insights to NWDAF consumers. The 3GPP has defined multiple use cases for NWDAF, including UE abnormal behavior detection, load prediction, and network performance optimization. Table 2.1 summarizes the NWDAF use cases according to 3GPP Release 17 standards [12] and provides a brief description of each service. (ii) In the data plane, the User Plane Function (UPF) handles data forwarding, packet inspection, and routing between the RAN and external data networks. Many of these components build upon existing 5G core architectures, ensuring backward compatibility while extending capabilities to meet the requirements of emerging 6G services [37]. As the official 6G standardization of the core network architecture is not yet finalized, the actual components remain under definition. However, it is expected that the future 6G core will build upon the 5G-Advanced CN, whose architecture, defined by 3GPP, is illustrated in Fig. 2.3.

Table 2.1: A classification of 3GPP defined NWDAF Events [12].

Categorie	Event ID	Description
Network Conditions	slice_load_level	Network slice load level computation and prediction.
	nsi_load_level	Network slice instance load level computation and prediction.
	nf_load	Load analytics information and prediction for a specific network function.
	user_data_congestion	Congestion information—current and predicted for a specific location.
	network_performance	Network performance computation and prediction on the gNB status information, gNB resource usage, communication performance and mobility performance in an Area of Interest.
	qos_sustainability	QoS sustainability—reporting and predicting QoS change.
Device Behaviour	ue_mobility	UE mobility analytics and expected behaviour prediction.
	ue_comm	UE communication analytics and pattern prediction.
	abnormal_behaviour	UE abnormal behaviour detection and anomaly detection, e.g. being misused or hijacked.
Service Experience	service_experience	Service experience computation and prediction for an application or UE group.
	red_trans_exp	Redundant Transmission Experience related analytics. These analytics may be used by the SMF to determine whether redundant transmission shall be performed, or shall be stopped.
	wlan_performance	Quality and performance of WLAN connection of UE computation and prediction.
	dn_performance	User plane performance computation and prediction.
	sm_congestion	Session Management Congestion Control Experience information for specific DNN and/or S-NSSAI.
Network Planning	dispersion	Location or network slices where UEs disperse most of their data volume and sessions transactions.

Vertical applications, accessed by UE through the CN and the Internet, represent the final layer of the network architecture, where digital services are delivered. These applications can be hosted in Multi-access Edge Computing (MEC) environments or in centralized cloud data centers, depending on their latency, bandwidth, and computational requirements. MEC brings computation and storage closer to the network edge, reducing latency and enabling real-time AI-driven services, immersive experiences, and context-aware network functions [39]. By offloading computation from centralized clouds to edge nodes, MEC enables instantaneous data analyt-

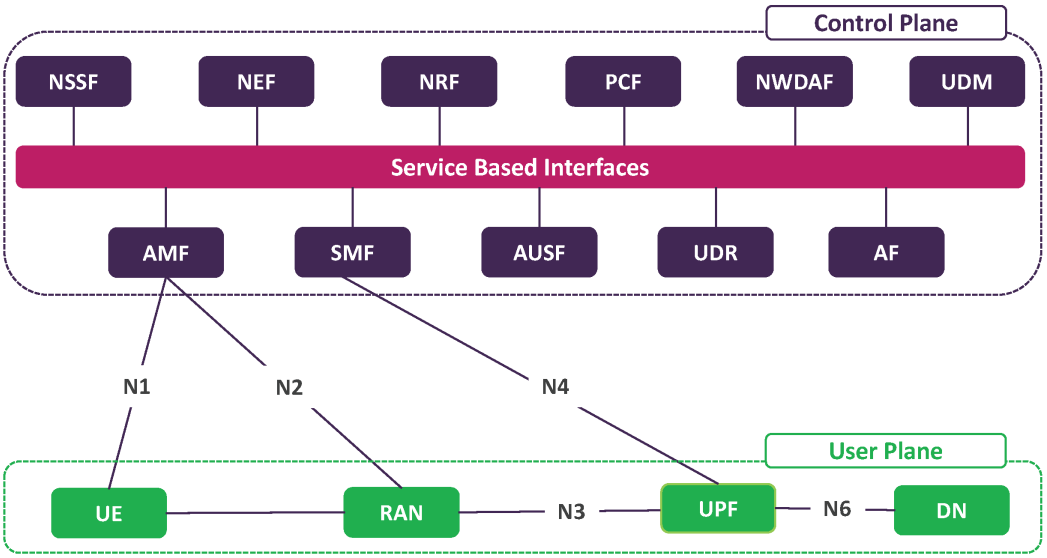


Figure 2.3: 3GPP CN architecture [38].

ics, decision-making, and actuation, essential for time-sensitive use cases such as holographic telepresence, autonomous driving, and industrial automation. In the context of 6G, MEC and distributed cloud infrastructures together will provide a unified computing continuum, ensuring that vertical applications dynamically adapt to user mobility, network conditions, and service demands.

Management Layer

The management layer is in charge of coordinating and controlling all network and computing resources in a 6G system. Its goal is to keep the network running efficiently, reliably, and with minimal human intervention. It includes several key components: (i) the ZSM framework, which enables self-configuration, self-healing, and self-optimization so that the network can automatically adapt to changes or faults while maintaining service quality [8]; (ii) IBN, which allows users to express their goals in simple, high-level terms, such as improving latency or increasing bandwidth, and automatically translates them into technical configurations and policies [1]; (iii) AI-native closed-loop automation, which uses ML to predict issues, allocate resources dynamically, and detect anomalies, enabling the network to react proactively and autonomously to traffic changes or unexpected events [13]; and (iv) security and reliability management, which applies AI-based monitoring and protection mechanisms to detect threats, mitigate attacks, and ensure continuous service availability. Altogether, these capabilities make 6G networks more intelligent, adaptive, and self-sustaining.

These concepts are highly interconnected [40]: AI-native automation detects potential issues or performance degradations, ZSM resolves them autonomously, and the entire process is framed under IBN to ensure that the network consistently meets the defined intents [6]. In this way, the management layer provides a transparent, intent-driven approach where anomalies and operational adjustments are handled automatically. As IBN is the main focus of this thesis, its details are presented in Section 2.3.

2.2.4 Challenges

While 6G networks promise transformative advancements in connectivity and service delivery, they also introduce significant challenges that must be addressed to realize their full potential. These challenges are particularly pertinent in the context of IBN, which aims to simplify network management by allowing operators to specify high-level objectives or "intents" that the network autonomously translates into actionable configurations and policies.

- Complexity: The integration of diverse technologies, such as terahertz communications, RISs, and AI-driven automation, into 6G networks increases the complexity of network management. IBN seeks to mitigate this by abstracting low-level configurations and enabling dynamic adaptation to meet specified intents. However, the complexity of translating high-level intents into precise configurations remains a significant challenge [1].
- Trustworthiness: Ensuring the fairness, transparency, and robustness of AI-driven management is critical. In IBN, the network's ability to autonomously interpret and implement intents must be trustworthy to prevent unintended behaviors. Establishing mechanisms for validating and verifying AI decisions is essential to maintain operator confidence and system reliability [41].
- Scalability: The exponential growth in connected devices and data traffic necessitates scalable management solutions. IBN offers scalability by enabling the network to autonomously adjust to changing demands. However, ensuring that IBN frameworks can scale effectively across diverse and dynamic 6G environments poses significant challenges [41].
- Security and Privacy: Defending against adversarial AI and securing critical infrastructures are paramount. In IBN, the network's autonomous decision-making processes must be protected from malicious interference. Implementing robust security measures and privacy-preserving techniques within IBN frameworks is an ongoing area of research [40].

IBN provides a unifying framework to address complexity, scalability, and trustworthiness by allowing the network to interpret high-level intents and autonomously adapt to dynamic conditions. In the next section, we explore this concept in detail.

2.3 Intent-Based Networking

In this section, we first provide an overview of the IBN concept. Next, we present a classification of its stages, followed by the standards and reference architectures supporting IBN, and we conclude with a discussion of its challenges.

2.3.1 Overview

IBN is a paradigm for autonomous network management in which high-level objectives, or intents, are expressed by network operators or end-users in natural or structured language and automatically translated into low-level network configurations and policies. The key principle of IBN is to allow stakeholders to specify "what" the network should achieve, rather than "how" it should be configured. This abstraction reduces operational complexity and aligns network behavior with service and business objectives [42]. IBN has emerged in response to the increasing complexity, heterogeneity, and dynamism of modern networks, including 5G and forthcoming

6G deployments. Traditional manual configuration and static policies are no longer sufficient to meet stringent performance, reliability, and scalability requirements. By capturing, translating, deploying, and continuously assuring intents, IBN enables automated, adaptive, and self-optimizing network operations. The development of IBN is supported by both academic research and industrial initiatives. Standardization bodies and consortia, such as Internet Engineering Task Force (IETF) [43], TM Forum (TMF) [44], and ETSI ZSM [8], have defined reference architectures and frameworks for intent expression, translation, and assurance, providing the foundation for real-world deployment. As a result, IBN represents a critical evolution in network management, offering a coherent framework to deliver agile, reliable, and intelligent services in increasingly complex network environments.

2.3.2 IBN Stages

The lifecycle of an IBN system can be structured into five main stages [42], as illustrated in Fig. 2.4, which collectively form a closed-loop automation framework that ensures intents are accurately captured, deployed, and continuously satisfied. These stages are listed below:

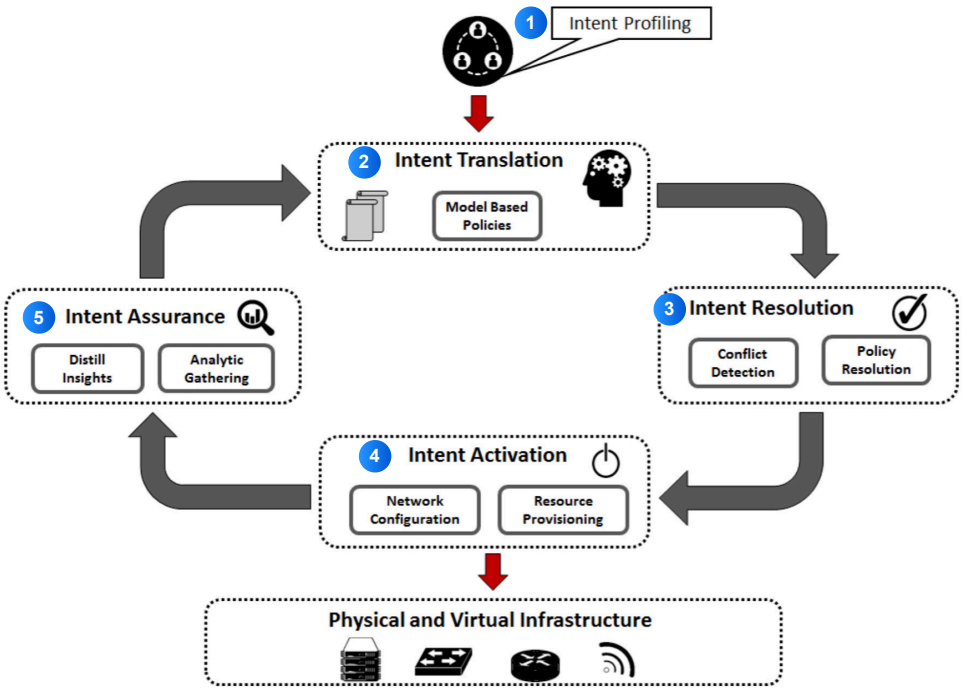


Figure 2.4: Main IBN stages [42].

1. Intent Profiling: The first stage involves capturing high-level user or operator requirements. Unlike traditional command-line or API-based configurations, intents are expressed in a human-friendly format, such as natural language, structured templates, or graphical interfaces. The system may also interact with the user to refine ambiguous or incomplete intents, ensuring that the captured intent accurately reflects the desired network outcome. This stage is essential for abstracting network complexity and aligning the system with user expectations.

2. Intent Translation: Once an intent is captured, it must be transformed into actionable network policies. This stage translates the abstract, high-level intent into formal, machine-

readable configurations that can be automatically deployed across the network infrastructure. Techniques such as model-driven approaches, policy compilers, and AI-based translation methods are often employed to ensure that the intended network behavior is preserved while accommodating the underlying network topology and capabilities.

3. Intent Resolution: Multiple intents may coexist in the network, potentially leading to conflicts or contradictions. The intent resolution stage identifies and disambiguates such conflicts, applying predefined rules, priority mechanisms, or optimization algorithms to reconcile competing intents. In cases where conflicts cannot be automatically resolved, the system may alert network operators for guidance. Effective resolution ensures that each intent is enforced without compromising other ongoing network operations.

4. Intent Activation: After ensuring that no conflicts exist, the translated network policies are deployed across the infrastructure. This deployment is customized to meet the specific requirements of each intent, whether it pertains to bandwidth allocation, security enforcement, or service prioritization. Automation tools, orchestration platforms, and SDN controllers are typically used to provision and configure network elements dynamically, allowing the network to adapt to user-defined objectives.

5. Intent Assurance: Networks are inherently dynamic, with varying traffic patterns, resource availability, and environmental conditions. Intent assurance continuously monitors network performance to verify that deployed configurations satisfy the original intent. If deviations are detected, proactive or reactive corrective actions, such as self-configuration, load redistribution, or adaptive policy updates, are triggered. This stage ensures the longevity and reliability of the intent, enabling the network to remain aligned with high-level objectives over time.

The interaction of these stages creates a closed-loop automation cycle that transforms IBN from a conceptual framework into an operational methodology for autonomous, intelligent, and resilient network management.

2.3.3 Standards and Architectures

IBN is supported by several key standards and reference architectures developed by leading industry bodies, providing a structured framework for its implementation and integration into modern network infrastructures. These standards define the principles, models, and interfaces necessary to realize intent-driven, automated network management, while reference architectures describe how components interact to support the full intent lifecycle.

The IETF has contributed foundational documents to standardize IBN concepts. RFC 9315 [43] clarifies the notion of an "intent" and provides an overview of the associated functionality, establishing a common understanding for both research and engineering communities. RFC 9417 [45] describes an architecture for intent assurance, highlighting the importance of monitoring and validating that service instances operate as expected. RFC 9418 [46] specifies YANG modules for representing assurance graphs, which decompose services into atomic assurance elements to facilitate systematic monitoring and validation. Together, these RFCs define both the conceptual and architectural basis for intent-driven monitoring and assurance.

The TMF has developed a comprehensive framework for IBN, emphasizing intent LCM and automation. The Intent Toolkit [44] helps members understand the role of intent in autonomous

networks and facilitates communication of requirements, goals, and constraints. TMF-TR290 [47] provides a standardized vocabulary and semantic framework for expressing, reporting, and managing intents. It defines core elements such as intent, expectation, condition, and target, ensuring consistent representation and interpretation of high-level network objectives across heterogeneous systems. TMF-921A [48] specifies a RESTful API for implementing the full LCM, including expression, translation, deployment, and assurance. These components collectively form a reference architecture for IBN, illustrating how the toolkit, TMF-TR290, and TMF-921A interact to enable automation and interoperability.

ETSI's ZSM group focuses on E2E automation and the role of intent in autonomous networks. ZSM-011 [8] emphasizes user-friendly input expressions and increased flexibility in automation. ZSM-016 [13] specifies intent-driven closed loops within the ZSM framework, showing how intents can guide autonomous network behaviors and support self-configuration and self-healing. These documents define a modular reference architecture for closed-loop automation, illustrating how intents interact with the ZSM functional components to achieve autonomous network operations.

2.3.4 Challenges

Despite its transformative potential, several challenges remain in the realization of IBN:

- Ambiguity and Complexity of Intent Translation [30]: Converting high-level, human-friendly intents into precise, machine-executable configurations is inherently complex. The translation process must accurately capture the desired behavior while accounting for the constraints and capabilities of heterogeneous and dynamic network infrastructures.
- Assurance and Trustworthiness [30]: Ensuring that deployed configurations fulfill the original intent is critical. Networks are dynamic, and maintaining trustworthiness requires continuous monitoring, validation, and corrective actions when discrepancies arise between intended and actual network behavior. Robust assurance mechanisms often rely on sophisticated analytics and closed-loop control.
- Verification [30]: Even after successful translation, it is essential to validate that the resulting configurations faithfully reflect the expressed intents. Verification involves testing, simulation, and sometimes formal methods to detect misconfigurations, conflicting intents, or unintended behaviors, ensuring the network aligns with the operator's goals.
- Scalability and Integration Complexity [30]: Large-scale networks must handle multiple concurrent intents efficiently, resolving conflicts and enforcing consistency across diverse infrastructures. Integrating IBN frameworks with existing management systems and legacy networks introduces additional complexity, requiring seamless interoperability without disrupting ongoing operations.

Addressing these challenges is crucial to fully realize IBN's potential. Recent advances in AI, particularly GenAI, provide promising approaches to support intent translation, verification, and assurance, enabling more reliable, scalable, and autonomous network management. In the next section, we lay the foundation and present a tutorial on AI methods that can be applied to network management.

2.4 Artificial Intelligence

Multiple AI approaches were adopted in this thesis, with a particular focus on LLMs in Chapters 3, 4, 6, 7, and 8, on autoencoders in Chapter 5, and on other methods such as DRL in Chapter 8. In addition, the XAI paradigm was employed in Chapter 6 to ensure the interpretability and transparency of the AI models used. Therefore, in this section, we first provide an overview of the key AI concepts, followed by a taxonomy of AI approaches that illustrates the relationships between the techniques, and finally discuss how XAI addresses the trust problem of black-box AI.

2.4.1 Overview

AI is a field of computer science focused on creating systems capable of performing tasks that typically require human intelligence, such as reasoning, learning, decision-making, and perception. Broadly, AI encompasses several subfields, including ML, where systems learn patterns from data; deep learning, which leverages neural networks with multiple layers for complex representations; and RL, where agents learn by interacting with an environment to maximize rewards. Over the decades, AI has evolved significantly, with notable milestones demonstrating its capabilities. Early symbolic AI systems achieved success in reasoning tasks, while programs such as HiTech defeated Arnold Denker in 1988, and IBM's Deep Blue defeated Garry Kasparov in 1997, illustrating AI's potential in strategic decision-making [49]. Since the 2000s, the explosion of data from sensors, chips, and networked devices has accelerated the development of data-driven AI techniques.

Today, AI is embedded in daily life and technologies, ranging from autonomous systems and robotics to Natural Language Processing (NLP) and intelligent network management. These advancements have positioned AI as a key pillar in next-generation networks, enabling intelligent decision-making, automation, optimization, and predictive capabilities at unprecedented scales. Understanding these foundational concepts is essential before exploring the relationships between different AI techniques, which will be presented in the subsequent taxonomy.

2.4.2 A Taxonomy of AI Methods

As shown in Fig 2.5, AI encompasses a range of approaches and methods, which can be broadly organized into the following hierarchy: (i) ML; (ii) Brain-inspired neural networks; (iii) Deep learning; (iv) DRL; and GenAI [50].

Machine Learning

ML is a core paradigm of modern AI that focuses on designing algorithms capable of learning patterns from data to make predictions or decisions without explicit programming. ML methods can be broadly categorized into supervised learning, unsupervised learning, and RL, each addressing different types of tasks and data [51].

Supervised learning involves training a model on a labeled dataset, where each input is paired with a known output. The goal is to learn a mapping function that can predict outputs for new, unseen inputs. Classical supervised learning algorithms include linear and logistic regression, decision trees, support vector machines, and ensemble methods such as random forests and XGBoost [52]. Supervised learning is widely applied in communication networks for tasks such as traffic prediction, anomaly detection, and resource demand forecasting, where historical

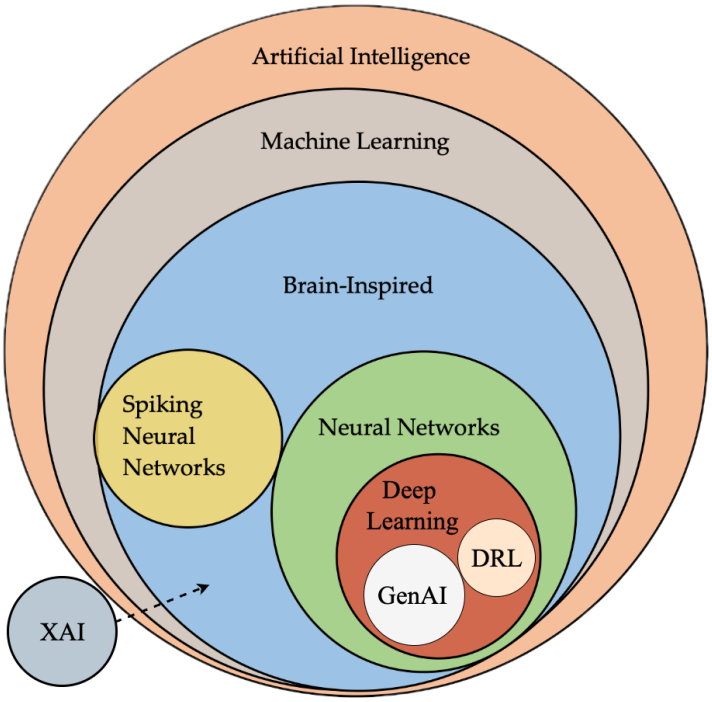


Figure 2.5: A taxonomy of AI approaches. [50].

labeled data is available and can guide the model's learning process. In this thesis, we employed XGBoost in Chapter 6 for anomaly detection in 6G networks.

Unsupervised learning, in contrast, deals with unlabeled data and aims to discover hidden structures or patterns within the data. Common unsupervised methods include clustering algorithms such as k-means and hierarchical clustering, as well as dimensionality reduction techniques like principal component analysis and autoencoders [53]. In networking contexts, unsupervised learning is particularly useful for detecting unusual traffic patterns, identifying groups of similar users or devices, and reducing the complexity of high-dimensional sensor data to facilitate downstream analysis.

RL is a learning paradigm in which an agent interacts with an environment to achieve a specific goal by maximizing cumulative rewards. Unlike supervised learning, RL does not rely on labeled input-output pairs; instead, the agent learns through trial and error, receiving feedback from the environment in the form of rewards or penalties. In this framework, the agent observes the current state of the environment and selects actions that influence the environment's future states. The overall objective of reinforcement learning is to maximize the expected cumulative reward, often expressed as the return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where r_{t+k+1} is the reward received at a future time step and γ is a discount factor that balances immediate and long-term rewards [54]. RL methods are generally classified into value-based and policy-based approaches. Value-based methods, such as Q-Learning [55], focus on learning the state-action value function $Q(s, a)$, which estimates the expected return of taking action a in state s and following the optimal strategy thereafter. The agent selects actions that maximize the Q-value, effectively choosing the best action according to its current knowledge. Policy-based methods, on the other hand, directly learn a policy $\pi(a|s)$, which defines the probability of selecting an action a in a given state s, aiming to optimize expected cumulative rewards without explicitly computing the value

function. Simple tabular methods such as the REINFORCE [56] algorithm can be used to update the policy based on observed returns and sampled trajectories.

Brain-inspired neural networks

Brain-inspired neural networks are computational models designed to mimic the structure and function of biological neural systems. They are composed of interconnected neurons that communicate through signals, inspired by the activity patterns observed in the human brain. Broadly, brain-inspired networks can be classified into conventional neural networks and spiking neural networks.

Conventional neural networks include classical architectures such as feedforward networks, convolutional neural networks, and recurrent neural networks [57]. These networks are capable of learning complex patterns from data through layered transformations and nonlinear activations. Deep learning models, which are essentially deep versions of these networks, leverage multiple layers to extract hierarchical features and achieve state-of-the-art performance in tasks such as image recognition, natural language processing, and sequence modeling. The details of deep learning architectures are discussed in the next step. Spiking neural networks [58], in contrast, communicate via discrete spikes at precise points in time, inspired by the temporal behavior of biological neurons. While they offer advantages in energy efficiency and event-driven processing, they are not the focus of this thesis.

Deep Learning

Deep learning is a specialized subset of neural networks that focuses on learning hierarchical representations of data through multiple layers of interconnected neurons [57]. Each layer in a deep learning model applies a linear transformation followed by a nonlinear activation function to its inputs. Mathematically, for a given layer l, the output $h^{(l)}$ can be expressed as:

$$h^{(l)} = f(W^{(l)}h^{(l-1)} + b^{(l)}) \tag{2.1}$$

where $h^{(l-1)}$ is the input from the previous layer, $W^{(l)}$ and $b^{(l)}$ are the weight matrix and bias vector for layer l, and $f(\cdot)$ is a nonlinear activation function such as ReLU or sigmoid. By stacking multiple layers, the network can learn hierarchical features, capturing low-level patterns in early layers and increasingly abstract representations in deeper layers.

Common deep learning architectures include convolutional neural networks [59], which use convolutional filters to detect local patterns in image and spatial data; recurrent neural networks [60], which incorporate temporal dependencies for sequential data such as text or time series; and transformers [61], which use attention mechanisms to model long-range dependencies and relationships in structured or unstructured data. Training deep learning models involves optimizing a loss function $\mathcal{L}(\theta)$ over the network parameters $\theta = \{W, b\}$, typically using gradient-based optimization methods such as stochastic gradient descent or its variants. The gradients are computed via backpropagation, which efficiently applies the chain rule of calculus to propagate errors from the output layer back through the network.

Deep Reinforcement Learning

DRL is an extension of RL that integrates deep learning to handle large and high-dimensional state or action spaces, which traditional RL methods struggle to address. By using neural

networks as function approximators, DRL can estimate value functions or policies in complex environments, enabling agents to make informed decisions even when the state space is too large for tabular methods [62]. Fig. 2.6 shows how the neural-network-powered agent interacts with the environment.

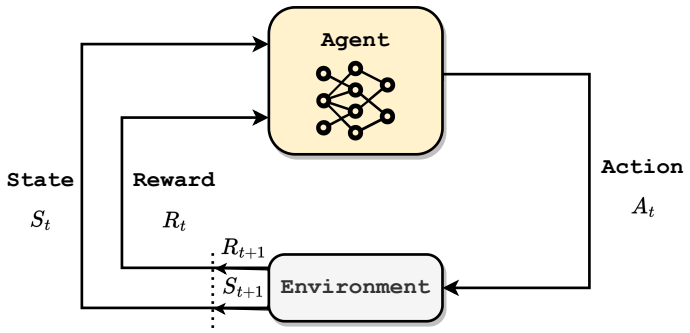


Figure 2.6: Deep Reinforcement Learning.

In value-based approaches such as Deep Q-Network (DQN), a neural network with parameters θ is used to approximate the Q-function $Q(s, a; \theta)$, which maps state-action pairs to expected cumulative rewards. The network is trained to minimize the temporal-difference error:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}\Big[\big(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\big)^2\Big] \tag{2.2}$$

where r is the immediate reward, γ is the discount factor, s' is the next state, and θ^- are the parameters of a target network. By minimizing this loss, the neural network learns to predict the expected return for each action, allowing the agent to select the action that maximizes expected rewards:

$$a^* = \arg\max_a Q(s, a; \theta) \tag{2.3}$$

In this thesis, we designed a DQN-based scheduler to efficiently route networking tasks to a set of LLMs in Chapter 8.

In policy-based approaches, a neural network with parameters θ directly represents the policy $\pi(a|s; \theta)$, which outputs the probability of taking action a given state s. The network is trained to maximize the expected return:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{\infty} \gamma^t r_t\Big] \tag{2.4}$$

Gradients of $J(\theta)$ with respect to θ can be computed using the policy gradient theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(a|s)\, Q^\pi(s, a)\Big] \tag{2.5}$$

Actor-critic methods combine both approaches: the actor network represents the policy $\pi_\theta(a|s)$, while the critic network estimates the value function $V^\pi(s; w)$ or Q-function. The critic provides gradients to improve the actor's policy, stabilizing learning and improving convergence in complex environments.

Generative AI

GenAI represents a class of AI techniques designed to generate new content, data, or solutions that resemble examples from a training dataset. Unlike traditional predictive models, which focus on estimating outputs for given inputs, generative models aim to learn the underlying distribution of the data in order to create novel and realistic samples. Key architectures in GenAI include Generative Adversarial Networks [63], and variational autoencoders [64]. More recently, large-scale models such as LLMs have demonstrated remarkable capabilities in generating human-like text, understanding context, and performing reasoning tasks across diverse domains.

The primary objective of GenAI is to synthesize outputs that are both coherent and meaningful, often leveraging deep neural networks to capture complex patterns and dependencies in high-dimensional data. Applications span multiple modalities, including text, images, audio, and structured data, and have catalyzed advances in creative content generation, simulation, and scenario modeling. The rise of transformer architectures [61] has particularly accelerated progress in GenAI, enabling models to process long-range dependencies, generate contextually consistent outputs, and scale effectively to massive datasets. As GenAI continues to evolve, it establishes itself as a core component of modern AI research, emphasizing creativity, adaptability, and the generation of novel information beyond simple prediction tasks.

Given the strong focus of this thesis on GenAI for IBN, Section 2.5 is dedicated to providing a more detailed discussion on GenAI, with particular emphasis on LLMs.

2.4.3 Explainability in AI

Despite their remarkable predictive capabilities, neural networks and other complex AI models are often considered "black boxes" because their internal decision-making processes are not easily interpretable. The large number of parameters, nonlinear interactions, and hierarchical feature representations make it difficult to understand how inputs are transformed into outputs. This lack of transparency poses significant challenges in high-stakes domains, such as healthcare, finance, and network management, where understanding the rationale behind model decisions is critical for trust, accountability, and compliance with regulations. Without explainability, it is also difficult to diagnose errors, detect biases, or ensure fairness in AI systems [65].

To address these challenges, the field of XAI has emerged, aiming to make AI models more transparent and interpretable. XAI methods can be broadly categorized into model-agnostic and model-specific approaches. Model-agnostic techniques treat the model as a black box and provide explanations without requiring access to internal parameters. Examples include feature perturbation methods, such as LIME [66], and SHapley Additive exPlanations (SHAP) [25], which assign importance values to each input feature based on cooperative game theory principles. Model-specific approaches, on the other hand, leverage knowledge of the model architecture to extract explanations. In neural networks, techniques such as saliency maps, gradient-based attribution, and layer-wise relevance propagation provide insights into which neurons or features contribute most to the model's predictions [65].. Hybrid methods combine aspects of both approaches to balance interpretability and fidelity.

XAI provides a framework for understanding, validating, and trusting AI models. By identifying the most influential features and revealing decision-making patterns, XAI techniques such as

SHAP help bridge the gap between complex neural network models and human understanding, enabling safer and more reliable deployment of AI systems in critical applications. In Chapter 6, we leverage SHAP as the XAI method to explain AI decisions and identify the root causes of anomalies for anomaly resolution in ZSM systems. In the next section, we delve into the details of GenAI.

2.5 Generative AI

This section starts with a brief overview of GenAI, followed by a taxonomy of GenAI methods, and then delves into the details of LLMs architecture, including their design, training, usage, and challenges.

2.5.1 Overview

AI and ML have become foundational technologies across many domains, enabling predictive modeling, pattern recognition, and autonomous decision-making. Traditional AI techniques primarily focus on tasks such as classification, regression, or optimization. In contrast, GenAI represents a paradigm shift toward creating new content, models, or solutions based on learned patterns from data. GenAI models do not merely analyze existing data, they can produce novel outputs, simulate scenarios, or generate alternatives that conform to underlying distributions, making them particularly valuable in research, design, and automation contexts [67].

Generative modeling aims to learn the statistical distribution of a dataset to produce new, realistic instances that resemble the original data. These models are probabilistic in nature and are used across multiple domains, including images, text, audio, and scientific simulations. Core objectives of generative modeling include representation learning, latent space exploration, and the ability to sample unseen but plausible data points [67]. Classical techniques include probabilistic graphical models, while modern approaches rely heavily on neural networks to model complex, high-dimensional data distributions.

2.5.2 A Taxonomy of GenAI Methods

GenAI encompasses a diverse set of methods designed to model data distributions and create novel outputs. While they share the common objective of learning how data is generated, these approaches differ in terms of underlying assumptions, architectures, and training strategies. The following families of models represent the foundations of modern GenAI:

- Autoencoders [64]: Autoencoders learn efficient latent representations by compressing input data into a bottleneck layer and reconstructing it back. Variants such as variational autoencoders extend this concept by introducing a probabilistic latent space, enabling smooth interpolation and sampling of new data points.
- Generative Adversarial Networks [63]: GANs consist of two competing neural networks, a generator and a discriminator, engaged in a minimax game. The generator learns to produce realistic data, while the discriminator distinguishes between real and generated samples. This adversarial training paradigm has been highly successful for image synthesis, video generation, and style transfer, although it suffers from challenges like mode collapse and training instability.

- Autoregressive models [68]: These models generate data sequentially, predicting the next element conditioned on previous ones. They are particularly effective for text and speech generation, with examples including language models such as Generative Pre-trained Transformer (GPT) [69] and audio synthesis models like WaveNet [70]. Autoregressive methods provide high-quality outputs but can be computationally expensive due to their sequential nature.
- Normalizing Flow Models [71]: Flow-based models transform simple probability distributions (e.g., Gaussian) into complex ones through a sequence of invertible transformations. Their advantage lies in providing exact likelihood estimation and efficient sampling, making them suitable for density estimation and generative tasks where probabilistic rigor is important.
- Energy-based models [72]: Energy-based models define an energy function over input data, where low-energy configurations correspond to likely samples. Although conceptually elegant, training energy-based models remains difficult due to the intractability of sampling, yet recent advances have revived interest in applying them to structured data and representation learning.
- Diffusion models [73]: Diffusion-based approaches learn to reverse a gradual noising process applied to data. By iteratively denoising random noise, these models can generate high-quality, diverse samples. Diffusion models such as DALL-E 2 [74] and Stable Diffusion [75] have recently become state-of-the-art in image generation, surpassing GANs in stability and fidelity.

These methods collectively form the backbone of GenAI research and applications. Each offers unique strengths and trade-offs, and their combination (often in hybrid architectures) has enabled breakthroughs in image synthesis, natural language generation, multimodal learning, and scientific discovery. In this thesis, we primarily focus on LLMs, which are autoregressive generative models, as the main foundation throughout most chapters. However, in Chapter 5, we additionally rely on autoencoder-based GenAI models to address anomaly detection tasks.

2.5.3 Large Language Models

LLMs are a prominent class of GenAI models, typically built upon the transformer architecture [61] and containing billions to hundreds of billions of parameters. They are trained on massive text corpora to predict the next token in a sequence, enabling them to generate coherent text, perform reasoning, and adapt to a wide range of language tasks. Here, a token refers to a basic unit of text that the model processes, which can be a word, sub-word, or even a single character, depending on the tokenization scheme [76].

The power of LLMs arises from their scale and architectural design. Empirical studies on scaling laws have shown that increasing model size, dataset size, and compute leads to predictable improvements in language modeling performance. Beyond continuous improvements, LLMs exhibit emergent abilities that do not appear in smaller models but arise as the scale grows [77]. Examples include In-Context Learning (ICL), instruction following, and multi-step reasoning, which enable LLMs to generalize to new tasks without additional gradient-based training. These models have become central to modern AI due to their ability to represent and generate language in a flexible, context-aware manner. Their success has motivated research into efficient training strategies, alignment with human instructions, and integration with external tools, which will be discussed in subsequent sections covering LLM architecture, creation, and usage.

Architectures

LLMs are overwhelmingly built on the transformer architecture [61], which provides excellent scalability, parallelization, and capacity to model long-range dependencies in sequences. The core building block of transformers is the multi-head self-attention mechanism, which allows each token in a sequence to attend to every other token, capturing contextual relationships efficiently. Layer normalization, feed-forward networks, and residual connections stabilize training and improve expressiveness. This architecture makes it possible to scale LLMs to hundreds of billions of parameters while maintaining the ability to process long sequences in parallel. Fig. 2.7 illustrate the transformer architecture.

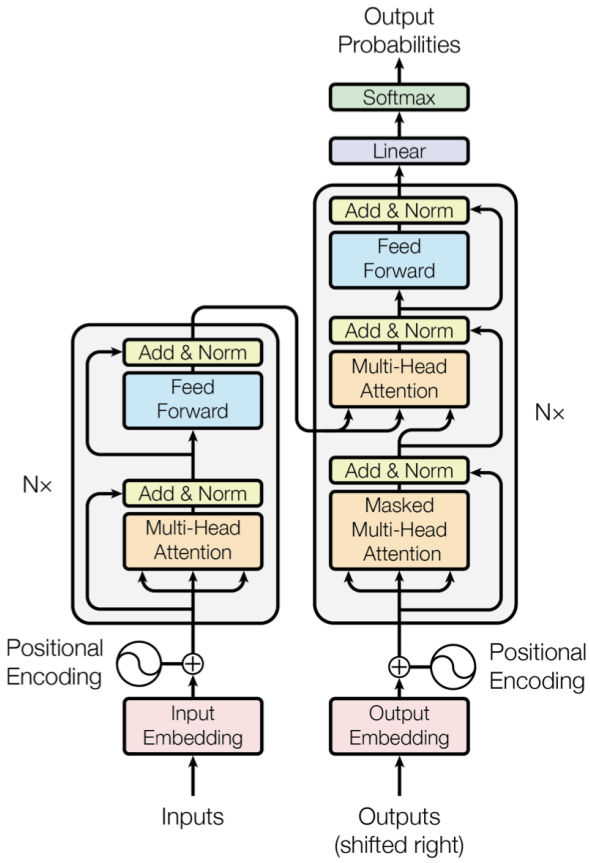


Figure 2.7: The transformer - model architecture [61].

The mainstream architectures of LLMs can be roughly divided into three categories: encoder-decoder, causal decoder, and prefix decoder [78]. The encoder-decoder architecture consists of two stacks of transformer blocks. The encoder transforms the input sequence into latent representations using multi-head self-attention and feed-forward layers, while the decoder autoregressively generates output tokens conditioned on the encoded representations. This design enables tasks that require mapping from one sequence to another, such as translation or summarization, and forms the basis of models like T5 [79] and BART [80]. The causal decoder architecture uses a unidirectional attention mask to ensure that each token can only attend to past tokens and itself. The same transformer stack processes input and output tokens, predicting the next token in the sequence autoregressively. This design is highly effective for language modeling and generation tasks, including text completion, dialogue, and code generation. The GPT series [69]

of models are canonical examples of causal decoder LLMs, demonstrating capabilities such as ICL and instruction following. The prefix decoder architecture modifies the causal decoder by allowing bidirectional attention over a prefix of the input sequence, while maintaining unidirectional attention for the generated output tokens. This enables the model to encode the prefix more effectively, capturing richer contextual information before starting autoregressive generation. Prefix decoders combine some advantages of encoder-decoder models with the efficiency of causal decoders, allowing them to perform tasks such as instruction-following or code generation more effectively. Representative models based on prefix decoders include GLM-130B [81] and U-PaLM [82].

These three architectures capture the majority of modern LLM designs. Encoder-decoder models excel in sequence-to-sequence tasks with complex input-output mappings, causal decoders scale efficiently for autoregressive generation, and prefix decoders provide a hybrid approach that leverages bidirectional context while maintaining autoregressive output generation. Their design choices, combined with large-scale pre-training, enable the emergent abilities that distinguish LLMs from smaller models, such as reasoning, translation, and task generalization.

Building LLMs

Building LLMs is a complex, multi-stage process that requires careful design choices, large-scale data processing, and specialized training techniques. The first step is assembling a massive and diverse text corpus, which can include books, scientific articles, web pages, code repositories, and structured datasets. The quality of the data is critical: it should be clean, diverse, and deduplicated to avoid skewed learning. Efficient tokenization, such as byte-pair encoding or SentencePiece, ensures manageable vocabulary sizes and sequence lengths while retaining important semantic information [83].

Once the dataset is prepared, the model undergoes pre-training. This is the first major stage in building an LLMs, where the model learns general patterns of language from a large, diverse dataset. Using self-supervised learning, the model predicts either the next token in a sequence (autoregressive) or masked tokens within a sequence (encoder-decoder), allowing it to capture syntax, semantics, and world knowledge without explicit labels. This process requires enormous computational resources and can take weeks or months to complete. Research on scaling laws, such as the Kaplan and Chinchilla laws [84], guides the efficient balance of model size, dataset size, and compute. Techniques like mixed-precision training, gradient checkpointing, and distributed parallelism are employed to improve training stability and enable models to scale to hundreds of billions of parameters.

After pre-training, LLMs are adapted to specific tasks or domains through fine-tuning. Supervised fine-tuning uses labeled datasets to optimize performance on tasks like question answering, summarization, or translation. Instruction tuning further improves the model's ability to follow natural language prompts, enabling it to generalize to tasks it has not seen before [83]. Additionally, Reinforcement Learning from Human Feedback (RLHF) aligns the model with human preferences, ensuring that its outputs are helpful, safe, and behave as expected [85]. These fine-tuning steps allow LLMs to effectively leverage the general knowledge gained during pre-training while performing well in real-world applications.

Once trained and fine-tuned, LLMs are ready for adaptation and inference. Prompt engineering is used to guide the model toward producing desired outputs, while few-shot or ICL enables

it to perform new tasks by simply providing examples within the input [86]. Integration with external tools, such as calculators, search engines, or domain-specific knowledge bases, extends the model's capabilities beyond what it learned during pre-training. Efficient inference at scale is achieved through deployment frameworks that leverage batching, quantization, and memory optimization, allowing the model to handle large inputs quickly and reliably.

Finally, building LLMs involves several practical considerations. Compute and memory resources must be carefully managed, particularly for very large models, and data limitations may require augmentation or synthetic generation when existing corpora are insufficient. Rigorous evaluation is essential, using both task-specific benchmarks and tests of generalization, while ongoing monitoring helps detect undesired behaviors such as hallucinations, inconsistencies, or biases. Addressing these factors ensures the creation of LLMs that are powerful, versatile, and safe for deployment. Fig. 2.8 presents a simplified LLM-building pipeline.

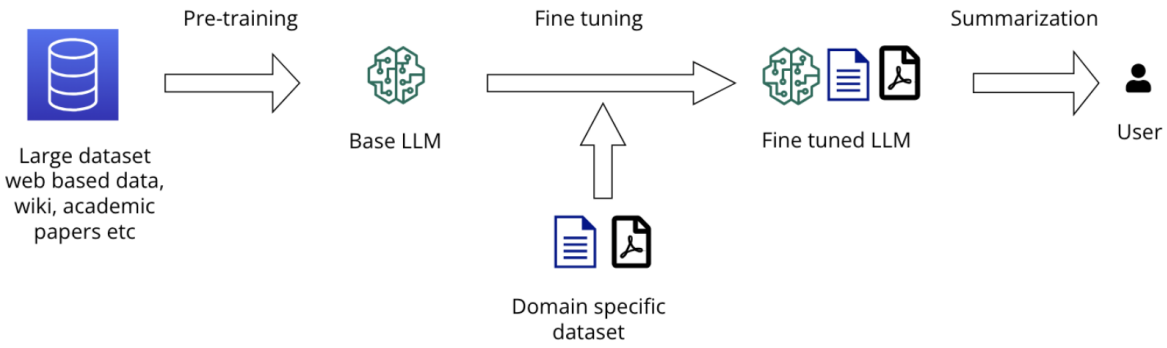


Figure 2.8: Building LLMs pipeline [87].

Adapting and Using LLMs

After pre-training, LLMs often need to be adapted to specific tasks or domains to achieve optimal performance. Traditional fine-tuning updates all the model's parameters on task-specific datasets, which can be computationally expensive and demand large amounts of labeled data. To make fine-tuning more efficient, Parameter-Efficient Fine-Tuning (PEFT) methods have been developed. Techniques like Low-Rank Adaptation (LoRA) [88], adapters, and prefix-tuning modify only a small subset of parameters while keeping the majority of the model frozen, reducing computational costs while preserving the knowledge learned during pre-training. Another approach, freeze tuning, selectively freezes most layers and trains only a few top layers or specialized modules, striking a balance between efficiency and adaptability. These strategies allow LLMs to be tailored for specific tasks or domains without the full cost of retraining the entire model.

In addition to weight-based adaptation, LLMs demonstrate impressive capabilities for ICL, which allows them to perform new tasks simply by providing examples or instructions at inference time, without updating the model's parameters. Variants of ICL include few-shot prompting, where a small number of input-output examples are provided; zero-shot prompting, where the task is described only using natural language instructions; and Chain-of-Thought (CoT) prompting, where intermediate reasoning steps are included to guide complex decision-making. CoT prompting has been shown to significantly enhance performance on arithmetic, logical, and

multi-step reasoning tasks [89]. Another powerful approach is Retrieval Augmented Generation (RAG), where the model dynamically accesses external knowledge sources or documents during generation. By incorporating relevant context on the fly, RAG improves factual accuracy, domain coverage, and up-to-date knowledge without retraining the model, making it particularly useful for tasks where pre-training data may be outdated or limited [90].

In practice, adapting LLMs often combines fine-tuning, PEFT, and ICL strategies, depending on the task requirements, available compute resources, and data constraints. Fig. 2.9 shows both adaptation techniques. Prompt engineering plays a central role, as carefully designed prompts can guide the model toward the desired behavior. Advanced approaches may involve multi-turn prompting, agent-based orchestration, or tool-augmented generation, enabling LLMs to perform complex workflows, reasoning, and decision-making tasks effectively.

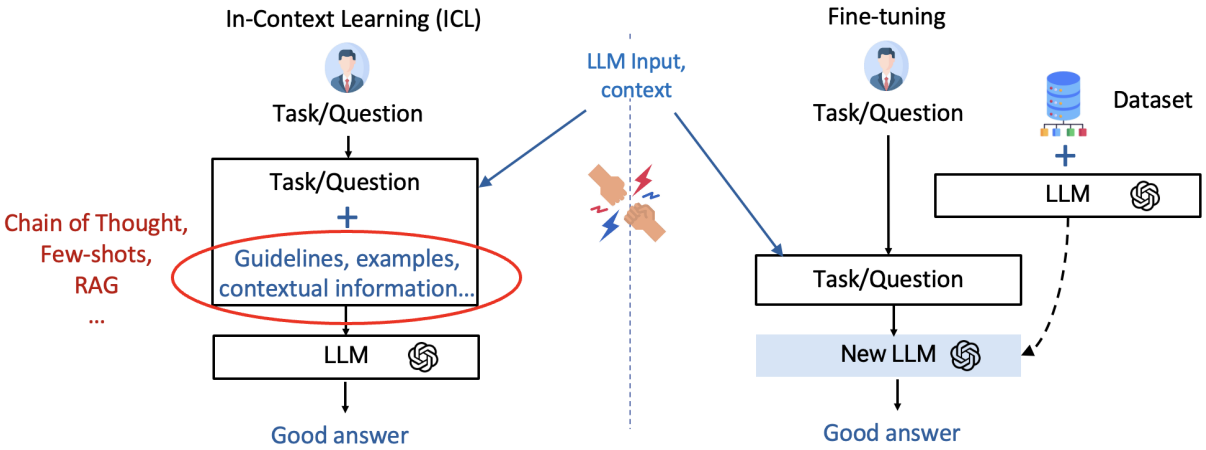


Figure 2.9: Fine tuning and In-Context Learning.

Applications and Use Cases

LLMs, with their generative, reasoning, and contextual understanding capabilities, are applied across a wide range of tasks and domains. These models are available as open-source options, such as Llama, Mistral, and Falcon, which allow researchers and developers to modify and deploy them freely, as well as closed-source commercial models like GPT-4, Claude, and Bard, which are provided as APIs with usage restrictions [78]. In NLP, LLMs excel at text generation, summarization, translation, sentiment analysis, and question answering. Techniques such as ICL and few-shot prompting enable rapid adaptation to new tasks with minimal labeled data, while RAG improves factual accuracy and domain relevance. These approaches also support advanced knowledge synthesis and reasoning, enabling multi-step problem solving in areas such as medical question answering, scientific literature review, and legal document analysis. Combining RAG with CoT reasoning allows LLMs to provide coherent, context-aware outputs [89].

In software engineering and multimodal applications, LLMs generate code, debug programs, and explain complex logic, with CoT prompting aiding multi-step coding tasks such as algorithm design or problem decomposition [91]. Fine-tuning and PEFT methods enable specialization in specific programming languages, frameworks, or enterprise environments. LLMs are also integrated with vision, audio, and other modalities to create systems capable of text-image, text-video, and text-audio understanding, such as image captioning, video summarization, and

speech-to-text with semantic reasoning. Hybrid architectures combining LLMs with autoencoders, diffusion models, or specialized transformers facilitate rich cross-modal reasoning and content generation.

LLM agents and agentic frameworks, powered by ICL, extend these capabilities by orchestrating complex workflows autonomously. Using multi-turn prompts, tool-augmented reasoning, and dynamic retrieval, agents can plan, make decisions, and solve problems interactively without additional training [92]. Agentic frameworks, such as LangChain[1], AutoGPT[2], and BabyAGI[3], provide structured environments for chaining LLM reasoning steps, integrating external tools, and managing multi-step goals, enabling advanced autonomous systems in enterprise, research, and creative applications. Overall, LLMs provide a versatile foundation for language understanding, reasoning, content generation, multimodal integration, and autonomous workflows, with adaptability through fine-tuning, PEFT, and ICL expanding their practical impact across industries.

Challenges in LLMs

Despite their remarkable capabilities, Large Language Models face several significant challenges that impact their performance, deployment, and reliability.

- Hallucinations and factual errors [93]: LLMs may generate outputs that are plausible in form but incorrect in content. This phenomenon, commonly referred to as hallucination, is particularly problematic in applications requiring factual accuracy, such as scientific research, medical advice, or decision support. Techniques like RAG and grounding LLMs on structured knowledge sources can mitigate, but not entirely eliminate, this issue.
- Computational resources and efficiency [93]: Training and running LLMs require enormous computational power, memory, and energy. Models with hundreds of billions of parameters can only be trained on large GPU clusters with optimized distributed training frameworks. Even inference for large-scale deployment can be costly, motivating research into model compression, quantization, and parameter-efficient fine-tuning strategies like LoRA or adapters.
- Scalability and integration complexity [93]: Deploying LLMs in real-world systems involves scaling them for multiple users, integrating them with existing software pipelines, and maintaining low-latency responses. Handling large context windows, maintaining conversational state, and orchestrating multi-modal inputs further complicate deployment.
- Verification and trustworthiness [93]: Ensuring that LLM outputs are correct, reliable, and consistent is a major challenge. Unlike rule-based systems, LLMs do not guarantee adherence to logical constraints or domain knowledge. Verification techniques, including prompt-based checks, external fact verification, and post-processing pipelines, are critical to establish trust in applications that require high-stakes decisions.
- Ethical and alignment considerations [93]: LLMs may produce biased, inappropriate, or unsafe outputs if not carefully aligned with human values. Alignment tuning using techniques like RLHF is necessary, yet ensuring comprehensive alignment across diverse contexts remains an open problem.

[1]https://www.langchain.com
[2]https://github.com/Significant-Gravitas/AutoGPT
[3]https://github.com/yoheinakajima/babyagi

2.6 Conclusion

In this chapter, we provided a structured overview of the state of the art in 6G networks, IBN, AI, and GenAI. We discussed the architecture, key enablers, and management challenges of 6G networks, followed by an examination of IBN as a paradigm for abstracting network complexity through high-level intent expression, including its lifecycle, taxonomies, and standardization efforts. We then reviewed AI techniques and subsequently explored GenAI, focusing on LLMs, agents, memory mechanisms, tool integration, and agentic frameworks. Adaptation strategies such as fine-tuning, PEFT, and RAG were presented, highlighting how these models can perform reasoning, planning, and task execution in complex domains.

Overall, this overview establishes a clear understanding of the key concepts, technologies, and techniques in 6G networks, IBN, and GenAI, providing the conceptual foundation for the following chapters, in which we present our contributions regarding the application of GenAI in IBN. These contributions begin with intent translation in Chapters 3 and 4, followed by intent assurance in Chapters 5 and 6, and conclude with GenAI-driven operations in Chapters 7 and 8.

Part I

Generative AI for Intent Translation

Chapter 3

Intent Translation to Service Configuration

3.1 Introduction

The current IBN model of expressing intents still requires significant effort in writing the JSON and YAML structures, demanding a detailed comprehension of the model specified by the North-Bound Interface (NBI). This process is not always straightforward, and adhering to the structure of these NBIs is time-consuming. A natural evolution for IBN is to move beyond the use of human-readable structures and transition towards natural language. An example of an intent request using natural language for deploying a Communication Service (CS) for a 5G network serving XR users is:

Example 3.1.1. *I need a network composed of three XR applications: an augmented reality content server, a mixed reality collaboration platform, and a virtual reality simulation engine. Each application requires 4 vCPU and 2 Gigabytes (Gbytes) of memory. All XR applications are connected using 5 Gbytes/sec links. The clients are connected through a 5G network located in the Nice area and tolerate a maximum latency of 5 ms.*

In this example, the Intent comprises different sub-intents specific to the technological domain involved in supporting this CS. Indeed, one part is dedicated to the needed computing resources to run the mentioned applications on the Edge/Cloud, another one for networking, and the last part to the RAN. Once the intent is specified, there are many required steps to reach the deployment and propagation of the intent's objective up to the different infrastructures composing the technological domains (Edge/Cloud, networking, and RAN). These steps correspond to the intent LCM procedures: *(i) Intent decomposition* that extracts the information on each technological domains that need to be involved in the deployment phase; *(ii) Intent translation* that translates decomposed Intents to Infrastructure-Level Intents (ILIs) specific to each domain. Then, *(iii) Intent negotiation*, *(iv) Intent activation*, and *(v) Intent assurance* are the steps that enforce the Intent on the technological domain infrastructure.

Intents LCM based on natural language represents the simplest form of IBN-enabled system communication with the network. Nevertheless, natural language's unstructured and ambiguous nature poses challenges for IBN systems in extracting essential information and producing precise low-level configurations. Additionally, users from diverse regions can add complexity, necessitating an approach capable of interpreting Intents across multiple human languages, even in the presence of grammatical errors. Fortunately, with the rapid explosion in the NLP area,

LLMs become very powerful in understanding human languages.

To address the gap in the literature regarding intent LCMs, this chapter introduces a novel, LLM-centric high-level architecture designed to manage intents life-cycle from an E2E perspective. This involves defining, handling, and enforcing intent objectives. Our architecture leverages cutting-edge AI advancements, particularly LLMs, to tackle all previously mentioned E2E intent life-cycle procedures. We demonstrate the efficiency of this architecture by upgrading the EURECOM 5G facility [3] with natural language intent handling, i.e., decomposing and translating Intents using LLMs. Subsequently, intent activation and assurance are managed using its existing functionalities [3]. This upgrade relies on open-source, state-of-the-art LLMs incorporating HF to learn from previous experiences. The major contributions of this chapter are as follows:

- *E2E IBN LCM architecture:* We propose a comprehensive, LLM-centric architecture that spans all stages of IBN from intent definition, decomposition, and translation to activation and assurance. This architecture ensures seamless interaction with the Network Management System (NMS) for all users, including those with limited domain knowledge who will use natural language instead of ILIs. It leverages the latest AI advancements, particularly LLMs.
- *Identification of key open issues and challenges:* Within our architecture, we identify and discuss the main open issues related to IBN that need to be addressed. These challenges are crucial for advancing beyond the current state-of-the-art and providing a solid foundation for future research and development in this area.
- *LLM-based intent decomposition and translation system:* To showcase the effectiveness of the proposed architecture, we develop an innovative system within the 5G EURECOM facility [3] using LLMs for intent decomposition and translation. This system utilizes few-shot learning and HF to transform natural language intents into ILIs.
- *Real-world deployment and testing:* Our proposed framework is implemented and tested in real-world scenarios using a single NVIDIA A100 GPU with 40GB of vRAM, leveraging the Code Llama LLM [94]. This demonstrates the practical applicability of our architecture on the 5G facility [3], enabling the configuration of network services using natural language.

3.2 Related Works

Various research efforts have addressed numerous challenges in the various components of IBN. For instance, the authors of [95] tackled the intent translation process by developing a chatbot that elicits context-specific information from users of packet-optical networks. To address intent activation, researchers in [96] proposed an intent negotiation framework to resolve conflicts arising from limited resource availability in IBN. Additionally, *Zheng et al.* [97] proposed an intent assurance solution for data center IBN systems, utilizing specific data preparation procedures and ML models for time series forecasting. Furthermore, several research works have proposed E2E IBN architectures. For example, *Velasco et al.* [41] presented a scalable 5G architecture by integrating secure ML-powered IBN, fostering application-level resilience and intelligence. Concurrently, with the explosion of LLMs in the NLP domain, the research community is shifting towards utilizing and standardizing them in the context of networking and telecommunications.

A few studies have employed LLMs in IBN [11, 98]. Researchers in [11] aimed to generate Python code from natural language intent with LLMs, enabling infrastructure management.

However, their approach relied on a customized Python library to execute the generated program, limiting its applicability to other infrastructure management systems. Similarly, authors in [98] use LLMs along with other AI techniques to achieve autonomous fault localization, strategy generation, and strategy verification. While these approaches demonstrate the potential of LLMs for specific, well-structured intents, they do not address the full lifecycle of E2E intent LCM, which includes decomposition, negotiation, translation, activation, and assurance across multiple domains. Our system generates standardized ILI, such as NSD and RAN Descriptor (RAND), aimed at universal acceptance across public and private 6G infrastructures adhering to these standards. This is achieved through a two-stage process: decomposing the intent into different domains and generating standardized ILI for each domain. By handling multi-domain intents in a standardized manner, our approach is able to process more complex and potentially ambiguous intents, where references to different domains may be scattered across multiple sentences, a scenario not explicitly addressed in prior works. To implement this, we leverage open-source LLMs, which provide full control over the model behavior, unlike the closed-source ChatGPT-based solutions used in [11, 98]. The effectiveness of the proposed architecture is demonstrated through the decomposition and translation steps of E2E intent LCM, while activation and assurance rely on the existing functionalities of the EURECOM 5G facility [3].

3.3 System Design

In this section, we begin by introducing our high-level architecture that aims to handle the life-cycle of intents from their definition up to their admission and deployment over the infrastructure, considering the context of multi-domain CS deployment in 5G and beyond. Then, we expose the different open challenges related to the realization of the architecture's objectives and discuss some research directions. Finally, we introduce novel solutions that address two critical challenges related to natural language-based decomposition and translation employing an LLM-centric approach. We demonstrate the application of the proposed solutions using the 5G facility [3] as an example to deploy a CS spanning over two technological domains: Edge/Cloud and RAN.

3.3.1 Proposed Architecture

To effectively address intent life-cycle, we propose a high-level architecture for natural language-based intents, as depicted in Fig. 3.1. In this system, the manager or user interacts with the Multi-Domain Intent Handler (MDIH) component to deploy a CS across various infrastructure domains. The MDIH assists the manager or user in correcting, updating, or negotiating the intent to ensure that the latter can be deployed. This interaction is envisioned to be similar to using a chatbot. Each infrastructure domain is managed by a Local Management System (LMS) that utilizes ILI to deploy the corresponding CS components on the infrastructure. An example of ILI is the NSD defined by the ETSI NFV group [15] for deploying services on virtualized platforms, or the helm chart model used by Kubernetes[1] for deploying cloud-native services. Our proposed system is designed to be infrastructure and network architecture agnostic, offering native support for all technological domains comprising 6G, i.e., cloud, edge, transport, and RAN. The intent-based architecture illustrated in Fig. 3.1 aims to abstract the underlying infrastructure as much as possible. The only infrastructure-specific aspect lies in the translation step (at the Domain Intent Handler (DIH)), which is essential for generating ILI per domain.

[1] https://kubernetes.io

In many cases, ILI also abstracts the complexities of the underlying layers.

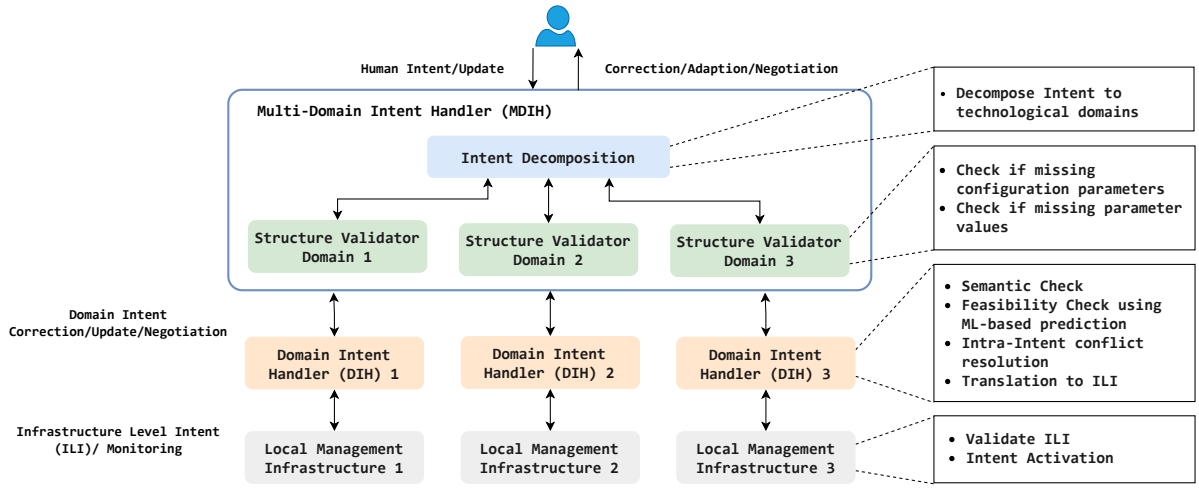


Figure 3.1: High-level architecture design to handle natural language-based intents life-cycle.

In the proposed solution, the CS is defined using natural language. No restriction on the formulation model (or grammar) is imposed; the user is free to define the intent by using natural language. The intent is then handled by a MDIH, which belongs to the CS provider and is integrated into the OSS/BSS as a component. The internal structure of MDIH includes an intent decomposition module, tasked with decomposing the intent into sub-parts corresponding to each domain. The intent decomposition module should understand the semantic (i.e., the meaning) of the intent to decompose it; in Fig. 3.1, we assume three domains, e.g., Edge/Cloud intent, networking intent, and RAN intent. While considering three domains in this example, the proposed approach can be generalized to a higher number of domains. To decompose the intent, we rely on LLMs, known for their ability to understand natural language and extract semantic information. After decomposition, each subpart is sent to the structure validator module, which validates the intent's structure for each technological domain. The validation process checks that all required parameters and associated values to deploy a CS on a domain, are included in the intent. For instance, an intent to deploy a CS on Edge/Cloud should indicate the software image location and resources needed for deployment. If a parameter or value is missing, the structure validator request the user to correct the intent. The process ends when the intent structure is valid. As stated earlier, a chatbot-like system interacts with the user, providing proposals to correct the intent formulation until final validation. The chatbot approach corrects the intent's structure, avoids intra-intent conflicts, and ensures the intent's feasibility on the infrastructures. The objective is to prevent any structural error before forwarding the request to the DIH.

Subsequently, each domain-specific intent is handled separately by a DIH, which has the role of doing the semantic validation of the intent and the translation to ILI specific to an LMS handling a domain-specific infrastructure. The different actions the DIH has to execute are in the following order:

1. Check if there are intra-intent conflicts between the objectives. For instance, if the intent requests radio throughput higher than a value (V1) and interference lower than a value (V2), then a conflict exists as increasing the radio throughput will also increase the interference due to the usage of a higher data rate modulation scheme. In this case, a correction

of the intent through the chatbot is proposed to the end user to avoid conflicts. When the intent is conflict-free, the following action is triggered; otherwise, an update is requested from the intent owner through the chatbot.

2. Check if the intent can be satisfied, which involves the feasibility check process. Indeed, the latter aims to check if the intent objectives can be fulfilled on the domain-specific infrastructure considering the available resources. This step is essential as it is equivalent to an admission control process, avoiding that the intent is deployed while its objectives cannot be sustained. This requires that the DIH uses a prediction of the resource evolution of an infrastructure aiming to decide if the intent can be accepted. In this context, an external module to the DIH will run AI/ML prediction models that consume monitoring information collected from the LMS. The predicted model is built per LMS.
3. Translate the intent to ILI and activate the intent by sending it to the LMS. The translation process will be done using an LLM-based module that translates natural language to ILI as expected by LMS. The LMS will validate the ILI; if any error is detected, a request to correct the ILI is sent to the DIH. This closed control loop between the DIH and LMS allows the improvement of the LLM Knowledge Base (KB) to enhance future translation accuracy.
4. Request the collection of intent's Key Performance Indicators (KPIs) to start the intent assurance step, which consists of validating the intent performance and achieving the requested service level agreement. Besides, the assurance step consists of deriving LCM decisions using reinforcement learning or policy-based approaches to correct the performance if a degradation is detected or correct inter-intent conflicts.

3.3.2 Challenges and Open Issues

Although the proposed architecture offers a promising E2E framework for intent LCM, its realization faces several challenges. These challenges encompass various aspects of intent processing procedures, including intent decomposition and structure validation, intent semantic validation and intra-intent conflict resolution, as well as intent translation.

Intent decomposition and structure validation

Intent task decomposition poses a significant challenge in IBN, prompting researchers to explore novel AI-based methods. However, the complexity and ambiguity of natural language make decomposing tasks particularly challenging. On the other hand, validating the intent structure involves ensuring the presence of required parameters in the user's intent. This presents multiple challenges: How do we extract the corresponding intent for each domain? How do we validate the intent structure? How do we interact with the user? LLMs are actively being developed to support various NLP tasks. Owing to their context detection and human language comprehension abilities, they can be utilized for intent decomposition to understand which intent concerns which domain. This feature is beneficial in 6G, as the latter involves heterogeneous technological domains. Subsequently, structure validation can be achieved through conventional rule-based methods, such as human intervention (via HF), or by employing context-aware techniques to extract essential parameters from user inputs. For instance, named entity recognition is beneficial for detecting specific required parameters. Nonetheless, incorporating HF can introduce security vulnerabilities when dealing with erroneous or manipulated input. To address these risks, a recommended approach is to establish robust HF validation mechanisms. For instance, one effective strategy is to engage experts to validate feedback before it is used by the system.

Intent semantic validation and intra-intent conflict resolution

On the one hand, semantic validation involves verifying that the user's intent aligns with the capabilities of the underlying system. Researchers have employed NLP techniques, such as LLMs, and KGs, to identify key concepts and relationships in a given text [99]. They then utilize AI/ML approaches, such as deep learning, to predict and verify conformity with the underlying system for feasibility checks. Notably, LLMs can be used synergistically with AI/ML to perform semantic validation and feasibility checks, respectively. On the other hand, conflicts may arise between users and service providers due to a misalignment between service requirements and the capabilities of underlying resources. To address these conflicts, intent negotiation modules have been developed to initiate negotiation processes [96]. These modules generate alternative intents based on resource availability, allowing users to accept or reject them. Challenges include resolving conflicting objectives within intents and assessing the feasibility of infrastructure. Despite proposed frameworks, standardization in intent negotiation is still being developed. In this context, LLMs represent a promising option due to their advanced reasoning capabilities. However, LLMs require substantial knowledge before making decisions, necessitating further training. Consequently, one approach is to emphasize creating training and evaluation telecommunications datasets. Subsequently, training LLMs on these datasets, such as [100], could create expert LLMs in this domain capable of efficiently making informed decisions to resolve conflicts within intents.

Intent translation

IBN systems are a significant area of research interest, with researchers proposing new AI-based methods to translate users' intents into network configurations, operations, and maintenance strategies. Tools such as chatbots [95] have emerged to simplify the intent translation process. Additionally, reasoning approaches have been used in conjunction with NLP to improve translation performance. On the other hand, LLMs also offer promising reasoning abilities for translation tasks. However, in the context of IBN, a few challenges remain: How do we translate natural language intents into ILIs? How do we validate the correctness of the generated ILIs? First, LLMs can be adapted to this task using fine-tuning or few-shot learning. However, this requires having a high-quality KB that contains intents and ILIs couples, which is scarce. Second, rule-based methods such as HF or advanced NLP based on deep learning approaches can be used to validate the results of the LLMs.

3.3.3 Our Intent Decomposition and Translation framework

Fig. 3.2 depicts an instantiation of the architecture outlined in Fig. 3.1, focusing on deploying natural language intents using the EURECOM 5G facility [3]. Two technological domains are considered: Edge/Cloud and RAN. Initially, a natural language intent is processed by the MDIH, using an LLM to decompose it into Edge/Cloud intent and RAN intent. These are then forwarded to respective DIHs, also employing an LLM to translate domain-specific intents into ILIs. In the Edge/Cloud domain, the ILI is represented by an NSD; in the RAN domain, it is a RAND. Although only the NSD JSON structure is illustrated in Fig. 3.2, the RAND is also implemented using JSON and configures RAN parameters such as slicing, throughput, maximum latency, bandwidth parts, etc. These configurations described by the RAN intent pertain to non-real-time configuration, equivalent to the O1 configuration of Open RAN (O-RAN)[2]. Subsequently, the NSD is sent to the Network Function Virtualization Orchestrator (NFVO),

[2]https://www.o-ran.org

responsible for deploying network services within the 5G infrastructure. Simultaneously, the RAND is delivered to the RAN orchestrator, which operates similarly to the Service Management Orchestration (SMO) of O-RAN, managing service deployment within the 5G infrastructure. In this latter, the RAND configuration is enforced through the SMO/RAN Intelligent Controller (RIC) using rApps and xApps. To illustrate, Example 3.1.1 will be decomposed into two intents: First, the Edge/Cloud intent would be: *"I want three applications: an augmented reality content server, a mixed reality collaboration platform, and a virtual reality simulation engine. Each application requires 4 vCPU and 2 Gbytes of memory."* Second, the RAN intent would be: *"The clients are connected through a 5G network located in the Nice area and tolerate a maximum latency of 5ms."*. Then, the system will create: (i) an NSD containing three applications with the requirement in CPU and RAM for each one; and (ii) a RAND containing one xApp. This xApp will be deployed on top of the RIC by the RAN orchestrator and will manage the radio resource allocation to ensure satisfying a 5ms latency for all users.

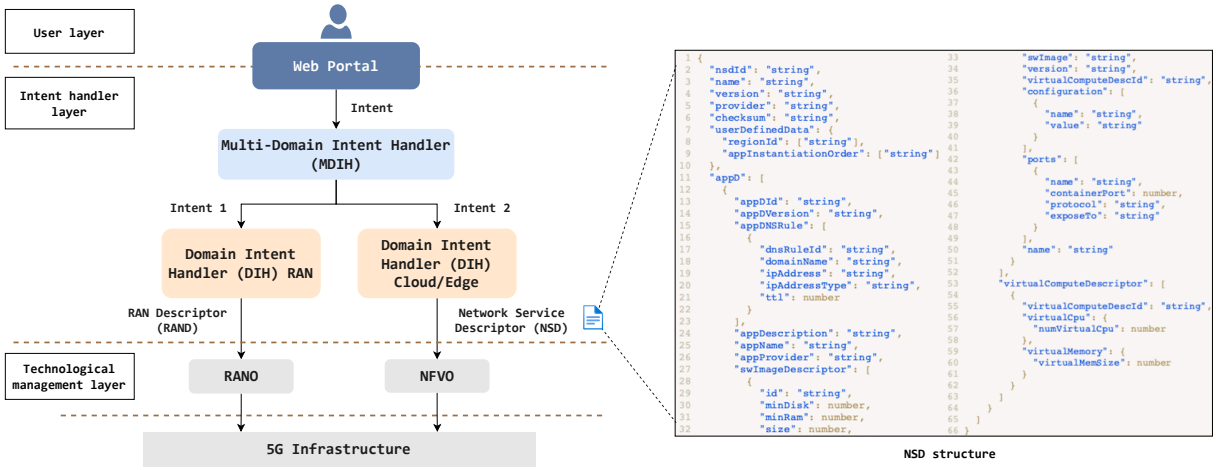


Figure 3.2: Natural language-based intent LCM in the EURECOM 5G facility [3].

Fig. 3.3 illustrates the detailed LLM-based decomposition and translation framework. Both the MDIH and DIHs utilize the same LLM-based system design. This shared design employs a three-stage process to establish an efficient intent-to-2-domain-intents pipeline (for MDIH) and Intent-to-ILI pipeline (for DIHs). In Stage 1, historical examples are retrieved from the central decomposition KB: KB_d (for MDIH), or domain KBs: Edge/Cloud KB_c; and RAN KB_r (for DIHs). In Stage 2, these examples are utilized to decompose the intent (for MDIH) or generate the ILI (for DIHs). Here, an LLM is employed through ICL. Finally, in Stage 3, users provide HF regarding the quality of the decomposition (for MDIH) or the generated ILI (for DIHs). This feedback is incorporated into the corresponding KB if validated by an administrator. Below, we discuss each stage in more detail. We will use the index $j \in \{d, c, r\}$ to represent the three systems.

1. *Few-shot examples extraction:* In this stage, the input, denoted as Q_{uj}, is processed by a sentence transformer model. The model maps each query into a dense vector in a latent semantic space, capturing semantic relationships beyond surface-level token matches. This allows queries with similar meanings, but different wordings, to be closely positioned in the vector space. The objective is to retrieve relevant examples from the corresponding KB_j, where each entry is organized as a tuple (Q_{ij}, A_{ij}). Here, Q_{ij} represents historical queries, and A_{ij} comprises LLM responses, validated either manually or through HF. Retrieval of examples involves measuring the cosine similarity between the new input query Q_{uj} and

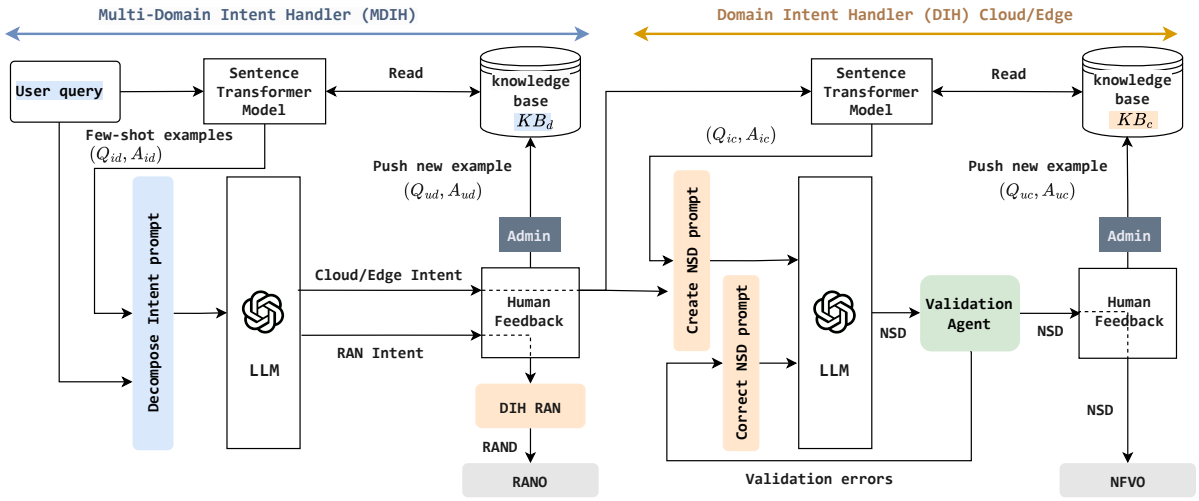


Figure 3.3: LLM-based intent decomposition and translation system.

each query Q_{ij} in KB_j. Cosine similarity is chosen as it effectively captures the angular distance between vectors in high-dimensional spaces, making it robust to differences in magnitude, which is common in sentence embeddings. It is calculated as:

$$S(Q_{uj}, Q_{ij}) = \frac{Q_{uj} \cdot Q_{ij}}{\|Q_{uj}\| \|Q_{ij}\|} \tag{3.1}$$

where $\cdot$ represents the dot product, and $\|\cdot\|$ denotes the Euclidean norm of the vector. Subsequently, the AI model extracts a set of n tuples, denoted Top_n, representing the most similar historical examples. These tuples, in the form (Q_{ij}, A_{ij}), are then forwarded to the next stage for further processing. Alternative NLP similarity measures (e.g., Manhattan distance, Euclidean distance, or token-level metrics) were considered, but cosine similarity over dense embeddings provided the best balance of semantic relevance and computational efficiency in our experiments.

2. *Intent decomposition or intent translation:* The previously generated few-shot examples are assembled into a prompt, which serves as an input for the LLM. This prompt provides explicit instructions to guide the LLM in its task, with a clear directive stating either *"Your job is to decompose intent"* or *"Your job is to generate the ILI."* Alongside this instruction, the prompt includes additional rules defined by the administrator, the few-shot examples (Q_{ij}, A_{ij}), and the new input Q_{uj}. Subsequently, the LLM will generate the decomposition or the ILI. However, it is essential to note that effective performance requires the LLM to have a substantial context window, particularly when dealing with a large number of examples (n is significant). Besides, the LLM should be pre-trained on code-related data, given its mission to translate intent into JSON structures. In this context, we have selected the CodeLlama model [94]. A single ILI can comprise more than 2k tokens, and providing numerous few-shot examples necessitates a substantial LLM context window. CodeLlama models offer a maximum context window of 100k tokens, making them well suited for our use case.

3. *Intent translation validation:* In intent translation, the validation agent ensures the LLM's output aligns with the ILI's structure through three steps: *syntax validation* for ILI JSON conformity, *semantic validation* using regular expressions to check parameter types and values, and *correlation validation* for parameter relationship consistency. It also confirms

ILIs compatibility with EURECOM's NFVO/RAN orchestrator NBIs. Validated ILIs are forwarded to users via the Graphical User Interface (GUI) for HF; otherwise, the LLM receives a prompt to correct the NSD with specific instructions on detected errors.

4. *Human feedback:* Indirect reinforcement learning from HF is a crucial component of our system. When a user is satisfied with either the intent decomposition or the generated ILI, they can provide HF to the administrator for inclusion in the KB_j. Otherwise, the user can correct the LLM response and include the corrected version. This feedback, which consists of the query and the correct response (Q_{uj}, A_{uj}), is used to generate more accurate and complete intent decompositions or ILIs in the future.

3.4 Performance Evaluation

The section is structured into two subsections: *Experimentation setup*, which details the experimental setup, and *Experimentation results*, which presents the performance of our framework.

3.4.1 Experimentation Setup

Our experimental setup consists of two machines, each equipped with 36 Intel(R) Xeon(R) Gold 6240R CPUs running at 2.40GHz. The second machine also has an Nvidia A100 GPU with 40GB of vRAM. The first machine runs the Kubernetes-based test cluster and the EURECOM 5G facility components, whereas the second machine hosts the LLM-based system. The same LLM is used for both intent decomposition and translation. We use the LangChain[3] framework for handling LLMs and ChromaDB[4] to store KB embeddings. We gathered our foundational KBs data from a variety of sources, including EURECOM's past and ongoing research projects. We compared several other popular open-source LLMs, including Mistral 7B and Llama 13B, and found that the CodeLlama model with 34B parameters[5] produced the best results in both decomposition and translation tasks. We used the MPNet v2 sentence transformer model[6]. We set the maximum number of tokens to 3k, as the longest ILI in our initial KBs is 2k tokens long. We set n to 10 for the decomposition task and 4 for the translation task, as it is the maximum number of few shot examples that fit on the GPU. Additionally, we set the LLM temperature to 0.1. It is important to note that these parameters are flexible and can be adjusted to meet future requirements.

3.4.2 Experimentation Results

This subsection evaluates the system's performance in the context of intent decomposition and translation. The evaluation focuses on decomposing natural language intents into the two technological domains (Edge/Cloud, RAN) and subsequently translating Edge/Cloud intents into NSDs. Since both the Edge/Cloud and RAN domains rely on software-based components, the evaluation results from the Edge/Cloud domain can be generalized to the RAN domain. In order to gather performance data, feedback was solicited from 10 volunteering users within EURECOM. Each volunteer assessed our platform by creating 10 CSs and providing feedback on intent decomposition and Edge/Cloud intent translation.

[3]https://www.langchain.com
[4]https://www.trychroma.com/
[5]https://huggingface.co/TheBloke/CodeLlama-34B-Instruct-GPTQ
[6]https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Intent decomposition and translation feedback

To enable IBN, our system must understand users' intents. We validated this understanding with a rating approach. After each experiment, volunteers evaluated both the intent decomposition and translation steps using a scale of 0 to 5, with 5 being the highest score. Fig. 3.4 shows the average user rating from 10 CS creations. Using averages helps mitigate individual biases among users. If one or two users provide biased ratings, these are balanced out in the mean calculation. Our approach demonstrated significant efficiency in intent decomposition, reflected by an initial rating score of about 4.5. However, users initially expressed dissatisfaction with the intent translation component, requiring modifications before submitting NSDs to the NFVO. Despite this, the average rating consistently exceeded 3.5, indicating that only minor adjustments were necessary, primarily related to configuring unfamiliar parameters. Over time, the system learned from feedbacks and past examples, generating NSDs identical to those desired by users.

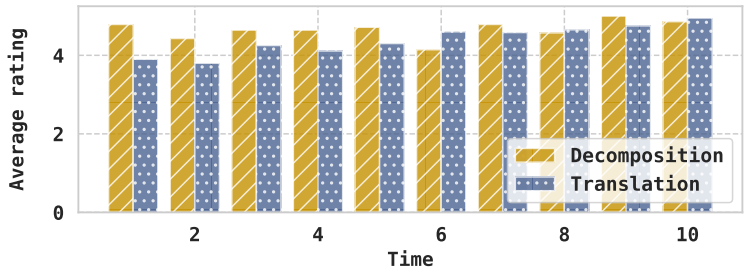


Figure 3.4: Mean rating score throughout time.

Intent decomposition and translation latency

Fig. 3.5 illustrates the relationship between the number of requested Edge/Cloud applications and the time required to decompose intents and generate a valid and correct NSD using the LLM-based system. From this figure, we can observe that the intent decomposition process adds only a few seconds to the overall E2E time (yellow surface). This is because the process involves breaking down the intent into smaller parts, resulting in generating almost the same number of tokens in every decomposition task. However, as the number of requested applications increases, the translation time also increases (blue surface). As illustrated in Fig. 3.5, the E2E time exceeds 2 minutes when translating requests containing more than 3 Edge/Cloud applications. This is because the system generates more tokens for each Edge/Cloud application in the NSD, resulting in increased intent translation time.

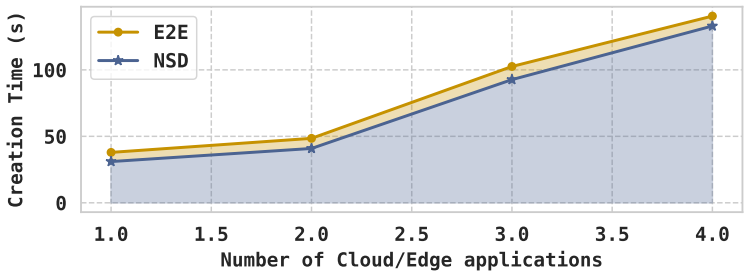


Figure 3.5: Impact of the number of applications on decomposition and translation time.

Validation agent iterations

Throughout all NSD generations, the validation agent was executed only once for each generation. This demonstrates that the output of the CodeLlama LLM consistently adheres to the NSD structure from the first iteration, showcasing CodeLlama's effective learning of the format only from few-shot examples. This proficiency highlights its ability to generate correct JSON structures due to its training on code-related data.

3.5 Conclusion

This chapter presented an innovative intent LCM architecture that revolutionizes network configuration and management by enabling natural language interaction with networks. By leveraging cutting-edge advancements in AI, particularly LLMs, the proposed architecture automates the entire intent lifecycle, from intent decomposition and translation to intent negotiation, activation, and assurance. This approach significantly simplifies network management, eliminating the need for expertise in low-level configurations and enabling network administrators to focus on high-level network objectives. To validate the effectiveness of the proposed solution, we presented an initial implementation and evaluated it in real-world deployments. The results demonstrated the architecture's capability to translate natural language intents into actionable network configurations.

However, the contributions in this chapter address only a single management task, i.e., creating and configuring a network service. In real-world scenarios, management tasks are numerous and are typically handled by the OSS, which is responsible for service planning, configuration, service monitoring, infrastructure health management, performance assurance, and more. These broader management tasks are explored in the next chapter, where we design an LLM-based approach capable of handling the full spectrum of network management tasks from high-level natural language intents.

Chapter 4

Intent Translation to Service Planning

4.1 Introduction

The era of 6G is expected to support a wide range of advanced services, such as XR and VR, which will enhance human lifestyles by enabling critical use cases like teleoperated medical surgery and autonomous transportation [29]. These applications will impose stringent QoS and Quality of Experience (QoE) requirements that traditional networking systems are unable to meet. To address this, existing infrastructures will not only be upgraded with additional functionalities like RIS and cell-free networks, but methods for sharing and virtualizing these infrastructure components will also become more advanced and efficient [101]. Consequently, the 6G network management layer, enabled by the OSS and responsible for managing and orchestrating this infrastructure, must be highly advanced and AI-powered, as emphasized by standardization institutions such as ETSI and 3GPP [102, 103].

The next-generation OSS will have diverse and heterogeneous responsibilities. Some of the key functionalities are: (i) managing the lifecycle of E2E 6G services (e.g., creation, activation, deletion), which may span one or more 6G domains, such as the RAN, TN, CN, and Edge/Cloud; (ii) managing infrastructure resources that are shared across different services and with other OSSs (e.g., those belonging to different network operators); and (iii) enabling observability and monitoring, as well as supporting ZSM, which involves detecting and resolving anomalies without human intervention in services or infrastructure components [104]. These requirements have led standardization institutions to propose various APIs to facilitate interactions between users and OSSs, using simplified structures to declare user intention (intent), a concept known as IBN [42].

The IBN concept was introduced as a simplified interface to the OSS, allowing users to define high-level goals in a declarative manner. However, current IBN systems are limited: they rely on heterogeneous APIs for specific OSS functionalities (e.g., TMF APIs), requiring users to manually learn and orchestrate multiple APIs, which is time-consuming and error-prone. Moreover, existing research often focuses on generating a single configuration artifact, such as the body for one API call [105] (presented in Chapter 3), neglecting the broader problem of translating a high-level intent into a sequence of management actions subject to dependencies and conditions. This Chapter addresses this challenge by introducing OSS-GPT, a system that performs intent-driven planning and execution: given a high-level service intent, OSS-GPT autonomously generates a sequence of OSS API calls, including their payloads and parameters, and executes

them to achieve the desired network outcome. The system employs multi-agent LLMs that collaboratively perform hierarchical planning, handling conditional dependencies between API calls and dynamically adapting to new functionalities or endpoints by leveraging only the OSS API specifications [106]. This approach extends traditional intent translation toward dynamic service lifecycle management, enabling fully automated, multi-step orchestration while maintaining adaptability and scalability. The main contributions of this work are therefore manifold:

- Designing a general-purpose IBN system that allows natural language interaction with the OSS.
- The IBN system uses multi-agent LLMs to satisfy high-level user intent. Together, these agents plan API calls, execute them, and report the results to the user, enabling a chatbot system. Among the agents, we propose a specialized agent trained solely for generating request bodies following specific standards, called blueprint generator.
- The system was implemented at EURECOM's OSS, adhering to ETSI standards for deploying and configuring 6G services using the NSD [107]. In this context, all agents are trained using ICL at inference time, except for the NSD generator. The latter was trained using LoRA [88], resulting in an LLM specialized in generating NSDs.
- Evaluation was performed in real-world conditions using the EURECOM's OSS [3], demonstrating that all these API calls can be replaced by a single chatbot, enabling natural language IBN.

4.2 Related Works

To address the complexities of current IBN systems, we emphasized the use of natural language in the previous chapter (Chapter 3). However, the system was limited to a single task, i.e., configuring only one network service using the NSD and the RAND. On the other hand, Fuad et al. in [9] proposed an LLM-based approach for intent profiling and translation, but their work is limited to the TN domain where they generate BGP and firewall configurations from natural language. In addition, researchers in [108] also explored natural language intent profiling using a different NLP technique, called Named-Entity Recognition (NER), for intent detection and translation applied specifically to the TN domain. However, their approach focuses on a single, clear intent definition and one translation step. A significant limitation of NER tools is their inability to handle different languages of intent or address issues like typing errors, unlike LLMs, which are more flexible in dealing with such complexities [109]. These studies highlight that no work has yet implemented a natural language-based approach for all the stages of IBN that addresses complex Intents across a wide range of tasks, while encompassing all 6G technological domains.

Our goal is to enable natural language-based IBN using multi-agent LLMs. This system translates high-level natural language intents into a sequence of OSS API calls, executing them sequentially to fulfill the user's original request. However, existing approaches, such as offline introspective plan-then-execute methods [110] or the ReAct framework [111], face challenges in adapting to API feedback and generating viable plans. RestGPT [112], attempts to address these issues by employing a more dynamic planning process. It adopts an iterative, coarse-to-fine online planning mechanism, i.e., when given a complex instruction, an LLM first generates high-level natural language sub-tasks, which are then mapped by another LLM to more granular API call plans. While RestGPT excels at handling complex tasks for simpler API definitions,

it faces two primary challenges: (i) when applied to 6G OSS with long, standardized request bodies (service blueprints), this method struggles with generating these complex request structures, and (ii) it relies heavily on few-shot learning for each LLM agent, which is tightly coupled to the API specifications. As a result, any changes in the API specifications require generating new few-shot examples, making the learning process inefficient.

In OSS-GPT, we opt for an online planning and execution approach, similar to RestGPT, but address its challenges in two ways. First, to improve the efficiency of generating standardized service blueprints, we introduce a specialized LLM trained using the LoRA technique. This LLM is specifically designed to generate the complex, standardized service blueprints required for 6G OSSs. Second, to overcome the inefficiency of few-shot learning, we designed OSS-GPT to learn directly from the API specifications, without the need for additional few-shot examples. Unlike RestGPT, which employs a coarse-to-fine approach to translate high-level intents into multiple natural language intents using few-shot examples, OSS-GPT, trained solely on API specifications, directly selects and executes individual API calls, one by one. This approach enables autonomous updates within the OSS, driven solely by changes in the API specifications. In summary, we present a fully autonomous, user-friendly, intent-driven OSS for next-generation 6G network management that offers enhanced efficiency by eliminating the need for manual learning updates.

4.3 System Design

In this section, we will define the problem related to planning and executing API calls, followed by the presentation of our solution design, OSS-GPT.

4.3.1 Problem Definition

Let the natural language intent be represented as a textual input I. The objective is to map I to an output, a natural language response R, which summarizes the results of executing $\mathcal{C}$. The response R provides an intent-driven summary of I, where $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ is a sequence of API calls. Executing these calls sequentially ensures that the intent I is fulfilled. Each API call C_i is defined by an endpoint $E_i \in \mathcal{E}$, where $\mathcal{E} = \{E_1, E_2, \ldots, E_m\}$ is the set of available API endpoints. Each endpoint E_i is characterized by:

$$E_i = \langle m, p, i_s, o_s \rangle,$$

where m is the Hypertext Transfer Protocol (HTTP) method (e.g., `GET`, `POST`, `PUT`), p is the endpoint uniform resource identifier structure, i_s is the expected input parameters (e.g., body, headers, query), and o_s is the structure of the output or response.
The problem is defined as finding a mapping $f : I \to R$, where:

$$f = (f_p, f_e, f_s),$$

with:

- $f_p : I \to \mathcal{C}$: generates the plan (i.e., the sequence of API calls) by selecting and ordering API endpoints $E_i \in \mathcal{E}$.
- $f_e : \mathcal{C} \to \mathcal{D}$: executes the planned API calls $\mathcal{C}$, producing a set of execution results $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$, where D_i is the output of C_i.

- $f_s : \mathcal{D} \rightarrow R$: synthesizes the execution results $\mathcal{D}$ into a natural language response R, summarizing the outcome of the API sequence.

The planning step must consider:

- Logical dependencies between API calls (e.g., when one API's output is required as input for another).
- Constraints defined by the input and output schemas of the APIs.
- The overall intent I, ensuring that $\mathcal{C}$ addresses all necessary sub-tasks implied by I.

The execution step must:

- Generate or retrieve the input required for each API call C_i based on i_s.
- Monitor the responses to ensure they conform to the expected o_s.
- Handle any errors or replan f_p dynamically as needed.

The summarization step must:

- Extract key information from $\mathcal{D}$ based on the outputs o_s of the APIs.
- Generate a concise and coherent natural language response R that aligns with intent I.
- Handle ambiguities or incomplete information in $\mathcal{D}$ using contextual cues from I.

4.3.2 Solution Design

To tackle the aforementioned problem, we propose a multi-agent LLM framework, called OSS-GPT. Our solution involves multiple LLMs that work collaboratively to fulfill the intent I and provide the natural language response R. These LLMs function as a chatbot, providing OSS users with the results of their intents. As shown in Fig. 4.1, OSS-GPT is composed of the following components:

Assistant

The role of the *Assistant* is to maintain a chatbot-like interaction with the user. It accepts natural language input I and responds in natural language output R. This agent is trained using ICL, which handles two cases: (i) If the user asks general questions not related to OSS functions, the *Assistant* will respond directly to the user's query; (ii) If the user requests intents to be executed by the OSS, the *Assistant* forwards the intent to the *Planner* and waits for the *Reporter*'s response to provide it to the user. This design allows for seamless natural language interaction between the OSS users and the system.

Planner

The *Planner*'s primary responsibility is to generate a sequence of API calls $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ that will fulfill the intent of the user I. Upon receiving the intent from the *Assistant*, the *Planner* analyzes the request and determines the appropriate set of API calls needed to achieve the desired outcome. To accomplish this, the *Planner* is trained using ICL, where it leverages a knowledge base containing brief descriptions of available API endpoints. Due to context size limitations, the full specification of each endpoint $E_i \in \mathcal{E}$ cannot be included in the context; instead, the

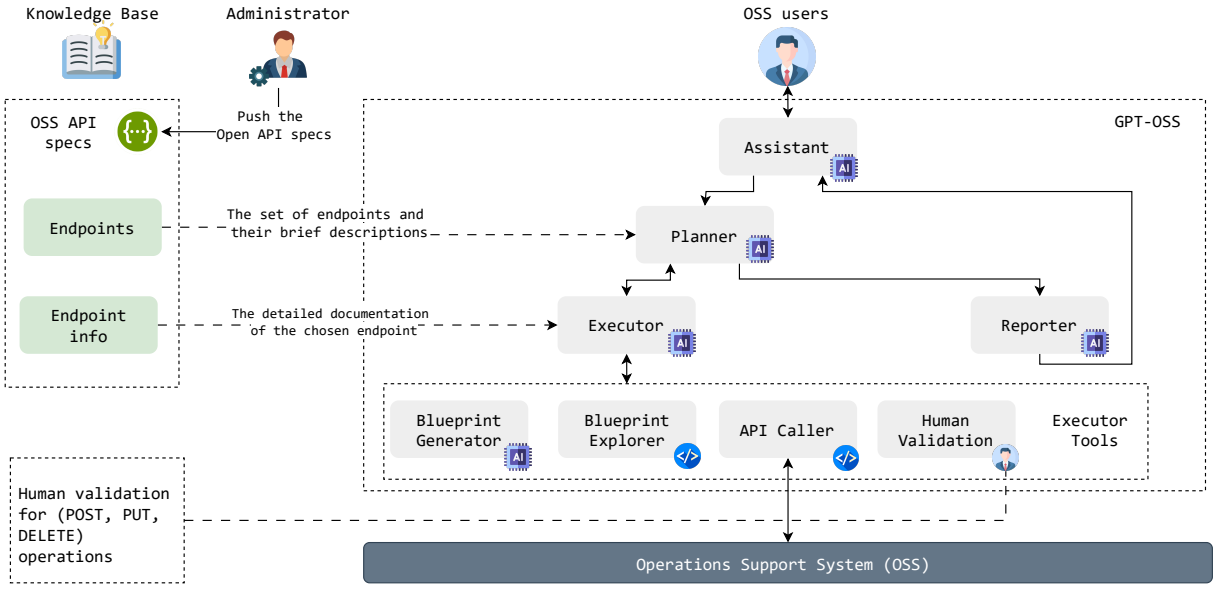


Figure 4.1: OSS-GPT design.

Planner only has access to these concise descriptions, which provide enough information to select relevant API calls without overloading the context. The *Planner* operates by selecting and sending one API call at a time to the *Executor*. Once the *Executor* has executed the API call:

- If the previous execution was successful, the *Planner* proceeds to generate the next API call in the sequence.
- If an error or failure occurs during execution, the *Planner* will dynamically replan, adjusting the sequence of API calls as needed to fulfill the intent I.
- If no further planning is required, meaning I has been fulfilled, the *Planner* requests the *Reporter* to generate the natural language response R.

This dynamic replanning mechanism ensures that the system remains flexible and robust, even when some executions fail or need to be modified in response to changing conditions.

Executor

The primary responsibility of the *Executor* is to execute each API call selected by the *Planner* and return the execution results. The *Executor* is also trained using ICL, which provides detailed descriptions of the available endpoints E_i to guide the execution process. Upon receiving the API call instructions from the *Planner*, the *Executor* not only executes the request but also plan the necessary tools required for the successful execution of the API call. Although additional tools may be employed, the Executor typically relies on the following three core tools:

- *Blueprint Generator*: Used to generate the necessary blueprints for POST requests, such as standardized structures (e.g., NSDs). This tool is an LLM specifically trained to generate request bodies from natural language inputs.
- *Blueprint Explorer*: Utilized for retrieving existing information for PUT requests (e.g., when updating or modifying data), ensuring relevant context is included in the execution.

This is a straightforward program that performs GET requests to the OSS to fetch the required data.

- *API Caller*: Responsible for executing API endpoints by making appropriate HTTP requests, including all necessary parameters (headers, body, query parameters, etc.). This is a straightforward program that executes all API calls via the OSS.
- *Human Validation*: Ensures user approval is obtained before executing critical API operations, such as POST requests for creating services, PUT requests for modifications, and DELETE requests. This tool is requested before the *API Caller* is invoked for these operations. Users can activate or deactivate this tool based on their preference for system autonomy, enabling either human oversight or a fully autonomous intent-driven system.

For example, when the *Executor* receives a request to create a 6G service, such as a 6G CN, it first invokes the *Blueprint Generator* to produce the necessary blueprints for the POST request. Afterward, the *Executor* uses the *API Caller* to send the request to the appropriate API endpoint E_i, based on the specified HTTP method m, and waits for the response. Upon receiving the response, the *Executor* performs the following tasks:

- It verifies that the response conforms to the expected output structure o_s.
- If the response is successful and matches the expected schema, the *Executor* returns the result D_i to the Planner for further processing.
- If the response indicates an error, or if there is a mismatch with o_s, the *Executor* notifies the *Planner*. In such cases, the *Planner* may initiate a replan.

In addition to these tasks, the *Executor* is responsible for providing debugging information, which supports dynamic replanning by the *Planner*. This ensures that the system remains resilient and capable of adapting to unexpected issues that may arise during the execution of the API calls.

Reporter

The *Reporter* agent is an LLM that takes the conversation history between the *Planner* and *Executor*, along with the user's intent I, as input, and generates a natural language report R as output. It is trained using ICL and sends the response to the *Assistant* agent.

4.4 EURECOM's OSS-GPT

In this section, we first present the design of EURECOM's OSS and then proceed with the instantiation of OSS-GPT within EURECOM's OSS framework, focusing specifically on the *Blueprint Generator* tool, which is specific to the OSS. This contrasts with the other agents described in the previous section, which are general-purpose.

4.4.1 EURECOM's OSS

The design of EURECOM's OSS is inspired by the ETSI Management and Orchestration (MANO) architecture standards, adhering to the principles of the Service-Based Management Architecture (SBMA). The OSS is structured into multiple microservices, each dedicated to a specific role: (i) Service Manager, responsible for deploying and configuring 6G services, using

the NSD from ETSI to define service configurations; (ii) Infrastructure Manager, tasked with provisioning virtualized and physical infrastructure; (iii) Monitoring Manager, deploying components responsible for KPI collection and monitoring of services and infrastructure [113]; and (iv) Fault Manager, which detects and resolves faults within services and infrastructure, leveraging the KPIs collected by the monitoring manager. These microservices communicate with each other via API calls, coordinated through a centralized API gateway that serves as the entry point and houses the complete API specifications. Without OSS-GPT, users interact with the API gateway through HTTP requests, which are routed to the appropriate microservice based on the type of request (e.g., service creation, infrastructure provisioning, KPI collection, or ZSM activation). Figure 4.2 illustrates the high-level design of EURECOM's OSS.

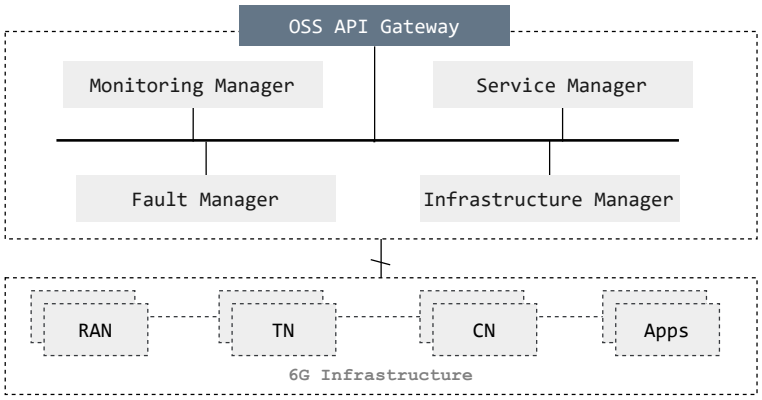


Figure 4.2: EURECOM's OSS components.

To integrate OSS-GPT within EURECOM's OSS, training all the AI components (LLM agents) shown in Fig. 4.1 is essential. For all AI components except the *Blueprint Generator*, training with EURECOM's OSS API specifications is sufficient, as these components rely on ICL using the specifications during inference, thus only pushing the API specs into the knowledge base is required. However, the *Blueprint Generator* has a distinct role: it must generate accurate NSDs directly from natural language inputs. To achieve this, we developed a specialized LLM specifically trained for NSD generation. We opted for fine-tuning in this case because, in our previous work [105], this task was explored using ICL with a 34B LLM. While ICL demonstrated high accuracy, the complexity of the task resulted in significant generation times (over one minute for three applications). This delay created a bottleneck, particularly in multi-agent LLM systems where other agents must wait for completion. Fine-tuning enabled us to create a smaller, specialized LLM that eliminates the need for extensive context learning, processing only the natural language intent to generate accurate NSDs efficiently.

4.4.2 Blueprint Generator - The NSD-expert

The Blueprint Generator tool is responsible for generating NSDs when service creation or configuration requests are present in the user's intent. In the previous chapter (Chapter 3), we built a similar system using open-source LLMs adapted with few-shot learning to generate NSDs. The main limitation of this approach is the generation time, which increases when multiple applications are requested in the intent. In few-shot learning, several examples of NSDs need to be included in the context, saturating it and consequently increasing generation time. Additionally, only LLMs with a large number of parameters (e.g., 34B as seen in Chapter 3) perform well as few-shot learners, which is inefficient at this step due to high latency. To address this, we pro-

pose creating a smaller LLM (3B parameters) through fine-tuning, capable of generating NSDs with low latency and without requiring few-shot examples. We call this model the *NSD-expert*.

To train the *NSD-expert*, two main steps are crucial: (i) dataset generation; and (ii) LLM training. Dataset generation posed a significant challenge because there is no open dataset available containing natural language intents paired with their corresponding NSDs. To address this, we developed a dataset from scratch, containing examples that map natural language inputs to their respective NSD representations. Another challenge was adapting existing open-source LLMs through fine-tuning without compromising their general abilities to generate coherent text and respond to human queries. The goal was to create an LLM capable of generating NSDs when required, while still understanding and responding to natural language in a chatbot-like manner. This capability is essential for use cases where the user might need to modify a generated NSD via natural language interactions. To achieve this, a method was needed to inject the specialized knowledge of NSD generation into an existing LLM without erasing or degrading its prior knowledge. Next, we describe these two steps.

Dataset collection & augmentation

In the experiments described in [105], we requested volunteers to test an ICL-based NSD generator LLM. Volunteers provided scores for the LLM's output and corrected any mistakes in the generated NSDs. This process resulted in a small, high-quality dataset containing 100 entries of natural language intents paired with their corresponding NSDs. To augment this dataset, we leveraged the ICL capabilities of LLMs. Specifically, we used few-shot learning by providing 3-4 examples from the existing dataset in the LLM's context, accompanied by an instruction to generate a new example. For instance, let

$$\mathcal{D}_{\text{existing}} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

represent the original dataset, where x_i denotes the natural language intent and y_i is the corresponding NSD. Given a subset $\mathcal{D}_{\text{few-shot}} \subseteq \mathcal{D}_{\text{existing}}$, the LLM generates a new example (x_{n+1}, y_{n+1}) by conditioning on the context:

$$P(y_{n+1}|x_{n+1}, \mathcal{D}_{\text{few-shot}})$$

The newly generated examples were reviewed for quality and subsequently added to the knowledge base, effectively expanding the dataset iteratively. This method ensured that the augmented dataset retained high relevance and correctness.

Training

For training, we employed LoRA [88], a PEFT technique designed to reduce computational complexity while effectively adapting LLMs to specific tasks. As shown in Fig. 4.3, LoRA adds trainable low-rank matrices to the pre-trained weights of an existing LLM, freezing most of the original weights and significantly reducing the number of parameters that need adjustment. This approach was chosen due to its favorable trade-off between adaptation efficiency and resource requirements, allowing large models to be fine-tuned on limited hardware without significant loss in performance. Alternative parameter-efficient techniques, such as knowledge distillation, RAG-based approaches, or teacher-student methods, could also be considered. For example, teacher-student distillation can approximate the performance of a large model at a lower computational cost.

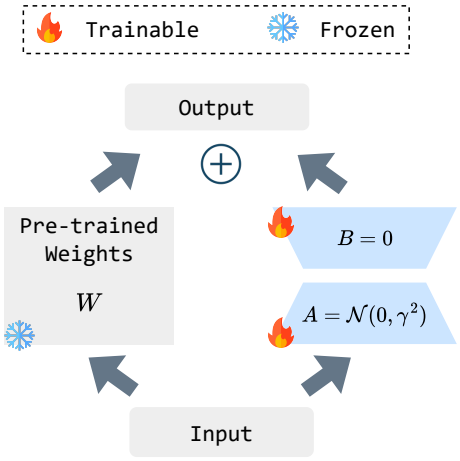


Figure 4.3: LoRA [88].

Let $W \in \mathbb{R}^{d\times k}$ denote a weight matrix in the pre-trained LLM. In LoRA, this matrix is approximated as:

$$W \approx W_0 + \Delta W$$

where W_0 is the frozen pre-trained weight, and

$$\Delta W = AB, \quad A \in \mathbb{R}^{d\times r}, \quad B \in \mathbb{R}^{r\times k}, \quad r \ll \min(d,k)$$

ensures the rank of the adaptation matrices A and B is low, reducing the number of trainable parameters from $d \times k$ to $r \times (d + k)$. The process of fine-tuning can thus be expressed as minimizing the loss $\mathcal{L}$ over the task-specific data:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \ell\big(f_\theta(x_i), y_i\big)$$

where f_θ is the LLM with LoRA applied, and $\ell(\cdot)$ is the task-specific loss function. The adaptation is efficient because only ΔW is updated during training, while W_0 remains static. LoRA enabled us to specialize the LLM for NSD generation while retaining its original language understanding capabilities.

4.5 Performance Evaluation

The section is structured into three subsections: *Experimentation Setup*, which describes the experimental configuration; *Experimentation Results*, which presents and analyzes the performance of the OSS-GPT approach; and *Discussions*, which provides additional insights from the experiment, discusses challenges, and outlines potential future work.

4.5.1 Experimentation Setup

Our experimental setup, illustrated in Fig. 4.4, involves two machines hosting the Kubernetes-based cluster and the OSS-GPT framework, respectively. The first machine, equipped with an Intel(R) Xeon(R) Silver 4314 CPU (2.40GHz) and running Ubuntu 22.04.3 LTS, manages a single-node Kubernetes cluster that hosts the virtualized 6G services and infrastructure components. At EURECOM, we utilize OAI components to create an E2E 6G service, encompassing the RAN (oai-nr-ue and oai-gnb), CN (oai-amf, oai-smf, oai-upf, etc.), and edge applications.

The second machine, a MacBook with an Apple M1 Pro chip featuring an integrated GPU, hosts the infrastructure management components, including OSS components and OSS-GPT, running on Docker and bare-metal, respectively. The OSS-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 8 billion parameters running on the Ollama framework[3].

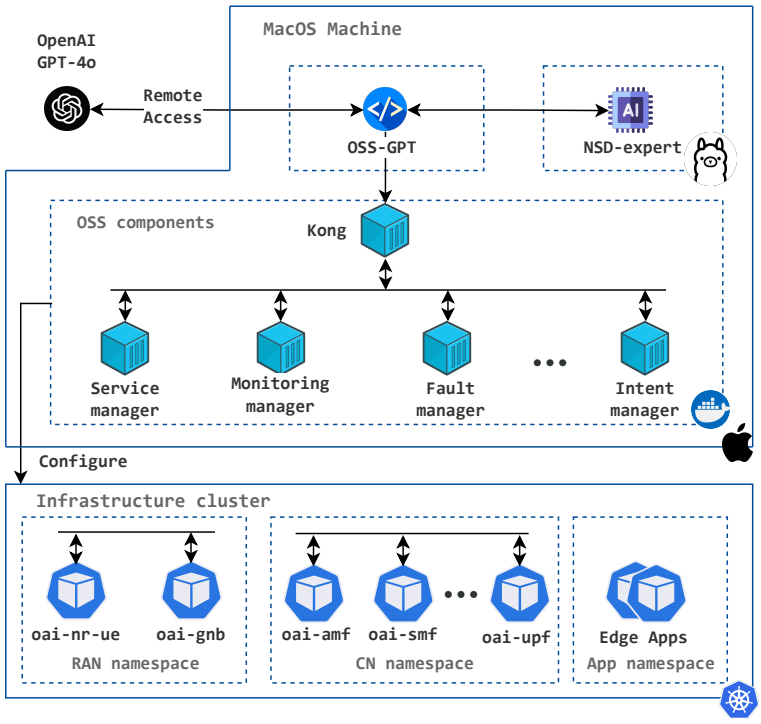


Figure 4.4: Experimentation setup.

The *NSD-expert* model was fine-tuned on a third machine equipped with an NVIDIA A100 GPU with 40GB of vRAM, using Llama 3.2 (3B parameters)[4] as the base model and applying the LoRA technique. For training, we employed the Unsloth framework[5] with a training setup that included 120 steps, a learning rate of 2e-4, and a linear learning rate scheduler. The model was trained using a per-device batch size of 2 with gradient accumulation over 4 steps to simulate a larger effective batch size. Mixed-precision training was enabled, dynamically selecting FP16 or BF16 based on hardware compatibility, and the AdamW optimizer was used with an 8-bit implementation to reduce memory usage.

4.5.2 Experimentation Results

In this subsection, we first evaluate the quality of the *NSD-expert*. Next, we assess the overall performance of the framework, OSS-GPT. Finally, we analyze the cost-effectiveness of OSS-GPT in realistic scenarios, focusing on training costs, inference execution time, and pricing.

[1]https://www.langchain.com/langgraph
[2]gpt-4o-2024-08-06
[3]https://ollama.com
[4]https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct
[5]https://unsloth.ai

NSD-expert evaluation

Fig.4.5 shows the loss function of fine-tuning the Llama3 LLM on the generated dataset. This demonstrates that the LLM's loss function was decreasing, indicating effective training, and it converged after 80 steps. The resulting *NSD-expert* is evaluated in Fig.4.6 against the previous Llama3 version using four metrics: (i) Perplexity, which measures how well the model predicts the next token in a sequence; (ii) BERTScore, which evaluates the semantic similarity between generated and reference text using contextual embeddings; (iii) Cosine similarity, which measures the cosine of the angle between the vector representations of the generated and reference texts; and (iv) Exact Match (EM), which assesses the percentage of exact matches between the generated and reference outputs.

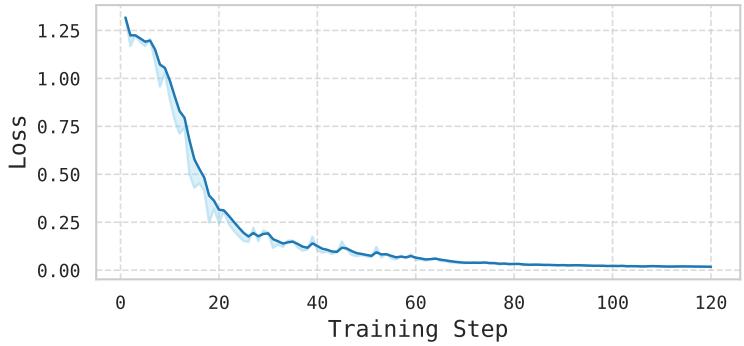


Figure 4.5: Training loss.

As shown in Fig.4.6, the perplexity metric of *NSD-expert* is very low, indicating that the model was confident in generating the JSON files. However, the perplexity of the old version of Llama3 is smaller than that of *NSD-expert*, suggesting that the Llama3 distribution makes more sense for text generation. This is because perplexity is an appropriate metric for evaluating text generation, but it does not effectively assess structured outputs like JSON. To evaluate the quality of the JSON generation, we used Cosine similarity, BERTScore, and EM. The BERTScore and Cosine similarity values are low for Llama3, indicating that this model struggled to generate the JSON structure correctly. In fact, it did not even recognize that the NSD output is a JSON structure. This is further evidenced by the 0 score for Llama3 in the EM metric. In contrast, *NSD-expert* generated highly accurate JSON files, closely matching the reference JSON file, achieving approximately 0.99 for BERTScore and Cosine similarity, and a 60% EM, which is a very strong score. This demonstrates that the *NSD-expert* LLM can be trusted by the OSS-GPT system. Next, we evaluate the whole OSS-GPT system, using also the *NSD-expert*.

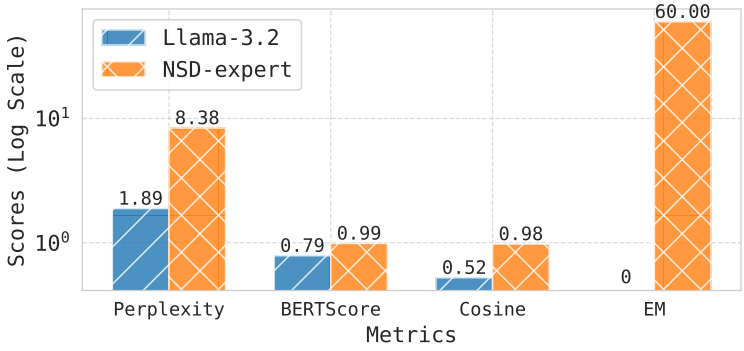


Figure 4.6: Metrics.

OSS-GPT evaluation

As mentioned earlier, the OSS-GPT functions as a chatbot from the OSS user's perspective, enabling interactions to obtain information about services and infrastructure, create or delete them, and more, using high-level intents. An example of this interaction is illustrated in Fig. 4.7, where OSS-GPT generates a plan consisting of a set of API calls, including parameters and body requests for each API call, and executes them sequentially in the background. The minimum number of API calls required to satisfy an intent is referred to as the golden path. For example, in Fig. 4.7, two questions are presented. In the first question, the golden path consists of one API call, a single GET request to fetch the list of services. In the second question, the golden path consists of three API calls: a GET request to retrieve the list of services, a GET request to obtain information about a specific service, and a POST request to create a subservice with gNB configuration (generated by the blueprint generator).

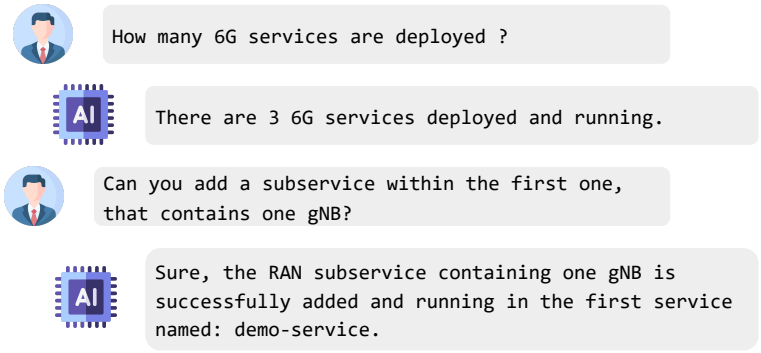


Figure 4.7: Chatbot example when trust is activated.

To evaluate the quality of OSS-GPT, we prepared a dataset of intents generated by domain experts, ranging from simple to complex ones requiring multiple API calls (golden paths). For this purpose, we defined 8 levels of intents, corresponding to 8 golden paths (n). A golden path of i indicates that OSS-GPT needs to make at least i API calls (where $i = 0$ means that OSS-GPT responds entirely on its own without making any requests to the OSS). Fig. 4.8 illustrates two key metrics: (i) the number of API calls executed to fulfill an intent (Path Cost) compared to the golden path (orange curve); and (ii) the accuracy of OSS-GPT in successfully fulfilling intents for each golden path (blue). It can be observed that for intents requiring fewer than 4 API calls, OSS-GPT achieved an accuracy close to 1.0, successfully fulfilling all intents, which represents a perfect score. However, for intents with golden paths greater than 4, the accuracy gradually decreases as the complexity of the intents increases and more API calls are required. Notably, for golden path 7, OSS-GPT maintained an accuracy of 0.80, which is a strong performance given the complexity of the task. This demonstrates the capability of LLMs in planning and executing API calls effectively. On the other hand, as the golden path increases, OSS-GPT tends to make more API calls than necessary. This indicates occasional errors in the planning process. Nevertheless, thanks to its replanning mechanism, most intents are eventually fulfilled successfully.

Cost assessment

In this part, we evaluate the costs associated with training and inferring OSS-GPT. Our approach involves creating a custom LLM, designated as the blueprint generator (i.e., *NSD-expert*), while leveraging GPT-4 for other components, as it is currently the leading reasoning model. Thus, the costs can be divided into two categories: training and inference: (i) The training of the *NSD-*

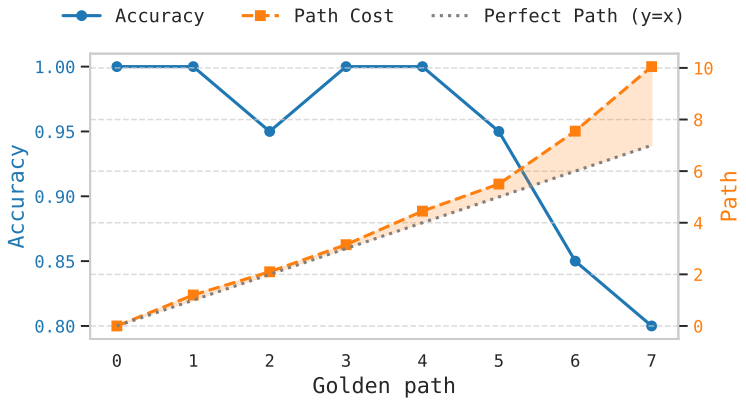


Figure 4.8: Accuracy vs golden path.

expert required only 3.07 minutes, with a reserved memory of 4.006 GB (of which 1.371 GB was allocated for LoRA). This training process is highly efficient and inexpensive, as the primary task of this LLM is generating NSDs. The domain-specific knowledge was injected rapidly, minimizing resource consumption; (ii) The inference cost is measured using two metrics: the generation time required for OSS-GPT to respond to queries and OpenAI's pricing, as GPT-4 is not an open-source LLM. These metrics were evaluated per golden path. As shown in Fig. 4.9, the execution time increases with the golden path. Remarkably, for golden paths up to 7, the execution time remained under one minute, demonstrating very fast planning and execution. Additionally, the pricing increases with the golden path due to multiple LLM requests. However, even for complex requests requiring seven API calls, the cost of $0.175 per request is affordable, especially for large-scale industrial OSS deployments. It is worth noting that OpenAI's pricing is continuously decreasing due to the competition with other closed-source and open-source models. Furthermore, advancements in open-source models, particularly in reasoning capabilities, are narrowing the gap between open and closed-source solutions. This progression makes open-source models a viable alternative for systems like OSS-GPT in the near future.

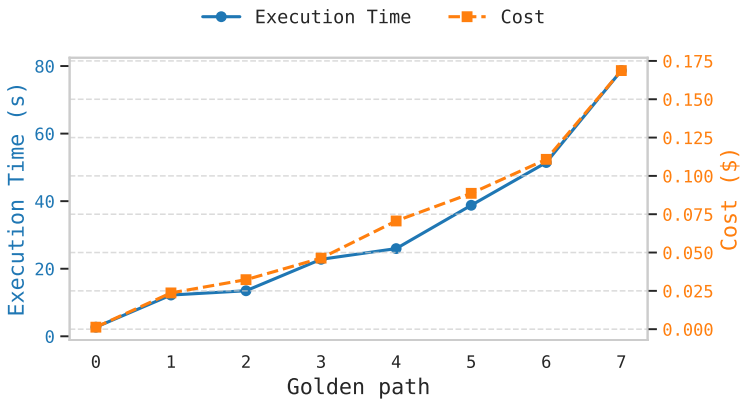


Figure 4.9: OSS-GPT cost assessment.

4.5.3 Discussions

From the results, we can conclude that implementing natural language-based IBN is a challenging but highly useful approach for enhancing user experience and eliminating the complexities associated with traditional standardized structures. Indeed, OSS-GPT demonstrates remarkable performance in simplifying the user experience for OSS users and reducing time spent on learning standards and adapting to new APIs. However, as the experimental setup revealed, EURECOM's implementation of OSS-GPT relies heavily on closed-source LLMs, such as GPT-4, which are less secure compared to local private LLMs. This reliance arises because existing open-source LLMs struggle to consistently generate structured outputs. This capability is critical for OSS-GPT, as each LLM must adhere to a predefined output structure to communicate effectively with other LLM agents. Current advancements in open-source LLMs face challenges in consistently generating accurate structures (structured output), which is crucial in multi-agent LLM systems. In such systems, other agents rely on the LLM's output to proceed with their tasks. This represents a critical area where open-source LLMs lag behind closed-source LLMs, such as OpenAI's GPT. Open-source LLM frameworks should focus on enhancing the accuracy of structured outputs to a 99% level. Achieving this would make multi-agent LLMs based on open-source LLMs a feasible solution. Another important consideration is that OSS-GPT, as presented, is not equipped to handle multiple OSSs. This capability is essential for the future of 6G networking, where tenants or service providers can send intent to multiple OSSs corresponding to different Mobile Network Operators (MNOs). Our future work will focus on addressing this limitation by developing a chatbot that implements natural language-based, intent-driven networking across multiple OSSs that serve multiple MNOs.

4.6 Conclusion

This chapter presented a novel approach to managing 6G networks by leveraging natural language-based, intent-driven management powered by multi-agent LLMs. By enabling natural language interactions, the proposed system simplified user engagement with complex OSS platforms and autonomously adapted to evolving API functionalities. This was achieved through advancements in LLMs, which enabled the understanding of natural language intents and the collaborative, hierarchical planning and execution of API calls. Experimental results at EURECOM demonstrated the effectiveness of this approach in automating 6G network management. Future work can focus on extending OSS-GPT to support multiple OSSs corresponding to different MNOs and on optimizing inference costs by adopting open-source LLMs instead of closed-source alternatives.

At this stage of the thesis, we have presented our contributions to the intent translation step of IBN. In the next part (Part II), we address the intent assurance step, which ensures that an intent continues to satisfy its requirements throughout its lifecycle by performing anomaly detection and subsequently resolving any identified issues. Chapter 5 focuses on anomaly detection, while Chapter 6 addresses both anomaly detection and resolution.

Part II

Generative AI for Intent Assurance

Chapter 5

Anomaly Detection with NWDAF

5.1 Introduction

Anomaly detection plays a key role in an IBN system, serving as the first critical step in the intent assurance stage. In this context, anomalies correspond to deviations from expected network behavior that lead to violations of the intent, typically expressed through SLA constraints. To this end, the network must: (*i*) monitor data from multiple sources (NFs, RAN, infrastructure, etc.); and (*ii*) derive analytics and insights to detect events indicating intent or SLA non-compliance and trigger appropriate actions. For example, by monitoring CPU and RAM usage of NFs alongside user-generated load, the network may predict anomalies caused by insufficient memory resources, potentially resulting in an SLA violation, which can be mitigated by increasing the allocated resources. However, most NFs and RAN vendors provide closed solutions, making data collection difficult or even impossible. To address this limitation, 3GPP introduced the NWDAF in Release 15 and has continuously enhanced it up to Release 17 [114].

Among the key use cases of NWDAF, the UE Abnormal behaviour detection acquired the attention of mobile operators. Indeed, the number of connected devices and the data volume are growing exponentially, making the abnormal detection task very challenging for mobile operators. Despite this, only limited research has been conducted in the area so far. Besides, only few functional 5G CN and NWDAF experimental prototypes are available such as [115]. Until this point, most of the previous works have relied on using 4th Generation (4G) cellular and internet traffic to extrapolate relationships for 5G CNs. In this context, authors of [116] explain abnormal behavior detection in NWDAF according to 3GPP. Furthermore, they extensively review the related work and summarize open problems and provide possible future research directions. However, they did not provide a solution to the problem. The authors of [115] show the integration of Open5GS 5G CN with a NWDAF prototype. However, they only focused on data regarding the NFs interaction ignoring the UEs data. Authors of [117] proposed a functional prototype of NWDAF integrated with Open5GS 5G CN. However, they did not leverage ML to provide advanced use cases.

In this chapter, we design a fully 3GPP-standardized NWDAF based on the microservices architecture that allows new use cases to be added in a plug-and-play fashion. The proposed architecture includes three layers: (*i*) the Exposure layer that implements the 3GPP NWDAF openAPI [118] to provide a fully 3GPP-standardized northbound interface; (*ii*) the Analytics layer that implements the NWDAF intelligence using ML techniques; and (*iii*) the Monitoring layer that uses the NFs service based interface to collect data from NFs, the O-RAN xApps to

collect data from the RAN and the Virtualized Infrastructure Manager (VIM) to collect data from the infrastructure. Then, we integrated our NWDAF with OAI 5G CN[1], mainly with the AMF and the SMF. Moreover, we designed and implemented a traffic anomaly detection algorithm based on LSTM Auto-encoder and plugged the algorithm in the NWDAF architecture as a microservice. We trained the LSTM Auto-encoder using real data from the Milano dataset [20]. Finally, we tested the algorithm at EURECOM 5G facility [3] using Commercial Off-The-Shelf (COTS) UE and real network conditions. The results demonstrate the effectiveness of the proposed algorithm in detecting traffic anomalies, providing a foundation for service-level intent assurance in an IBN framework.

5.2 NWDAF Realization

The section is structured into three subsections, covering different aspects of NWDAF. Subsection 5.2.1 provides an overview of NWDAF's architecture, while subsection 5.2.2 delves into technical details of the NWDAF's implementation. Finally, subsection 5.2.3 explores NWDAF interoperability aspects with OAI 5G CN.

5.2.1 NWDAF Architecture Aspects

The overall architecture of NWDAF is depicted in Fig. 5.1, which comprises three layers: (*i*) Exposure, (*ii*) Monitoring and (*iii*) Analytics. The Exposure layer is the entrypoint of the system, where clients can submit requests following 3GPP specifications. The Monitoring layer collects data from the infrastructure and stores it in a database. The Analytics layer performs computing tasks to derive the requested information.

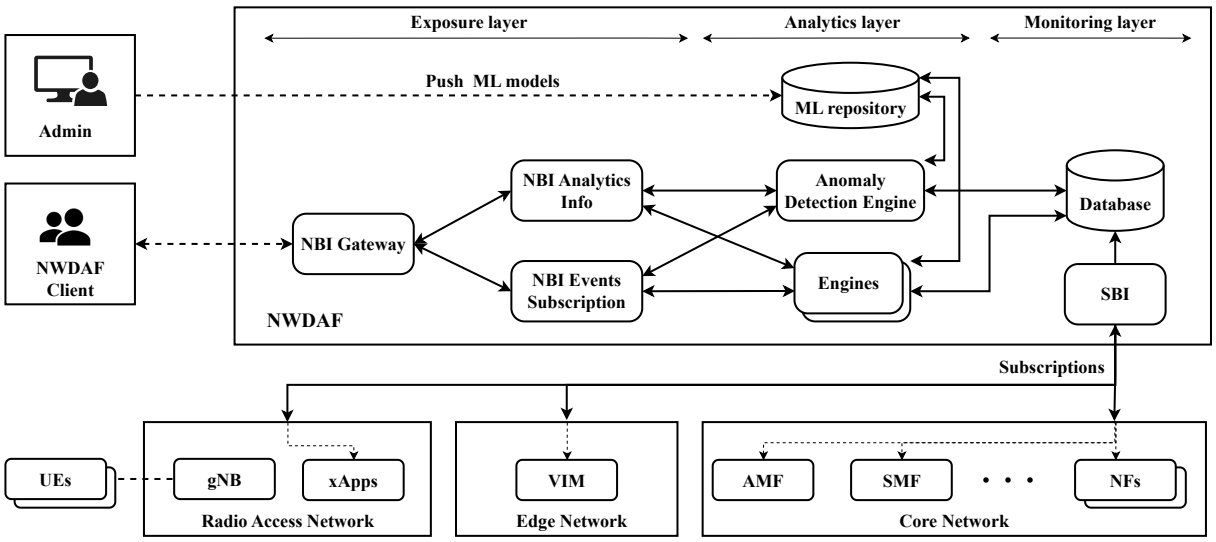


Figure 5.1: Microservices-based NWDAF architecture.

The Exposure layer

The Exposure Layer is responsible for the communication with external entities, such as service providers and NFs. One of its key components is the NBI Gateway module, which acts as a

[1]https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed

3GPP-compliant interface between NWDAF and its clients. The latter can either request information or subscribe for an event (i.e., the client waits till an event occurs to receive information from the NWDAF). The client request is routed to either the NBI Analytics Info module or the NBI Events Subscription module, depending on the nature of the request. On the one hand, the NBI Analytics Info module enables external entities to obtain real-time analytical data from the NWDAF, offering valuable insights into network performance. Its main responsibilities include: (*i*) handling requests from the NBI Gateway module; (*ii*) requesting the Analytics layer to compute the desired information; (*iii*) and generating and transmitting responses to clients based on the Analytics layer's output. On the other hand, the NBI Events Subscription module provides external entities with the ability to subscribe to specific network events and receive notifications when those events occur. This component performs several tasks, including: (*i*) handling requests from the NBI Gateway module; (*ii*) keeping track of existing subscriptions; (*iii*) communicating with the Analytics layer to compute the desired information; (*iv*) and delivering notification messages to the corresponding clients based on the responses received from the Analytics layer.

The Analytics layer

Within the microservices architecture of NWDAF, each service among the NWDAF services, which are summarized in Table 2.1, is mapped to an engine located in the Analytics layer. Each engine performs the required computations to derive the target service information. The Exposure layer is responsible for selecting the appropriate engine based on the type of service requested. For instance, The Anomaly Detection Engine depicted in Figure 5.1 employs an ML-based model to calculate the probability of traffic anomalies based on the history of the UE traffic patterns. The latter are retrieved from the database of the Monitoring layer by the engines. By comparing the current data pattern with the past, the engines can detect potential issues, and network administrators can take preventive measures. The ML models are stored in a ML repository which is populated by the network administrator as shown in Figure 5.1. The engines pull the ML model that corresponds to their proposed service.

The Monitoring layer

The SouthBound Interface (SBI) component is a vital element responsible for collecting and storing network data. It communicates with several entities, such as NFs for CN-related data, VIM for Edge Network-related data, and xApps for RAN-related data. Upon startup of the NWDAF, the SBI module subscribes to the CN NFs to receive notifications and stores the received notification data in the database. Furthermore, it requests VIM to collect the RAM and CPU utilization of various NFs in the Edge network. For more details on the KPIs collection mechanism, readers can refer to [113]. The NWDAF can also request RAN information leveraging O-RAN xApps and Key Performance Metrics (KPM) Service Model (SM) [119].

5.2.2 NWDAF

NWDAF was built using multiple technologies and programming languages. The RESTful API architecture is used for all components due to its scalability, ability to manage multiple data formats, and utilization of HTTP, a widely-used network communication protocol. Python is used for the development of engines that use ML models due to its extensive libraries and frameworks for data analysis and machine learning. GoLang is used for the remaining system components due to its efficient code generation, which is essential for high-performance system

components. This language combination allows greater system flexibility and efficiency, resulting in a more robust architecture. MongoDB is selected as the database due to its scalability and ability to manage unstructured data.

5.2.3 NWDAF Interoperability Aspects

The few experimental NWDAF prototypes are coupled with Open5GS CN[2]. The latter does not expose a standardized Event Exposure API. Thus, we selected OAI 5G CN, which implements the 3GPP-compliant Event Exposure API, to test our NWDAF. OAI offers a 3GPP-complient Release 16 5G CN and RAN that support COTS UEs in real radio conditions. The integration with the CN is done by subscribing to AMF and SMF Event Exposure APIs while the integration with the RAN is done by using FlexRIC [120] as O-RAN RIC. The latter hosts monitoring xApps used by the NWDAF to collect radio information.

5.3 NWDAF Use cases

As mentioned earlier, the NWDAF provides two main services: event subscriptions and analytics information. The NBI Events Subscription module enables clients to subscribe to or unsubscribe from various analytics events, while the NBI Analytics Info module allows clients to request and receive specific types of analytics information. Our NWDAF provides both Core and ML-based services, which we will explore into further.

5.3.1 Core services

Our proposed NWDAF supports three Core services:

- *Network Performance:* The NWDAF provides support for two types of network performance events: "num_of_ue," which measures the number of attach requests during a time window, and "sess_succ_rate," which measures the session success rate during a time window specified in the request. The NWDAF computes "num_of_ue" using the AMF notifications, more specifically the registration event. The NWDAF also supports filtering the number of attach requests according to a specific network area or a specific operator. "sess_succ_rate" is computed using SMF notifications, i.e., the Packet Data Unit (PDU) session establishment event. The NWDAF also supports filtering according to a specific data network name or a network slice.
- *UE communications:* "ue_comm" refer to the number of packets and bytes exchanged in the Uplink (UL) and Downlink (DL) directions for each PDU session. To incorporate these statistics into SMF notifications, OAI-UPF-VPP[3] was used during CN deployment. The SMF collects measurements reports from the UPF using the N4 interface using the usage report procedure of the N4 interface.
- *NF Load:* As mentioned earlier, the NWDAF computes the "nf_load" event data using data received from the VIM component. This data includes information on the CPU and RAM consumption of each NF.

Whenever a client subscribes for an event, the associated engine retrieves the required data from the MongoDB database, and performs computation on the requested time-window i.e., the time interval of the client's interest.

[2]https://open5gs.org
[3]https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-upf-vpp

5.3.2 ML-based services

One of the key services of our NWDAF is ML-based abnormal traffic detection, which is identified, in the 3GPP standard, by the event ID "abnormal_behaviour" and exception ID "unexpected_large_rate_flow".

- *Abnormal Traffic:* the NWDAF clients can subscribe to the "abnormal_behaviour" event to receive periodic updates on the probability of abnormal traffic. The latter is computed by the Anomaly Detection Engine leveraging an LSTM-based Auto-encoder. The LSTM-based Auto-encoder learns the traffic pattern using the history of the UE data measured from the UPF in both UL and DL directions. The traffic pattern contains information such as the data size in bytes at a given timestamp during a week. Fig. 5.2 depicts a basic illustration of the LSTM Auto-encoder model architecture.

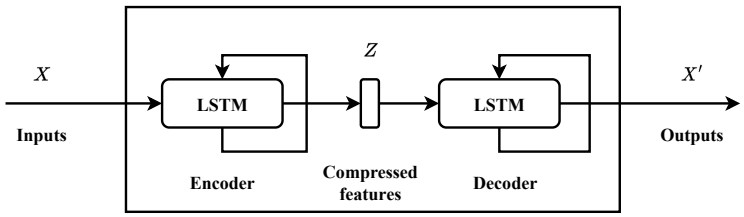


Figure 5.2: LSTM Auto-encoder architecture.

An Auto-encoder is a type of artificial neural network that consists of an encoder and a decoder. During the encoding phase, the input data, represented as $X = \{x_1, x_2, \ldots, x_n\}$, is compressed into a lower-dimensional space according to Equation 5.1.

$$Z = \sigma\left(WX + b\right) \tag{5.1}$$

Where Z is the output of the LSTM encoder, σ is the activation function of the LSTM encoder, W is the weights of the encoder layers, and b is the bias vector of the encoder layers. Similarly, the decoding phase is trained according to Equation 5.2 to obtain the output data $X' = \{x'_1, x'_2, \ldots, x'_n\}$ that is similar to the input dimension.

$$X' = \sigma'\left(W'Z + b'\right) \tag{5.2}$$

Where Z is the input of the LSTM decoder, σ' is the activation function of the LSTM decoder, W' is the weights of the decoder layers, and b is the bias vector of the decoder layers. The motivation behind the Auto-encoder architecture is the fact that it reduces the variance between input and output by training on only normal traffic without considering abnormal traffic. While the motivation behind combining LSTM and the Auto-encoder architecture is due to the nature of the input, which is correlated in time, e.g., traffic during night is different from traffic during the day. Besides, the LSTM is well known for dealing with the vanishing gradient problem [121], which makes the training more stable. In Equation 5.3, σ represents the Mean Absolute Error (MAE) between the input and output of the Auto-encoder.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |x'_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n} \tag{5.3}$$

Let β denote the average MAE over the training data, and β' the MAE of the inference data. Equation 5.4 calculates the traffic anomaly probability p using the difference between β and β'.

$$p = \min(\alpha \times |\beta' - \beta|, 1) \tag{5.4}$$

Where α is a weight to control the impact of the distances scale,i.e., when α is small enough, the distances will have less impact on the probability.

- *ML-based services perspectives:* The suspicion of Distributed Denial Of Service (DDOS) attack service, which is identified by the event ID "abnormal_behaviour" and exception ID "suspicion_of_ddos_attack", can be performed by analyzing the number of users in a similar way to abnormal traffic detection. The Auto-encoder will be trained with normal historical data, and if the current data pattern deviates significantly from past patterns, the likelihood of a suspicion of DDOS attack increases. This probability can be used in a closed-control loop to help the 5G CN mitigate DDOS attacks [122]. Additionally, radio link failures, identified by the "abnormal_behaviour" event ID and "unexpected_radio_link_failures" exception ID, can be predicted using a combination of LSTM and support victor machine, as proposed in [123].

5.4 Performance Evaluation

The section is structured into three subsections: Experimentation setup, which details the experimental setup; Experimentation results, which presents and analyzes the performance of the NWDAF; and Experimentation conclusion, which offers additional insights related to the experiment.

5.4.1 Experimentation Setup

Our experimental setup includes two machines, each equipped with 36 Intel(R) Xeon(R) Gold 6154 CPUs running at 3.00GHz. One of these machines is used to run the gNB based on OAI and is connected to a USRP B210, while the second machine hosts the 5G CN based on OAI and the NWDAF. In addition, we have a laptop with Ubuntu Operating System (OS) connected to a Quectel RM500Q-GL module which is considered as a 5G UE. To train our LSTM Auto-encoder, we leveraged the Milano dataset [20]. This dataset was collected by Telecom Italia for a year. It contains various information regarding user connectivity events. Our experiment focused on the volume of data exchanged with users. We filtered the dataset and organized the records into the following structure: (weekday, hour, minute, and internet data). We included the weekday as traffic varies from day to day, and the hour as traffic during night time differs from that during the day. The internet data metric, introduced by Telecom Italia, is proportional to the traffic volume and is used to conceal the true values. To test the LSTM Auto-encoder, we generate the anomalies by introducing long traffic flows that do not follow the Milano dataset pattern. The LSTM Auto-encoder includes two hidden layers for both the encoder and the decoder. We employ a learning rate of 0.001, batch size of 128, tanh activation function, and train the model using Adam optimizer for 20 epochs. The input sequence size is set to 12. An expert opinion was considered to set the weight α of Equation 5.4 to 0.6, as it produces reasonable anomaly probabilities and a realistic number of anomalies.

5.4.2 Experimentation Results

Fig. 5.3 shows the training and validation loss over the number of epochs. We observe that the loss trend is similar for training and validation sets and converges after approximately ten epochs. Additionally, Fig. 5.4 illustrates the Milano internet data for one week and the generated data using LSTM Auto-encoder. The figure shows that our LSTM Auto-encoder is able to learn the Milano dataset data which is characterized by high data volume during

the day hours and low data volume during the night hours. In addition, the Milano training data with injected anomalies, and anomaly probabilities are plotted in Fig. 5.5. This figure demonstrates that the LSTM Auto-encoder reconstructs input data but deviates when anomalies occur. Further, the distance between input data and generated data is correlated with the anomaly probability. The probabilities increase as the generated data diverge from the input data. The anomaly probability threshold is set to 0.5 to determine whether or not the traffic is an anomaly. Consequently, sequences that deviate from normal behaviour are highlighted in Fig. 5.6. For instance, Internet data values that are unusually high or unusually low are flagged as potential problems. As previously stated, the sequence size is 12 time steps. As a result, points preceding and following the anomalies are highlighted, as they are in the sequence containing the anomaly.

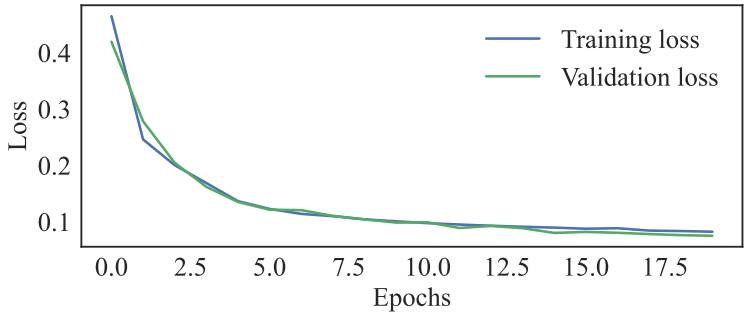


Figure 5.3: Training and validation loss over the number of epochs.

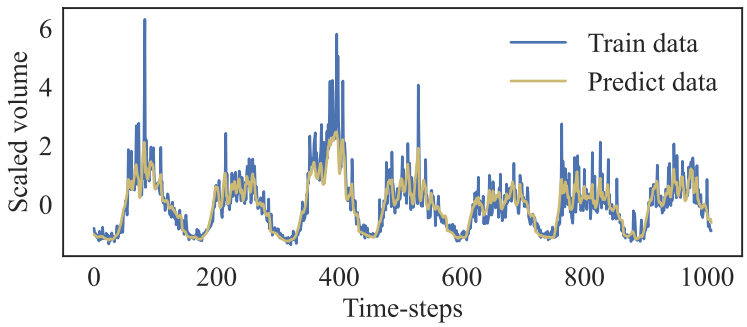


Figure 5.4: A comparison between train volume and generated volume.

5.4.3 Discussions

According to our experimentation, NWDAF can contribute significantly to improving 5G network orchestration. For instance, it can inform clients about the probability of abnormal traffic, which can be used to conduct corrective actions. Then, these clients can use Root Cause Analysis (RCA) tools to identify the underlying causes of network failures. For example, XAI and Machine Reasoning-based techniques can be used in conjunction with expert knowledge to identify the reasons for anomalies using other NWDAF services. In the context of 6G Network's ZSM, this can lead to an AI-based closed-loop fault management that will enable a self-managed network and minimize human intervention. This can also be integrated into NWDAF services by plugging a new microservice Engine for RCA that alerts NWDAF clients of the root cause. In the results, we observed some false positives. While these did not significantly affect the system performance, they could trigger unnecessary corrective actions, impacting operational efficiency.

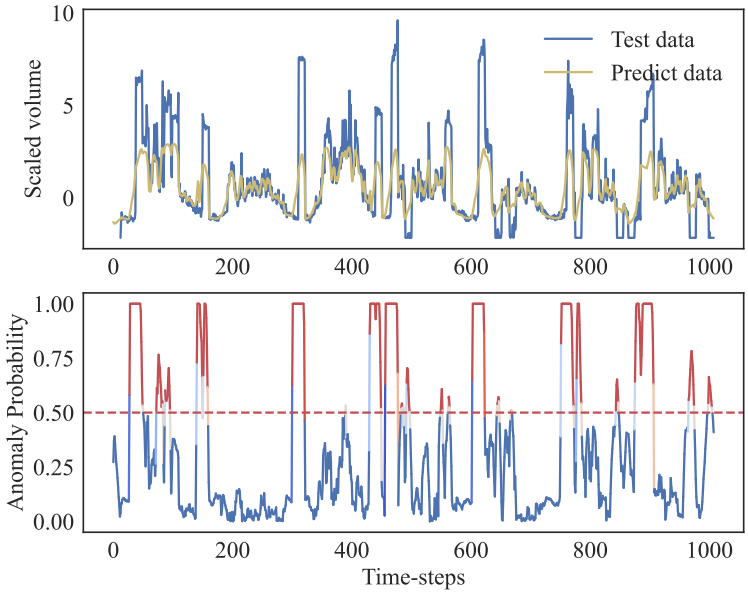


Figure 5.5: Anomaly probabilities for one week.

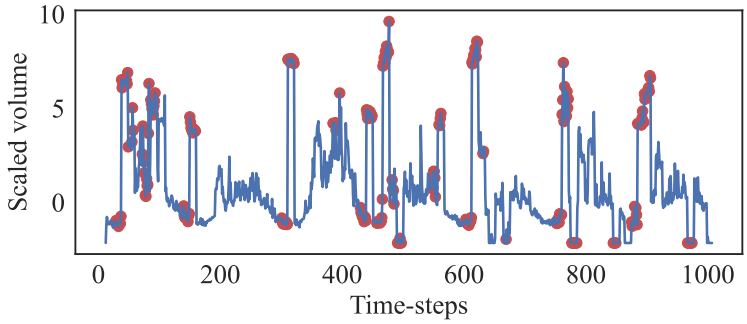


Figure 5.6: The anomaly detection for one week based on the Auto-encoder.

5.5 Conclusion

In this chapter, we proposed a microservices architecture for NWDAF to address the challenge of realizing the various use cases defined by 3GPP. By implementing the 3GPP use cases as microservices, we increased the flexibility and scalability of the NWDAF. We also designed and implemented an LSTM auto-encoder algorithm to detect abnormal traffic events, which was integrated into the NWDAF as a microservice. Experimental results demonstrated the ability of NWDAF to collect data from a real 5G CN and detect abnormal traffic generated by a real UE. Our solution contributed to enabling a self-managed network in the context of 6G's ZSM, reducing human intervention and ultimately improving network performance and end-user experience. Overall, this research advanced the development of NWDAF and its applications in Beyond 5G networks, particularly for abnormal traffic detection.

This chapter addressed the first foundational component of intent assurance (i.e., ZSM), namely anomaly detection. However, intent assurance encompasses both anomaly detection and automated resolution without human intervention. This aspect is explored in the next chapter (Chapter 6), where we integrate LLMs into a pipeline to enable full ZSM functionality.

Chapter 6

Anomaly Mitigation with XAI and LLMs

6.1 Introduction

Intent assurance is a key stage in IBN, requiring the continuous detection and resolution of anomalies throughout the intent lifecycle, with minimal or no human intervention, to ensure that intents are satisfied. In practice, during the assurance phase, intents are implicitly realized through service-level objectives and SLA constraints, whose violations indicate intent non-fulfillment. To address this challenge, ETSI established the ZSM group in December 2017. According to [124], ZSM-enabled NMSs prioritize the detection and prediction of network and service anomalies, enabling intelligent, autonomous, and self-healing management systems. This paradigm is particularly relevant in 6G networks, where stringent SLA requirements are expected for cloud-native services. A representative example involves detecting SLA violations, such as excessive latency or reduced throughput in cloud applications, and subsequently identifying their root causes. For instance, the system may determine that insufficient CPU or RAM allocation is responsible for the degradation and autonomously adjust these resources to restore service performance, thereby maintaining intent satisfaction without human intervention.

The process of ZSM anomaly resolution involves three steps: *(i)* detecting anomalies, *(ii)* extracting the root cause of anomalies, and *(iii)* resolving the anomalies based on the detected root cause. To address these steps, the research community heavily relies on advanced AI methods for anomaly detection [125]. However, ZSM-enabled ZSM requires not only detecting but also resolving these anomalies, which necessitates understanding their root causes. This becomes challenging with opaque AI models that lack explainability. For example, using deep learning-based anomaly detection modules provides predictions without insights into how these neural networks arrived at these predictions. To address this limitation, XAI solutions have emerged in research, offering insights into the root causes of anomalies [22]. These models provide information about which features contributed the most to the predictions, thereby revealing the root cause of the anomaly. For instance, if the CPU feature contributed the most to the latency SLA degradation, it indicates insufficient CPU resources for the given application. In this context, *Mekki et al.* in [23] used XAI to explain AI predictions and dynamically scale CPU and RAM resources of a microservice application based on a simple heuristic that leverages XAI values to enable ZSM.

However, XAI often presents explanations in numerical values that may be challenging for

users with limited domain knowledge to interpret, impacting the trustworthiness of ZSM systems. User-friendly explanations, on the other hand, aim to generate human-understandable explanations, ensuring they are understandable to diverse users and increasing trust in ZSM's autonomous decisions. Fortunately, with rapid advancements in GenAI, LLMs offer a promising solution. These models can generate human-like text in various human languages [2], making them an attractive choice. Moreover, LLMs excel in reasoning tasks without requiring additional training [2], which can be leveraged to resolve anomalies, enabling trustworthy ZSM autonomously. Thus, LLMs should be explored for decision-making problems in 6G networks (e.g., resolving anomalies).

While [23] proposed a closed-loop ZSM approach to detect and resolve cloud resource–related anomalies, their method relies on heuristics for anomaly resolution, which do not optimize resource allocation efficiently, as shown in Fig. 6.4. Moreover, it is not easily interpretable for users with limited domain knowledge, since the XAI outputs remain largely numerical and difficult to understand. To address these challenges, we extend [23] by proposing a trustworthy ZSM-enabled NMS design based on a novel anomaly detection and resolution pipeline: AI|XAI|LLM. Furthermore, we present a pipeline's use case wherein the NMS dynamically scales cloud resources (e.g., CPU and RAM) for a microservice application with specific SLA latency requirements. To achieve this, we employed: *(i)* XGBoost [24] as the AI model to predict SLA latency violations; *(ii)* SHAP [25] as the XAI model to provide numerical explanations of these anomalies, revealing their root causes; and *(iii)* Llama2 [26] as the LLM to generate human-friendly explanations using natural language. Subsequently, we consider two scenarios: if ZSM is enabled, Llama2 autonomously resolves the anomaly; otherwise, it offers recommendations on how to resolve the anomaly, requiring user intervention. The main contributions of this chapter are as follows:

- *ZSM-Enabled NMS Anomaly Detection and Resolution Pipeline:* We propose a novel anomaly detection and resolution pipeline to effectively enable trustworthy ZSM in NMSs by leveraging the latest advancements in AI. This pipeline integrates AI|XAI|LLM.
- *Dynamic Scaling Use Case with ZSM:* We implemented the proposed pipeline on a real-world 6G NMS use case to dynamically scale cloud resources for a microservice application within 6G networks. This implementation employed XGBoost [24] for AI, SHAP [25] for XAI, and Llama2 [26] for LLM.
- *Real-World Deployment and Testing:* The dynamic scaling use case pipeline was deployed on an edge computing cluster managed by Kubernetes[1]. Llama2 was deployed using a single NVIDIA A100 GPU with 40GB of vRAM. Comprehensive real-world tests, including multiple LLM evaluations, were conducted to optimize performance for the specific dynamic scaling use case.

6.2 Related Works

AI-based methods have been widely used for anomaly detection in networking. For example, in [126], the authors proposed an anomaly detection approach using the XGBoost classification algorithm to enhance network security by accurately identifying and classifying traffic anomalies. XGBoost [24] is a powerful AI technique that constructs an ensemble of decision trees to

[1]https://kubernetes.io

achieve high accuracy in classification tasks. However, a major limitation of these AI methods is their lack of interpretability. These models are often considered black boxes because they provide predictions without transparent insights into their decision-making process, making it difficult to understand the underlying causes of anomalies. As a result, XAI methods have been employed to provide interpretable explanations for the predictions made by these AI models, enabling the extraction of the root causes of anomalies for resolution.

Recent developments have extended the focus of XAI towards LLMs, wherein multiple strategies are employed to integrate XAI with LLMs. These strategies are explained in [127]. The 9th strategy in this latter concerns using LLMs to generate user-friendly explanations for XAI outputs. Our work builds upon this strategy by further exploiting LLMs' reasoning capabilities for anomaly resolution [2]. Additionally, some research works utilize LLMs for anomaly detection and explanation [128]. This is efficient, but detecting anomalies with an LLM requires triggering the LLM more frequently, which results in significant energy consumption. Our approach requests the LLM only when anomalies are detected. This is enabled by employing the AI-based anomaly prediction models before triggering the LLM, thus reducing the frequency of LLM calls, which effectively mitigates the energy consumption problem.

6.3 System Design

In this section, we present the high-level architecture of the ZSM framework and illustrate its practical application through a use case involving the scaling of cloud resources (i.e., CPU and RAM) for 6G microservices.

6.3.1 High-level architecture

The ZSM framework features a closed-control loop for managing 6G services, as illustrated in Fig. 6.1. It consists of three layers: *(i) 6G Infrastructure,* this layer includes cloud/edge clusters and radio units that support 6G services. It encompasses components like the 6G CN, RAN cloud solutions, and wireless base stations; *(ii) NMS Anomaly Detection and Resolution,* this module autonomously detects and resolves anomalies in the infrastructure, such as issues in base stations or cloud clusters. It operates above the infrastructure layer; *(iii) User Plane,* this top layer involves users deploying 6G services and interacting with the NMS.For instance, users deploying XR services may require a core network and a set of edge applications with specific SLA latency and throughput to support XR requests. These users also receive human-like explanations from the NMS regarding detected anomalies, recommendations to resolve them, or actions taken by the NMS if ZSM is enabled.

As shown in Fig. 6.1, the design of the NMS consists of three stages: *(i) Monitoring System (MS),* the MS collects KPIs from the 6G infrastructure, including network latency, packet loss, and resource usage metrics. These KPIs are essential for the Anomaly Detection Engines (ADEs) to identify anomalies; *(ii) ADEs,* the ADEs use AI to detect anomalies and XAI to provide numerical explanations. Each ADE focuses on specific anomalies, such as QoS issues or security threats, and uses XAI to explain feature contributions; *(iii) Analytics Engine (AE),* the AE processes outputs from the ADEs, using an LLM to reason about predictions and explanations. It provides comprehensive explanations for root causes and suggests actions. If ZSM is enabled, the LLM can autonomously execute these actions without human intervention.

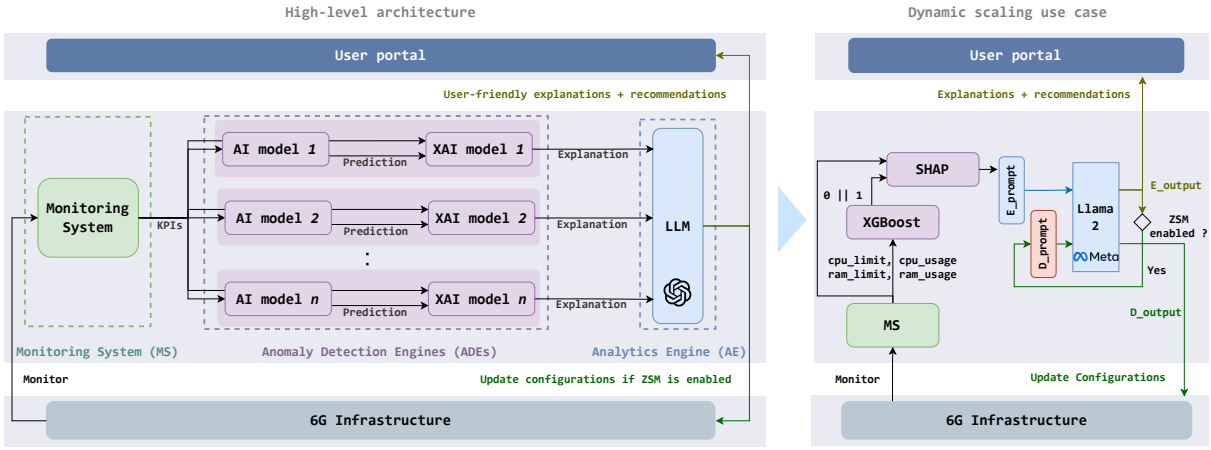


Figure 6.1: LLM-enabled trustworthy ZSM architecture design and use case.

6.3.2 Dynamic scaling use case

This use case ensures that CPU and RAM resources for a given microservice application are dynamically adjusted to prevent SLA latency violations. For this purpose, we employed XGBoost as the AI model, SHAP for XAI, and Llama2 as the LLM. Scaling up is performed using the pipeline as illustrated in Fig. 6.1, whereas scaling down is implemented using a simple heuristic. Below, we describe the scaling up process, and for more information about scaling down, readers can refer to [23].

XGBoost as AI

As depicted in the right-side of Fig. 6.1, XGBoost receives cloud monitoring KPIs from the MS, specifically CPU and RAM-related information. Based on this data, it predicts whether the application will violate the SLA latency. To achieve this, we trained it using the dataset of a microservice application presented by *Mekki et al.* [129]. First, we labeled the dataset's lines as SLA latency violated (y=0) or not (y=1) when the response time is lower or higher than a given threshold. In our case, and based on the experimental data from the dataset, we determine the threshold by measuring the application's latency when it is allocated more resources than it requires. This value represents the optimal response time that the application can achieve. Then, training is performed using CPU and RAM limits and usages as inputs, with the SLA violation label as the output.

SHAP as XAI

The XAI module of the ADE relies on a local explanation method based on SHAP to calculate feature importance scores that influence XGBoost's output. Negative values indicate that a feature drives the model's output towards 0, while positive values indicate that a feature drives the output towards 1. For example, if the CPU usage has a score of -2.56, it means that the CPU usage value pushed the model towards output 0 (SLA not respected) with a score of 2.56. Conversely, if the RAM limit has a score of +0.99, this indicates that the feature pushed the model towards output 1 (SLA respected).

LLama2 as LLM

Llama2 is employed as the AE. This model is trained using in-context learning, where knowledge is injected into the prompts. From the right-side of Fig. 6.1, we can see two prompts corresponding to two different LLM tasks. *E_prompt* contains information on how SHAP operates, alongside the context and values of XGBoost predictions and SHAP results. The objective here is to prompt Llama2 to employ deductive reasoning to interpret these values and provide a human-readable explanation to the user, along with recommendations on how to mitigate the SLA violation by updating CPU or RAM resources, or both (*E_output*). Deductive reasoning is used here because it involves applying general rules to specific instances to draw conclusions, i.e., given SHAP values, Llama2 can apply implicit rules to deduce the root causes and suggest appropriate mitigation strategies. The reasoning can be expressed through the following deductive rules: (i) *XAI values* $\Rightarrow$ *deduce the root cause*; and *(ii)* *x is the root cause* $\Rightarrow$ *propose increasing the limits of x.* For example, if the SHAP values show that CPU usage is a significant factor in SLA violations, the model can deduce that increasing CPU resources is necessary. Secondly, if ZSM is enabled, *D_prompt* provides the *E_output* with some instructions to Llama2 to perform Named-Entity Recognition (NER) and extract the values in a JSON format acceptable by the infrastructure (*D_output*). We opted for task decomposition into two tasks (i.e., first reasoning to generate *E_output*, then NER to extract *D_output*) because combining them resulted in more generation errors. LLMs excel at performing single tasks; therefore, task decomposition is essential.

6.4 Performance Evaluation

In this section, we first describe the experimentation setup. Following that, we outline three evaluation steps: *(i)* one for the entire framework, where we present the scenario of the dynamic scaling use case and the pipeline's results; *(ii)* the second for evaluating LLMs' reasoning and decision-making capabilities; and *(iii)* the third for assessing the trustworthiness of *E_output*. In all evaluations, we enable ZSM, allowing us to extract the new configuration (i.e., CPU and RAM allocation) from the LLM's *E_output* and apply it directly without human intervention. Finally, we discuss the evaluation results.

6.4.1 Experimentation Setup

Our experimental setup, illustrated in Fig. 6.2, includes two machines hosting the Kubernetes-based cluster and the Llama2 LLM, respectively. The first machine, equipped with an Intel(R) Xeon(R) Silver 4314 CPU (2.40GHz), runs Ubuntu OS 22.04.3 LTS and Kubernetes v1.28.3 to manage a single-node cluster hosting the MS and ADE components. The second machine, with an Intel(R) Xeon(R) Gold 6240R CPU (2.40GHz) and an NVIDIA A100 GPU (40GB vRAM), also runs Ubuntu OS 22.04.3 LTS and NVIDIA CUDA driver version 545.23.06. It uses Docker[2] to run the Llama2 LLM[3] (70 billion parameters) facilitated by textgen-webui[4]. The LangChain[5] framework handles LLM prompt preparation, with parameters set to a maximum of 2000 tokens, a temperature of 0.1, and a repetition penalty of 1.15. These parameters are flexible and can be adjusted as needed.

[2]https://www.docker.com/
[3]TheBloke/Upstage-Llama-2-70B-instruct-v2-GPTQ
[4]https://github.com/oobabooga/text-generation-webui
[5]https://www.langchain.com/

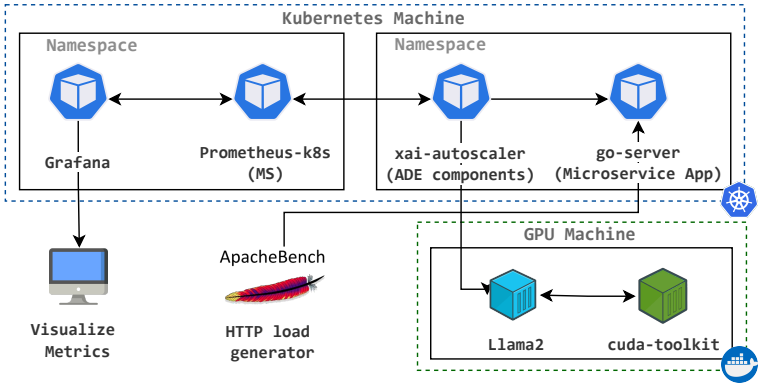


Figure 6.2: Evaluation setup.

6.4.2 Experimentation Results - Dynamic scaling framework

Scenario

We initially configured the microservice application with 0.25 CPU cores and 256 MB of RAM. To evaluate performance under load, we used ApacheBench[6] to generate high volumes of concurrent HTTP requests. The stress tests involved executing a predefined pattern of requests five times, with varying rounds of concurrent clients and total requests: 15 rounds with 50 clients and 200 requests, 10 rounds with 100 clients and 400 requests, 15 rounds with 300 clients and 500 requests, and 20 rounds with 50 clients and 50 requests. This pattern is illustrated in Fig. 6.2, with c representing concurrent clients and n the total requests. Throughout the tests, we monitored CPU and RAM allocation and utilization using Prometheus[7], as well as the LLM's generated *E_output* and *D_output*. For performance comparison, we executed two additional approaches: the state-of-the-art scaling method based on XAI and heuristic decision-making from [23] (denoted *"xai-heur"*), and a basic heuristic approach without the XAI module (denoted *"no-xai"*), also from [23]. The key distinction is that our approach (*"xai-llm"*) allows flexible resource allocation, while the heuristic approaches use fixed values predetermined in the program. Additionally, *"no-xai"* does not use an XAI module to provide the root cause, thus updating all resources (both CPU and RAM) when an SLA violation is predicted.

Results

From Fig. 6.3, it is evident that CPU and RAM allocation adapt dynamically to the application load. For instance, the RAM limit increases with usage during heavier loads, and the CPU behaves similarly. The XGBoost|SHAP|LLama2 pipeline efficiently manages this dynamic scaling. Additionally, before updating resources, LLama2 generates explainable, human-like *E_output*. Two examples are shown at the top of the figure at times t_1 and t_2: one for updating RAM, and the other for updating both CPU and RAM. This notification is sent to users to enhance trust in the pipeline's decisions. The system then extracts the new configuration from this text and generates *D_output* in a JSON structure, enabling efficient resource updates. However, minimal errors in CPU allocation occurred (e.g., at t_3). Therefore, we evaluate LLama2's role in this pipeline in the next subsection. For more information on XGBoost and SHAP, readers can refer to [23].

[6]https://httpd.apache.org/
[7]https://prometheus.io

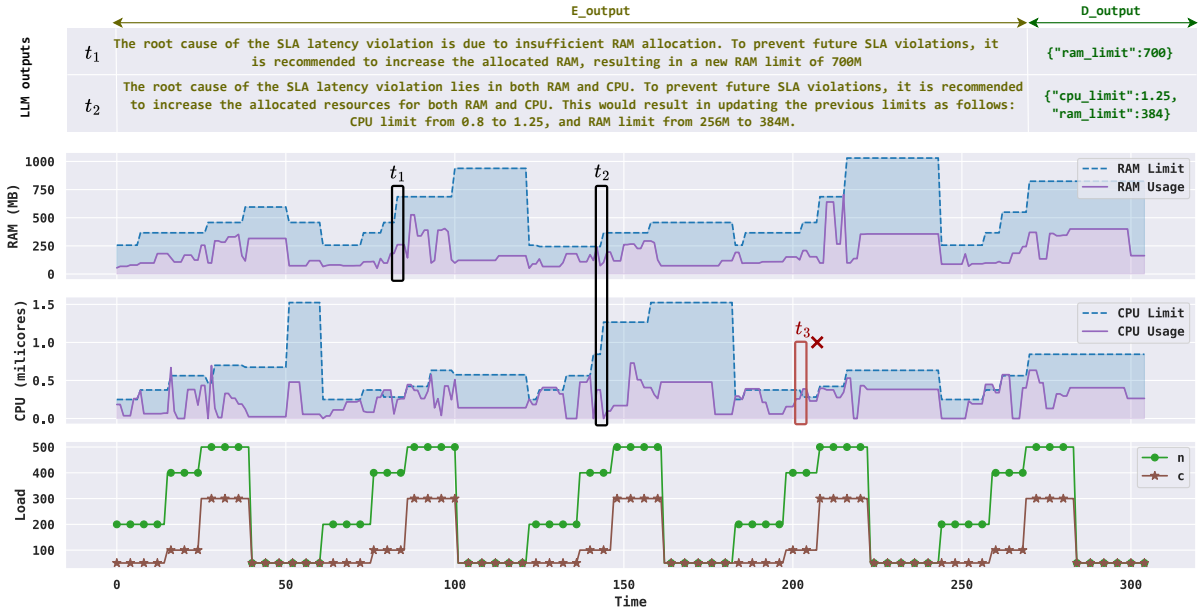


Figure 6.3: Dynamic CPU and RAM scaling in response to microservice load.

In Fig. 6.4, we compared our approach (*"xai-llm"*) with *"xai-heur"* and *"no-xai"* by calculating the mean CPU and RAM allocations for each (n, c) pair. We found that all three approaches yielded similar latencies but with different resource allocations. The *"no-xai"* method generally allocates more CPU and RAM due to the absence of the XAI module, which results in less efficient resource use. In contrast, our approach prioritizes RAM during dense requests, possibly because the LLM, trained to understand the impact of insufficient RAM on system stability, allocates more to prevent failures. For CPU allocation, our approach generally uses less, except in dense requests where it allocated more than *"xai-heur"*. Averaging these allocations across all (n, c) values, *"xai-llm"* ranks first in CPU allocation, followed by *"xai-heur"*, and then *"no-xai"*. Conversely, *"xai-heur"* ranks first in RAM allocation, followed by *"xai-llm"*, and then *"no-xai"*. Overall, these results demonstrate that the LLM can effectively serve as a decision-maker, competing with state-of-the-art scaling methods while offering user explanations.

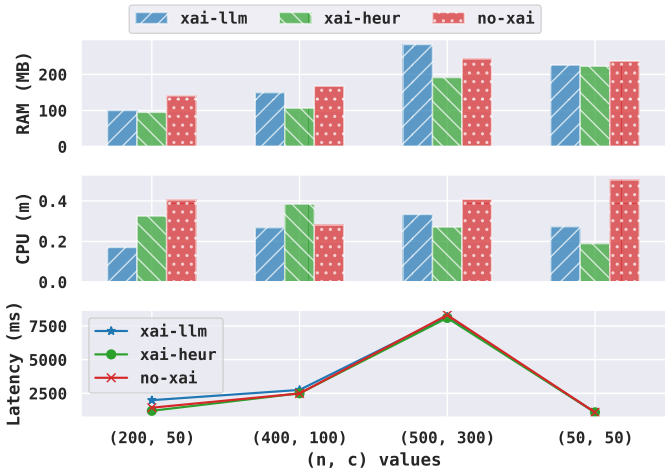


Figure 6.4: CPU and RAM allocation comparison.

6.4.3 Experimentation Results - LLM reasoning & decision-making

Scenario

In this scenario, we focused exclusively on cases where anomalies were predicted by XGBoost. We generated 100 random allocations for CPU and RAM, with values ranging between 0.25 and 2 cores for CPU, and between 128 MB and 2 GB for RAM. Alongside these allocations, we randomly assigned SHAP values between -5 and 5 to each feature. We evaluated the performance of various well-known open-source LLMs and OpenAI's GPT-4 on generating *E_output* and *D_output*. The LLMs were executed on all the generated samples for evaluation. During this process, we calculated the *Score* of each LLM on this test: +1 if the generated JSON contained the correct new configuration, which entails: *(i)* generating a correct JSON structure; *(ii)* including a feature in the JSON if its SHAP value is negative; and *(iii)* ensuring that the JSON value is greater than the randomly assigned value. A *Score* of +0 was assigned if these conditions were not met. Additionally, we categorized errors made by the LLMs as follows: *(i) JSON errors*, indicating failure to generate a correct JSON structure; *(ii) Amount errors*, where the new CPU or RAM limit was lower than the initial CPU or RAM, which should have been higher; and *(iii) Features miss*, where a feature with a negative SHAP value was missing from *D_output*. We also monitored the LLMs' execution times: *Explanation time* and *Update time* for *E_output* and *D_output* generation, respectively.

Results

Fig. 6.5 presents the scores and mean generation times for various LLMs on given tasks, highlighting both *E_output* and *D_output*. OpenAI's GPT-4 achieved the highest score (91/100). Among open-source LLMs, Llama2 70B scored 89/100 with 1 *JSON error*, 4 *Amount errors*, and 6 *Features miss*, making it the best open-source LLM for these tasks. Its execution time is 5.9 seconds for *E_output* and 2.6 seconds for *D_output*, resulting in an E2E time of 8.5 seconds, longer than most LLMs due to its size. CodeLlama 13B was the second-best with a score of 73/100 and an E2E time of 5 seconds. Vicuna 30B scored 57/100 but had an E2E time of 17 seconds. Phi2, despite being smaller (2B parameters), scored 41/100. Google's Gemma 7B scored 0/100 with 100 *JSON errors*, failing to generate JSON structures properly. Overall, Llama2 70B scored highest among open-source models, but its size and E2E time are significant. If these values are critical for other decision-making systems, CodeLlama 13B offers a good balance with a score of 73/100, 13B parameters, and an E2E time of 5 seconds.

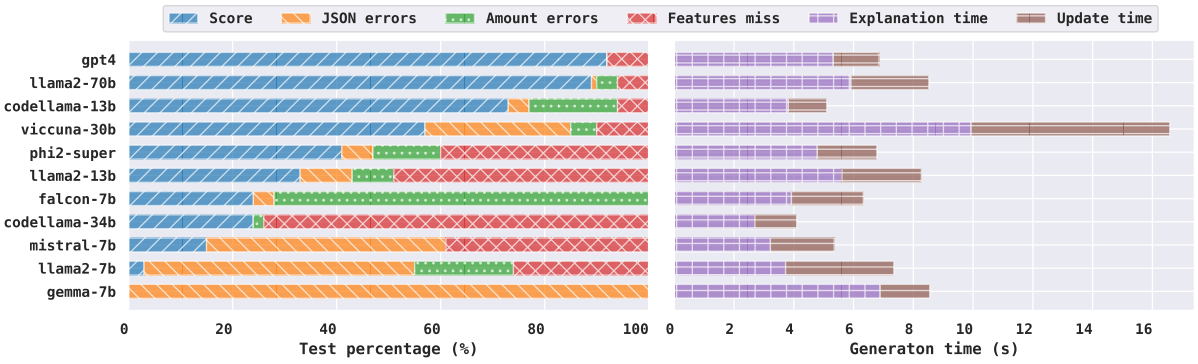


Figure 6.5: LLMs scores and *E_output*-*D_output* generation times for the dynamic scaling use case.

6.4.4 Experimentation Results - LLM's trustworthiness

Scenario

In this scenario, we evaluated the semantic quality of the *E_output* generated by each LLM to assess the comprehensibility of the texts and the trust users are likely to place in them. We used several established metrics for this evaluation: *(i) BLEU,* assesses n-gram precision between generated and reference texts, with higher scores indicating better fluency and adequacy; *(ii) METEOR,* measures quality considering exact word matches and semantic similarity using stemming and synonymy, providing a comprehensive evaluation of fluency and semantic fidelity; *(iii) ROUGE-L,* focuses on the longest common subsequence of words between generated and reference texts, reflecting content overlap and sequence similarity; and *(iv) BERTScore,* evaluates token-level semantic similarity using contextualized embeddings from BERT. Reference texts for these metrics were generated by 10 experts, representing the ideal *E_output*. High scores on these metrics indicate that an LLM produced well-explained and coherent text.

Results

Fig. 6.6 illustrates the aforementioned metrics applied to each LLM and sorted from highest to lowest scores. Among the evaluated models, Llama2 with 70B parameters achieves the highest metrics scores, indicating that it consistently produces text that is semantically similar to the expectations of domain experts. Therefore, we can confidently conclude that the *E_output* from Llama2 is trustworthy, enabling reliable ZSM. Surprisingly, Phi2 ranks second in the metrics ranking, suggesting that it also generates high-quality text closely aligned with expert expectations. While the *BERTScore* across most LLMs is approximately 0.7, indicating a high degree of semantic similarity with expert texts, the variations in wording choices are evident as shown by *BLEU*, *METEOR*, and *ROUGE-L* metrics. However, overall, nearly all LLMs generate text that closely matches in meaning (high *BERTScore*), albeit with differences in wording. Notably, Llama2 70B stands out as the model-producing text that aligns most closely in wording with user expectations.

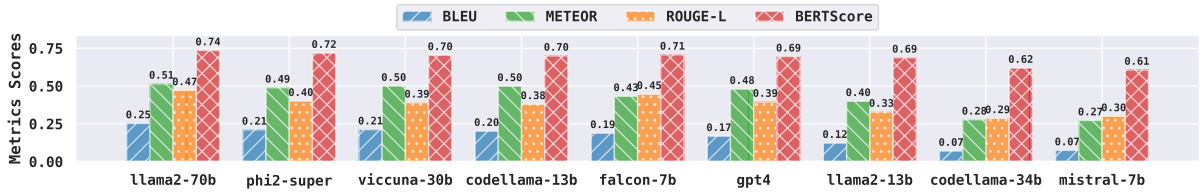


Figure 6.6: LLMs metrics score.

6.4.5 Discussions

We divided the performance evaluation section into three steps. Fig.6.3 demonstrates a real-world execution of the pipeline, showcasing its efficiency in predicting SLA violations and providing timely updates with human-readable output. This illustrates the potential for LLM-based trustworthy ZSM in 6G networks. Given the minimal errors made by the LLMs, we further evaluated the decisions made by these models as depicted in Fig.6.5. Our analysis identified Llama2 with 70B parameters as the best choice among open-source LLMs, particularly when execution time is not overly constrained. The explanation generation time is 5.9 seconds, allowing for comprehensive and explainable text generation, while the update time is approximately 2.3s when ZSM is enabled, much quicker than the time it would take for a user to read the *E_output*

and manually apply the commands. Additionally, Fig. 6.6 showcases the comprehensibility of the generated text using well-known metrics, providing insights into the trustworthiness of these texts. Llama2 ranks highest, indicating it produced text closest to expert expectations. From these explorations, we conclude that LLMs in ZSM can enable trust and powerful decision-making in 6G networks.

6.5 Conclusion

In this chapter, we presented a comprehensive framework for achieving trustworthy ZSM in 6G networks. Our approach integrates AI for anomaly detection, XAI for root cause analysis, and LLMs for generating user-friendly explanations and implementing corrective actions. We tackled a use case of this global architecture, which is an application deployed at the edge with some latency requirements. We relied on XGBoost to detect anomalies related to latency violations, SHAP to extract the numerical root cause, and Llama2 to explain the root cause and correct the anomaly. The performance evaluation demonstrated the pipeline's effectiveness in predicting and addressing SLA violations, providing timely updates and human-readable explanations that enhance user trust.

At this stage of the thesis, we have seen how GenAI can shape both next-generation intent translation and assurance in IBN systems. However, alongside leveraging these GenAI approaches, it is also essential to manage their operations effectively. Key considerations include creating GenAI models specifically tailored for IBN tasks and optimizing their inference when handling a dense set of tasks. These challenges are explored in the next part (Part III), where Chapter 7 focuses on building Telecom-aware LLMs, while Chapter 8 addresses optimizing their inference.

Part III

Generative AI Operations in IBN

Chapter 7

Building Telecom-aware LLMs

7.1 Introduction

LLMs have revolutionized numerous fields with their remarkable ability to understand and generate human-like text [78]. These models require extensive training on vast datasets, such as OpenWebText [130], Common Crawl [131], and Dolma [132], to capture and learn the nuances of human language from diverse linguistic contexts. By leveraging such comprehensive data, LLMs can mimic human language patterns and understand intricacies across different domains and dialects [133], making them indispensable tools for natural language understanding, generation, and a variety of downstream tasks. These tasks include sentiment analysis, machine translation, and summarization [78], which require not only a deep understanding of language but also the ability to infer and reason from context. In addition to commercial models like OpenAI's GPT series, including GPT-3 and GPT-4, there are notable open-source LLMs available, such as Meta's Llama series [132], Google's T5 series [134], and Gemma series [135]. These models provide the research community and industry with powerful tools to build upon and integrate into a wide array of applications, thus democratizing access to cutting-edge NLP technologies [78].

As the demand for beyond 5G/6G network services continues to grow in today's digital world, networks must be updated to meet their evolving requirements. XR and VR applications, for example, require critical QoS in terms of latency and throughput [5]. To address these needs, 5G networks need upgrades with new functionalities. Multiple standardization bodies are actively working to establish guidelines for implementing 5G. The 3GPP is making significant efforts to propose TSs for developers and technical personnel to aid in the development of 5G network procedures [27]. However, these TSs are often complex and voluminous, making it challenging for developers and readers to find the necessary information. In this context, chatbot assistants become essential in answering questions about the TSs' content. With the rapid advancement in GenAI, LLMs offer a promising solution. These models can learn from vast datasets and generate human-like text [78], assisting users in navigating and understanding the extensive 3GPP TSs documents related to 5G networks.

As Telecom-aware LLMs offer a promising approach in enhancing the efficiency of information provision and development of beyond 5G/6G networks (and other domains given their standards), creating them is a complex task [136]. Training existing pre-trained LLMs in the new Telecom domain is not straightforward, as this training necessitates a high-quality dataset specifically tailored for the given domain [109]. This involves collecting data from 3GPP TSs,

cleaning, and generating a structured dataset that is training-optimized (or training-ready), i.e., ready for the LLM to train on. Among these steps, the cleaning process is particularly complex due to challenges in handling figures and paragraphs referencing other paragraphs within the TSs (and from other TS documents). Additionally, data generation presents complexities related to post-processing the cleaned TSs as LLMs require a specific data structure to respond to users' Telecom-related queries effectively. Therefore, developing an effective Telecom-aware LLM requires a pipeline to generate these specialized datasets, ensuring that future LLMs can bridge this critical gap in Telecom understanding and application [137].

To this end, this chapter addresses the aforementioned challenges by proposing a data pipeline designed for dataset generation specifically to train (i.e., fine-tune) LLMs for understanding 5G technologies. The pipeline is capable of collecting and cleaning a set of 3GPP TSs and performing structured data generation using existing state-of-the-art LLMs. The resulting dataset is then used to train open-source LLMs to create new 5G-aware LLMs capable of understanding and generating 5G-related text. This approach can be extended to other 3GPP standards to develop 6G-aware LLMs as the 6G TSs are expected to be released in the near future. The major contributions of the chapter are as follows:

- We detail a robust framework for gathering, processing, and cleansing 3GPP TSs. This framework not only simplifies the transformation of highly technical, unstructured, and comprehensive documents into a clean format but also ensures the retention of critical information, making it conducive for LLM training.
- Within the aforementioned framework, we leverage state-of-the-art LLMs to generate prompt/completion pairs. This approach relies on the advancements of powerful LLMs to create 5G-related datasets for training other LLMs, with the goal of developing 5G-aware LLMs.
- As a result of the pipeline, we present an open-source dataset, namely the OAI Instruct dataset[1]. This dataset is uniquely tailored to enhance the comprehension and generation capabilities of LLMs concerning a subset of 22 3GPP TSs.
- To evaluate the pipeline's effectiveness, we perform a specific type of fine-tuning called freeze-tuning to adjust the parameters of open-source LLMs based on our previously crafted LLM, aiming to create 5G-aware LLMs and demonstrate the utility of our dataset.
- We thoroughly evaluate GPT-4[2], open-source LLMs, and fine-tuned (5G-aware) LLMs using the evaluation segment of our newly created dataset. This assessment critically analyzes the enhanced performance of these models on specialized tasks, highlighting the effectiveness of our dataset for both training and evaluation methodologies.

7.2 Related Works

Advanced LLM training techniques are continuously developed to enhance the outputs of LLMs. Among these, Instruct-type fine-tuning represents a significant evolution in the field of AI. The creation and utilization of instruct datasets are pivotal for developing effective domain-specific LLMs, which are essential for achieving high performance in specialized contexts. Instruct

[1] https://huggingface.co/datasets/Netsoft/oai-instruct
[2] gpt-4-0125-preview

datasets, by providing the necessary contextual examples, enable LLMs to comprehend and generate domain-specific content accurately. For example, models like T5, which operates within a unified text-to-text framework, highlight the success of using instruct datasets to attain state-of-the-art performance across various NLP tasks [138]. Moreover, GPT-3's fine-tuning process with instruct datasets has proven its enhanced capabilities in handling tasks that demand deep domain knowledge [139]. This trend underscores the critical role of well-constructed instruct datasets that not only embed detailed domain-specific knowledge but also support the linguistic capabilities required for broad NLP applications. Nowadays, open source LLM developers consistently fine-tune their foundational models on instruct datasets like the open source LLMs Meta-Llama-3-8B[3] and Meta-Llama-3-8B-Instruct[4], which diverge across the LLaMA family's various versions and parameter sizes, reflecting a standard practice in the field.

Research on data generation pipelines provides valuable methodologies for creating high-quality datasets. For instance, researchers in [140] explore techniques to enhance dataset diversity and quality, such as translation data augmentation. The latter augments training data by altering existing sentences in a parallel corpus to diversify training examples while preserving semantic equivalence. It involves selecting targeted rare words for substitution to provide new contexts for these words, using language models to suggest plausible substitutions, and ensuring that the substitutions and their translations maintain grammatical and semantic coherence. Similarly, the authors of [141] illustrate methodologies for constructing datasets that improve model performance, such as synthetic data generation and multi-task learning. Recently, with the rapid explosion in GenAI, LLMs are being widely used as a key component in dataset generation pipelines [142]. For example, tools like DataDreamer [143] simplify the creation of synthetic datasets and facilitate reproducible LLM workflows, addressing challenges such as model brittleness and reproducibility. Additionally, the open source model Bonito, introduced in [144], complements these capabilities by transforming regular text into specialized training exercises for language models. Bonito utilizes meta-templates from datasets like P3[5] to craft synthetic tasks for various domains, enhancing the diversity and applicability of training materials. However, both tools exhibit limitations in handling data from `.docx` files and struggle with the robotic text format typical of 3GPP specifications and technical reports. Furthermore, they do not effectively manage non-textual data such as figures and tables, which are crucial in these documents, comprising about 66% of the embedded knowledge.

While substantial progress has been made in developing domain-specific LLMs and creating instruct datasets, significant gaps remain in the availability of specialized datasets for the 5G domain. Although [145] and [100] provide methods to generate datasets for Telecom-specific LLMs and for benchmarking LLMs in telecom, respectively, these approaches are broad and cover the entire Telecom domain. Furthermore, despite figures and tables within TSs containing important information that does not exist in the text, these approaches do not take them into account in 3GPP TSs. Thus, an automated pipeline to convert 5G 3GPP TSs (including figures and tables) into a 5G-specialized LLM is absent from current research. Our contribution addresses these gaps by developing an innovative, LLM-centric data generation pipeline specifically for 5G. This pipeline generates instruct datasets from 3GPP TSs, enabling the creation of 5G-aware LLMs.

[3]https://huggingface.co/meta-llama/Meta-Llama-3-8B
[4]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[5]https://huggingface.co/datasets/bigscience/P3

7.3 System Design

Our proposed solution, 5G Instruct Forge, presents a comprehensive pipeline designed to transform the 3GPP TSs into a well-structured dataset optimized for fine-tuning, as depicted in Fig. 7.1. This transformation involves several stages: *(i) Specification gathering,* which can be done either through our automated scripts or manually. The specifications, generally in `.docx/.doc` formats, are then meticulously processed; *(ii) Cleaning and processing,* involves extracting essential elements from each document, such as the table of contents, tables, figures, abbreviations, and definitions. These elements are managed utilizing LLM data generation techniques. Following this, the documents are converted into plain text and concatenated to create an embedding database. This database plays a crucial role in the next stage; *(iii) Data generation using LLMs*, supports the creation of eight distinct task type entries using powerful state-of-the-art LLMs; *(iv) Post-processing,* verifies the correctness and integrity of the generated data.

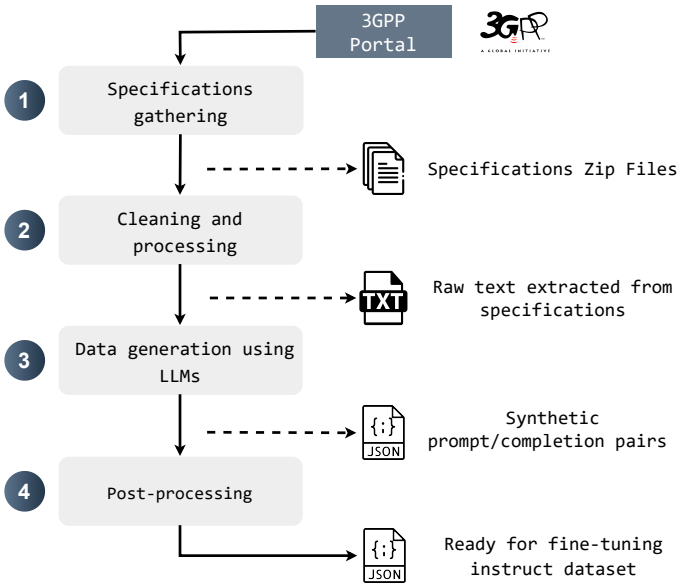


Figure 7.1: 5G Instruct Forge pipeline stages.

7.3.1 Specifications gathering

The first part of the project involves gathering 3GPP TSs from the 3GPP portal[6]. The latter is a central repository for all 3GPP TSs and reports, which are essential for developing and maintaining global telecommunications standards, particularly for mobile networks. The portal allows users to download all specifications from a specific release or a subset of specifications relevant to their needs. Users can also specify a list of 3GPP specifications by providing the release number and the specific ID of the documents, such as "23.209", where "23" represents the release number and "209" is the specification ID. The downloaded files are provided in `.zip` archives, which include the specifications in `.docx` or `.doc` formats, along with other files in `.yaml` or `.taml` formats. We are only interested in the `.docx` files and the `.doc` files are converted to `.docx` to standardize the process. The 3GPP TSs follow a consistent format and adhere to specific styling policies, including the use of bold text, various heading levels (T1), and

[6]https://portal.3gpp.org/

structured paragraphs, as shown in Fig. 7.2. This consistency in formatting makes it possible to systematically extract information using the `python-docx`[7] library.

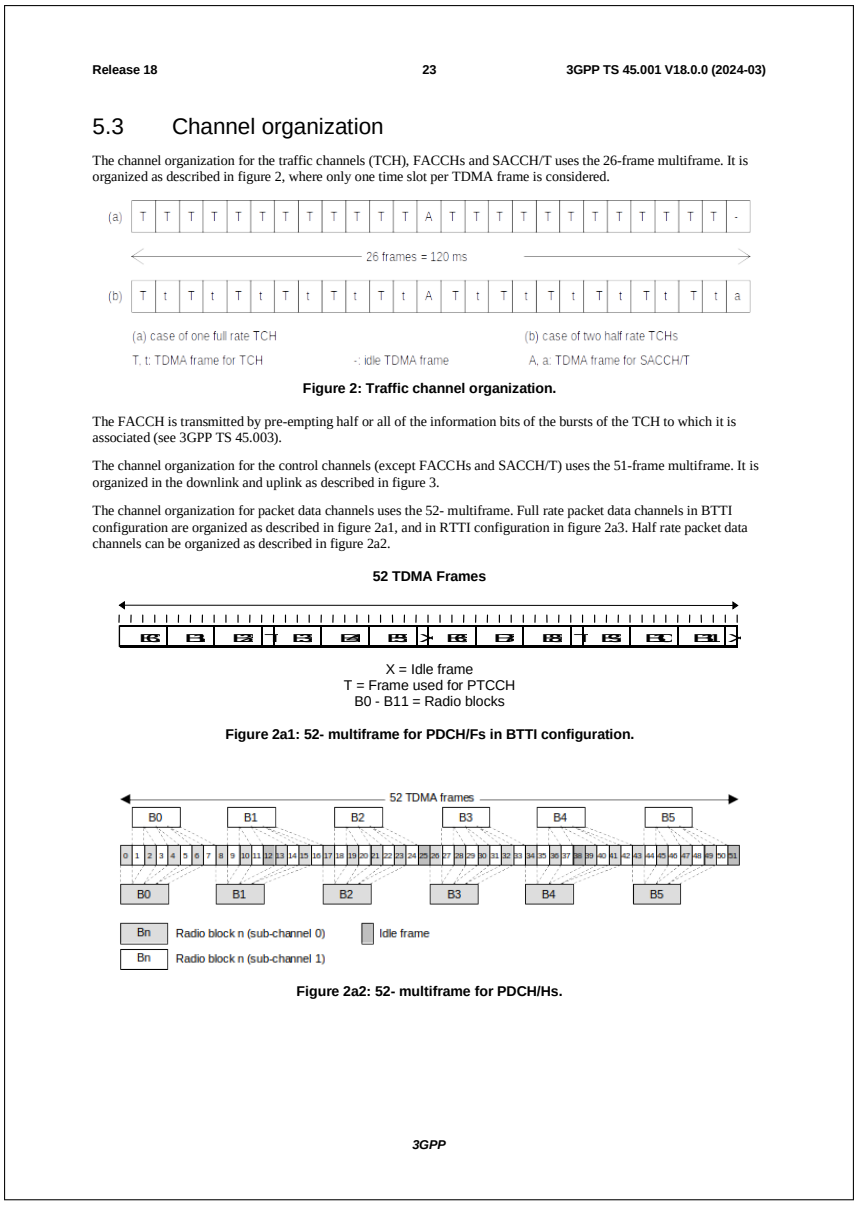


Figure 7.2: Page 23 of the 3GPP TS 45.001 V18.0.0 (2024-03)

7.3.2 Cleaning and processing

In the cleaning and processing phase, the primary objective is to refine the extracted 3GPP TSs to ensure the dataset is clean, relevant, and structured for fine-tuning purposes. This involves several key steps. First, we remove the initial and final sections of the specifications, as these often contain ancillary information such as titles, 3GPP contact details, addresses, phone numbers, and annexes primarily written for API purposes. These sections are not pertinent to the core 5G knowledge we aim to capture, so we choose to exclude them aiming at streamlining the

[7] https://python-docx.readthedocs.io/en/latest/index.html

documents to focus on the valuable content. Next, we extract the table of contents from the main text files and store them separately. They provide a navigational map of the document structure and are useful for referencing and indexing. Additionally, we isolate non-textual data, such as figures and tables, which often contain the bulk of the information in the specifications. Neglecting this data would result in a significant loss of content. Therefore, we treat this non-textual data separately, aiming to convert it into textual formats that an LLM can process and understand. This approach ensures that all critical information is effectively captured and utilized.

Figures and their annotations are processed using a highly accurate Vision-Text model, namely MiniCPM[8], which provides detailed textual descriptions of the figures, ensuring that all visual information is translated into a textual format suitable for LLM training. Fig. 7.3 illustrates the text generation from 3GPP TS figures using the Vision-Text model. The process involves first loading the vision model, then converting the image to PNG format, and finally processing the image using the model to obtain the textual description. Similarly, tables are handled by feeding them into an advanced LLM, which generates detailed descriptions of the tabular data. This approach ensures that complex information contained in tables is accurately represented in a textual form, making it more accessible for the language model.

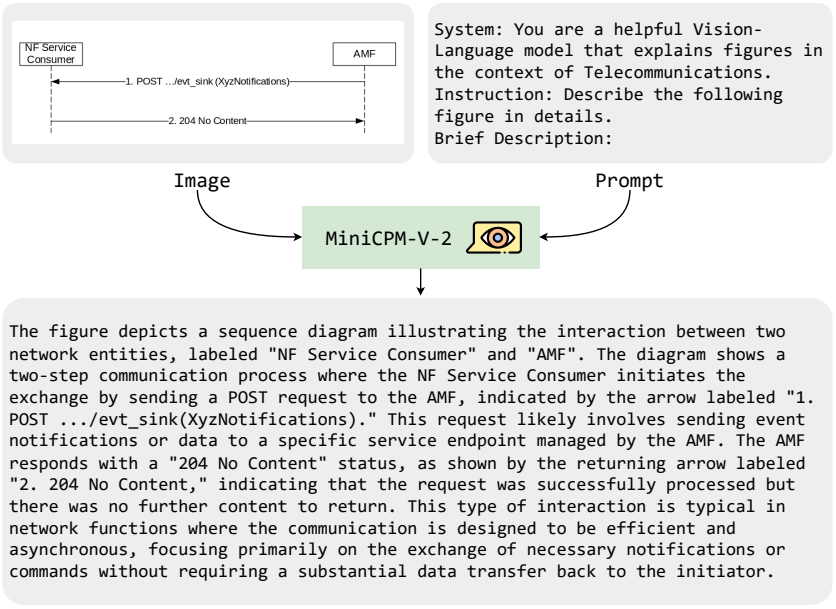


Figure 7.3: Figure textual description generation using MiniCPM.

In addition to processing structural elements of the specifications, we meticulously extract abbreviations and definitions from each document. This step is crucial as these elements encapsulate essential information that is imperative for both human understanding and LLMs to interpret the technical content accurately. Following the individual treatment of figures, tables, tables of contents, abbreviations, and definitions, we then extract the raw text from all `.docx` files of the specifications. This raw text is concatenated into a single, large `.txt` file, which forms the basis for the subsequent stages of our data generation pipeline. This consolidated text file facilitates a streamlined integration and manipulation of the data, ensuring a comprehensive dataset is

[8]https://huggingface.co/openbmb/MiniCPM-V-2

available for further analysis and machine learning applications. All these steps are summarized in Fig. 7.4.

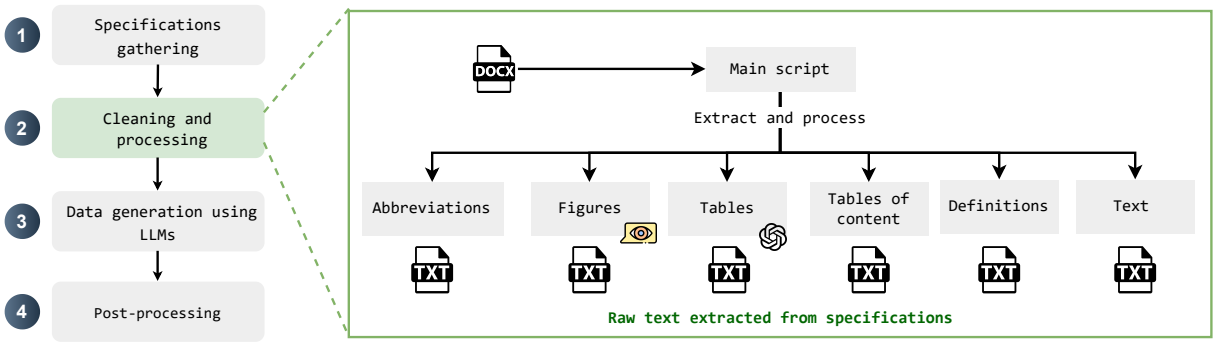


Figure 7.4: Illustration of the data cleaning and processing stage.

7.3.3 Data generation using LLMs

Fig. 7.5 illustrates the data generation stage using LLMs. This phase is a critical component of our pipeline, which focuses on transforming clean, raw textual data about 5G into a domain-specific dataset suitable for training LLMs to become 5G while maintaining their general language capabilities or evaluation. Our approach leverages both OpenAI's GPT-3.5 Instruct model[9] and the open-source Llama3 70B model[10] to create this dataset. The GPT-3.5 Instruct model is utilized for its superior accuracy and more controlled output, which reduces the number of inadequate entries during post-processing. Its advanced capabilities minimize the risk of generating inappropriate content. However, the use of GPT-3.5 comes with challenges, such as the need for data transmission to third parties, internet connectivity requirements, and potential latency issues. To address these challenges, we incorporate the Llama3 70B model, which, despite being less accurate due to its smaller size, offers advantages such as faster generation times and local operation with minimal latency. While this may result in a higher rate of inadequate entries that require additional filtering, it provides a valuable balance between control, speed, and data privacy.

The pipeline begins by creating an embedding database from concatenated text files generated in the previous step. This database is crucial as it facilitates the LLMs' contextual understanding during the generation of prompt/completion pairs. The concatenated text files are transformed into embeddings using an OpenAI model called text-embedding-3-large[11], which captures their semantic meanings. The resulting embeddings are essentially vectors of numerical values representing each word in the concatenated files and are stored in a vector database. We used Chroma DB[12] because it efficiently organizes and manages the embeddings, allowing the LLMs to quickly retrieve relevant information when generating responses. By adopting a RAG approach, the pipeline ensures that the LLMs access relevant context, thereby improving the quality and relevance of the generated data. Following the creation of the embedding database, the pipeline proceeds to generate either training data or evaluation data. The dataset for training will be an Instruct dataset used exclusively for training LLMs to create 5G-specialized models. In contrast,

[9]https://platform.openai.com/docs/models/gpt-3-5-turbo
[10]https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct
[11]https://openai.com/index/new-embedding-models-and-api-updates/
[12]https://www.trychroma.com

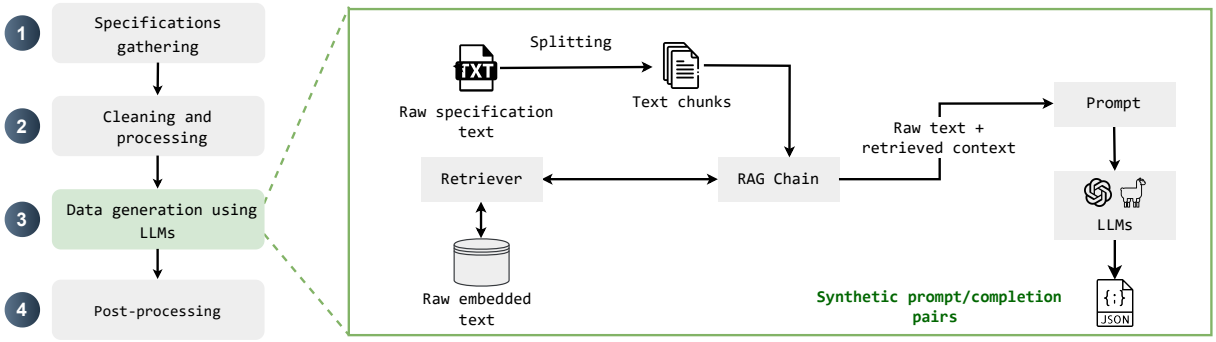


Figure 7.5: Illustration of the data generation using LLMs stage.

the evaluation data will be used to assess the created LLMs' abilities in 5G knowledge. Next, we delve into the details of each part.

Training data generation

Our pipeline is designed to produce a variety of training tasks, including question-answering pairs, fill-in-the-gap pairs, text reformulation, title and summary generation, and other specialized tasks. These tasks are important because they enable the LLMs to: (i) Learn raw knowledge through question-answering; (ii) Identify key entities and concepts with fill-in-the-gap tasks; (iii) Explore alternative expressions via text reformulation; and (iv) Correct misconceptions through tasks addressing false claims. Each type of task contributes significantly to the development of the LLM's domain expertise and language understanding. For question-answering tasks, the process involves two stages: generating questions and then generating answers. This two-pass approach ensures that the questions are well-formed and relevant, while the answers are accurate and informative. For simpler tasks, such as filling in the gaps, a single-pass approach is used where the LLM completes missing parts of the text.

Evaluation data generation

The evaluation data generation step is crucial for assessing the knowledge and capabilities of the trained LLMs. This step also relies on the concatenated text file containing the 5G specifications, but the structure of the evaluation dataset differs from that of the training data. We propose two types of evaluation datasets to test the LLM's understanding and generation abilities comprehensively: (i) The first type of evaluation dataset resembles an exam designed to assess both the knowledge and understanding of the LLM. This dataset includes columns for "question" and options labeled from 1 to 4, with the correct answer provided separately. To further enhance this dataset and introduce more diversity and challenge for the LLM, we incorporate yes/no questions where the only possible answers are "yes" or "no". This mixed format not only evaluates the model's ability to recall information but also tests its decision-making skills; (ii) The second type of evaluation dataset focuses on assessing the LLM's knowledge and generative capabilities through direct question answering. This dataset is similar in structure to the training data's question-answering task but without specifying an instruction. It includes questions derived from the 5G knowledge base, and the LLM is expected to generate accurate and contextually appropriate answers. This format allows for a straightforward evaluation of the model's ability to generate coherent and relevant responses based on its learned knowledge. By using these varied evaluation datasets, we can comprehensively assess the LLM's proficiency in 5G technology, ensuring that it not only understands the material but can also generate

high-quality responses.

7.3.4 Post-processing

In the post-processing phase, we focus primarily on verifying that the training and evaluation datasets conform to the predefined format. We systematically check for any entries that do not meet this format, identifying and removing those that are malformed or incomplete. By ensuring that all dataset entries adhere to this structure, we maintain the integrity and consistency necessary for effective training of domain-specific LLMs and their evaluation. This verification step is critical to eliminate noise and ensure that the final dataset is clean and reliable for use in developing 5G expert language models.

7.4 Performance Evaluation

The section is structured into three subsections: *(i) Experimentation setup,* which details the experimental setup; *(ii) Experimentation results,* which presents the experiments results; and *(iii) Discussions,* which offers additional insights related to the experiments.

7.4.1 Experimentation Setup

The evaluation preparation includes four steps: generating a 5G-related dataset for LLM training using the pipeline, creating 5G-aware LLMs, deploying 5G-aware LLMs, and preparing the evaluation datasets.

- *(i) Generating a 5G-related dataset:* We selected 22 3GPP TSs that were used to develop OAI[13], to generate the dataset using the proposed pipeline, named the OAI instruct dataset.
- *(ii) Creating 5G-aware LLMs:* Three open-source state-of-the-art LLMs were fine-tuned using the OAI instruct dataset with the freeze-tuning method within the LLaMA Factory framework [146], ensuring that a percentage of the core model's parameters were preserved while adapting it to the specific task. These LLMs are referenced in Table. 7.1, and the most important hyperparameters for LLM freeze-tuning are referenced in Table. 7.2;
- *(iii) Deploying 5G-aware LLMs:* The resulting 5G-aware LLMs were deployed on a single machine equipped with an Nvidia A100 GPU with 80GB of vRAM, utilizing the Llama Factory project for deployment. In this setup, we set the LLMs' temperature to 0.7 to balance creativity and coherence in the responses;
- *(iv) Preparing the evaluation dataset:* To validate the effectiveness of our methodology and the integration of knowledge into the resulting 5G-aware LLMs, we employed the evaluation benchmark from the OAI instruct dataset. This benchmark, comprising over 9,000 question-and-answer pairs, is specifically designed to measure the model's capacity to respond accurately to novel information about 5G technologies. Additionally, we incorporated a dataset featuring approximately 200 questions, each with four answer options (called 5G exam). This additional dataset aims to evaluate not only the LLM's expertise in 5G but also its proficiency in language comprehension. This dual assessment approach addresses concerns regarding potential compromises in the LLM's language capabilities due to the fine-tuning process.

[13]https://openairinterface.org

Table 7.1: Open-source LLMs used for training.

LLM	Number of parameters	Size	Reference
Llama3	8B	16 Gb	https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
Solar	10.7B	21 Gb	https://huggingface.co/upstage/SOLAR-10.7B-Instruct-v1.0
Mistral	7B	14 Gb	https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1

Table 7.2: Fine-tuning hyperparameters for each open-source LLM.

LLM	Trainable parameters	Number of epochs	Learning rate ($\times 10^{-5}$)	Per-device training batch
Llama3	9%	4	4	4
Solar	12%	4	4	4
Mistral	11%	4	5	4

7.4.2 Experimentation Results

The evaluation results are structured into four stages: *(i) OAI Instruct dataset,* which presents a dataset generated by the proposed pipeline to create 5G-aware LLMs; *(ii) 5G-aware LLMs training,* which outlines the chosen fine-tuning methodology and includes an ablation study on the hyperparameters of the corresponding fine-tuning approach; *(iii) 5G-aware LLMs quality,* which presents and analyzes the performance of the newly developed 5G-aware LLMs. At this stage, we conduct several evaluations: first, we assess knowledge injection through a multiple-choice exam designed to measure the LLMs's acquisition and application of new 5G domain knowledge. Second, we evaluate text generation quality to ensure that the LLMs produces coherent, contextually accurate, and linguistically consistent outputs across different prompts and question types. To this end, we apply metrics such as BERTScore and SemScore, while also monitoring generation times. This twofold evaluation allows us to distinguish between gains resulting from improved question understanding and those due to effective domain knowledge integration. Finally, *(iv) Expert satisfaction* assesses user satisfaction with the newly created 5G-aware LLMs, providing a qualitative perspective on the model's practical utility.

OAI Instruct dataset

As a proof of concept for our pipeline, we generated a dataset named OAI Instruct. This dataset was created by applying our data generation pipeline to a set of 22 3GPP TSs, upon which the renowned open-source implementation of 5G networks, OAI, is built. The resulting dataset, which is available at [14], will be used to create a specialized LLM capable of interacting with and understanding aspects of OAI (i.e., 5G). The OAI Instruct dataset includes 87,719 entries in the training set and 9,557 entries in the test set with a total size of ≈100Mb (≈80Mb for training & ≈6Mb for evaluation). The training dataset comprises columns for 'instruction', 'task_type', 'input', 'completion', and 'prompt', with a diverse range of instructions and task types. The test dataset is structured differently, featuring 'completion', 'prompt', and 'completion2' columns to evaluate the model's performance using known metrics, such as BERTScore[147] and SemScore [148], which require multiple references, hence the use of two completions instead of just one. Examples of entries in the instruct-type dataset are shown in Fig. 7.6.

In comparison to other domain-specific instruct datasets found in the literature, the finance-

[14]https://huggingface.co/datasets/Netsoft/oai-instruct

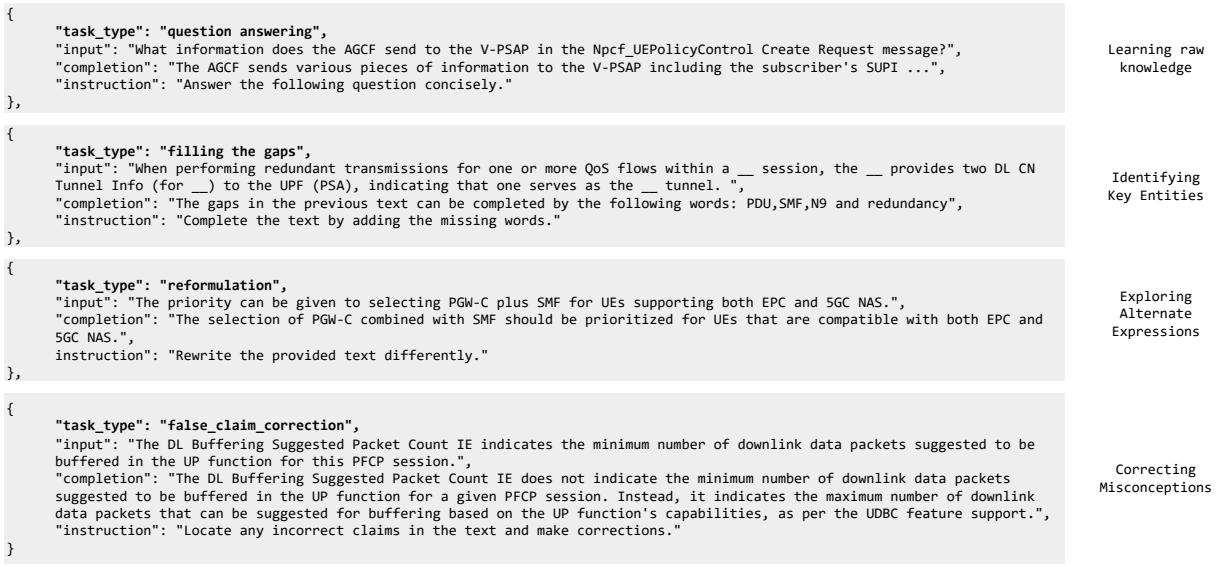


Figure 7.6: Example entries of OAI Instruct dataset, generated using 5G Instruct Forge.

alpaca dataset[15] is an illustrative example, with a size of 42.9 MB. It has successfully facilitated the fine-tuning of models such as distilgpt2-finance[16] and Llama-personal-finance[17], featuring a structure that includes the key fields of 'instruction', 'input', and 'output', similar to our dataset. Another noteworthy dataset is lawinstruct[18], which, due to its larger size and broader source inclusion, supports a more extensive range of fine-tuned LLMs. Our goal underscores the effectiveness of modest-sized datasets, suggesting that a well-structured dataset of ≈80Mb, coupled with the right fine-tuning approach and hyperparameters, can proficiently develop an LLM expert capable of learning, memorizing, and reasoning within a targeted subfield. This approach aligns with our strategy of creating highly specialized, efficient LLMs without the need for excessively large or comprehensive datasets.

5G-aware LLMs training

To demonstrate the effectiveness of our dataset, we initially employ it to fine-tune LLMs. Fine-tuning involves adapting pre-trained models to specific tasks or domains using specialized datasets, which mathematically entails adjusting the model's parameters θ to minimize a loss function $L(\theta)$ across the dataset. The objective of fine-tuning can be formalized as:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, f(x_i; \theta)) \tag{7.1}$$

where x_i represents the input data, y_i the corresponding labels, $f(x_i; \theta)$ the model's predictions, and N the number of training examples. The loss function L typically quantifies the discrepancy between the model's predictions and the actual labels, such as the cross-entropy loss in classification tasks. In practical terms, fine-tuning utilizes gradient descent-based optimization algorithms to update the parameters. The gradient $\nabla_\theta L$ of the loss function concerning the

[15]https://huggingface.co/datasets/gbharti/finance-alpaca
[16]https://huggingface.co/lxyuan/distilgpt2-finetuned-finance
[17]https://huggingface.co/dmedhi/llama-3-personal-finance-8b-bnb-4bit
[18]https://huggingface.co/datasets/lawinstruct/lawinstruct

parameters is computed, leading to the parameter updates:

$$\theta \leftarrow \theta - \eta \nabla_\theta L \tag{7.2}$$

where η is the learning rate. An efficient approach known as PEFT [149] involves updating only a subset of the model's parameters, thus reducing computational demands. Commonly, this entails freezing most of the model's layers and fine-tuning only the upper layers, expressed as:

$$\theta_f = \theta_f - \eta \nabla_{\theta_f} L, \quad \theta_u = \text{constant} \tag{7.3}$$

Here, θ_f represents the fine-tuned parameters, while θ_u denotes the unfrozen parameters. Techniques like LoRA [88] adjust a minimal percentage of the total parameters through small, trainable modules. However, we opted not to use LoRA due to the significant domain shift involved in our project. LoRA is better suited for minor modifications, such as subtle linguistic shifts or learning from minimal data. Instead, we employed the freezing method, which enables the integration of substantial new knowledge without compromising the original LLM's language capabilities. This approach effectively manages significant domain transitions while preserving the model's linguistic integrity [149].

Indeed, freeze-tuning requires several parameters to be set, which are specified in Table 7.2. We conducted an ablation study to determine the optimal hyperparameter configuration, which includes:

- *Percentage of trainable parameters:* This parameter underwent an ablation study to determine its optimal value, significantly affecting model performance for each open-source LLM. For example, Fig. 7.7 illustrates the ablation study of the Mistral LLM. This figure shows the performance of the 5G-aware Mistral LLM using freeze tuning on MMLU benchmark [150] and the 5G exam from our resulting OAI Instruct dataset. This latter involved an examination format with 200 questions, each presenting four potential answers but only one correct response. The objective of the fine-tuning is to inject knowledge into an LLM without compromising its existing knowledge. Therefore, the MMLU score measures whether the LLM has forgotten its default knowledge, while the 5G exam score assesses the knowledge acquired. From the figure, we observe that as the percentage of trainable parameters increases, the LLM learns the new knowledge better (yellow bars) without forgetting its old knowledge (blue bars). However, when training more than 11% in the case of Mistral, the LLM tends to forget much of its default knowledge and struggles to respond accurately to 5G questions because it loses the ability to answer questions based on its prior knowledge. Therefore, we choose 11% as the percentage of trainable parameters for the Mistral LLM.

- *Number of Epochs:* We chose 4 epochs based on preliminary experiments that showed diminishing returns on performance beyond this point. This allows the model to learn adequately without overfitting.

- *Learning Rate:* A learning rate of 4×10^{-5} was selected to ensure effective convergence. This range has been shown to provide a good balance between rapid convergence and stability during training, as demonstrated in the state of the art.

- *Per-Device Training Batch Size:* We set the batch size to 4, which allows for efficient use of computational resources while maintaining stability in gradient updates. This size was chosen to prevent memory overflow while ensuring that the model receives enough data for updates.

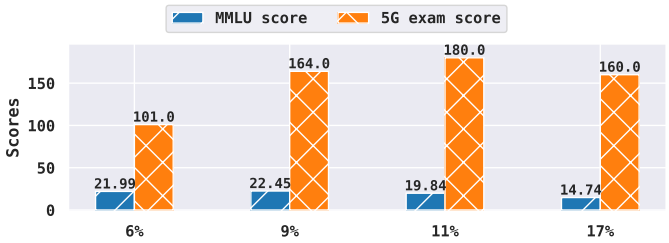


Figure 7.7: Impact of the percentage of trained parameters on MMLU [150] and 5G exam scores.

5G-aware LLMs quality

Following the fine-tuning process, we conducted a rigorous evaluation that consisted of comparing the LLM's answers to the benchmark questions against the reference answers from the dataset. To quantify the similarity between the generated and reference answers, we employed:

- *BERTScore [147]:* This metric uses cosine similarities between token embeddings from models like BERT to evaluate textual similarity and model performance. BERTScore includes three key metrics: precision, recall, and F1 score. Precision calculates the relevance of the candidate text by measuring the cosine similarity of each token to its closest counterpart in the reference text. Recall evaluates how comprehensively the candidate text covers the reference text. The F1 score combines these metrics, providing a balanced measure of textual accuracy and relevance, making BERTScore particularly effective at capturing paraphrases and maintaining semantic accuracy across different sentence structures.

- *ROUGE [151]:* This metric assesses the quality of summaries by computing overlap statistics between the generated text and a set of reference texts. ROUGE includes several key measures, such as ROUGE-N and ROUGE-L. ROUGE-N (e.g. ROUGE-1) measures the overlap of n-grams between the generated text and the references, serving as a proxy for precision and recall at the n-gram level. ROUGE-L focuses on the longest common subsequence, evaluating the fluency and order of the generated text relative to the references. Together, these metrics provide a comprehensive evaluation of textual coherence, consistency, and relevance, making ROUGE particularly effective for assessing the quality of text summarization tasks.

- *SemScore [148]:* This metric evaluates the semantic textual similarity of text generated by models. It consists of comparing the models' output directly with target responses. SemScore computes the cosine similarity between the embeddings of the model response and the target, using high-quality transformer models such as MPNet-Base [152] to generate these embeddings. This approach allows SemScore to effectively assess whether the generated text is contextually appropriate and semantically aligned with the target responses.

- *BLEU [153]:* This metric evaluates the quality of machine-generated text by measuring how well the output aligns with a set of reference texts. BLEU includes several key components, including precision scores that assess the overlap between machine-generated text and reference texts combined using a geometric mean. It also incorporates a brevity penalty to discourage overly short responses. This penalty ensures that the candidate texts not only align well with the reference texts but also cover an adequate length, providing a balanced measure of linguistic accuracy and completeness.

- *METEOR [154]:* This metric evaluates the quality of machine-generated text by assessing both exact word matches and semantic similarity between the candidate text and reference

texts. METEOR considers synonyms and stemming, providing a more nuanced evaluation of semantic accuracy. This approach helps capture the meaning of the text rather than just use the exact word, making METEOR particularly effective at ensuring translational adequacy and fluency while maintaining semantic integrity across different languages and structures.

From Fig. 7.8, we observe several key improvements across different subfigures. In Fig.7.8(a), the BERTScore precision, recall, and F1 scores show significant enhancements for the 5G-aware versions compared to the default version, indicating effective assimilation of domain-specific 5G knowledge. Additionally, these 5G-aware LLMs outperform OpenAI's GPT-4[19], with improvements of 5%, 6%, and 4% in BERTScore's F1, precision, and recall metrics, respectively. Fig.7.8(b) and Fig.7.8(c) illustrate notable gains in the ROUGE-L and ROUGE-1 metrics for the 5G-aware models. The enhanced ROUGE-L precision, recall, and F1 scores in Fig.7.8(b) suggest better content and structure preservation. Similarly, the improved ROUGE-1 scores in Fig.7.8(c) reflect better content retention and contextual relevance. In both cases, the 5G-aware LLMs outperform GPT-4. Finally, Fig. 7.8(d) highlights superior performance in SemScore, BLEU, and METEOR metrics for the 5G-aware LLMs, indicating better alignment with reference texts and enhanced semantic accuracy. Overall, GPT-4's default version exhibits limitations in handling specialized topics like 5G, underscoring the need for fine-tuning. Additionally, GPT-4's proprietary nature poses accessibility and cost issues, emphasizing the importance of fine-tuning accessible LLMs to address knowledge gaps in evolving technological fields.

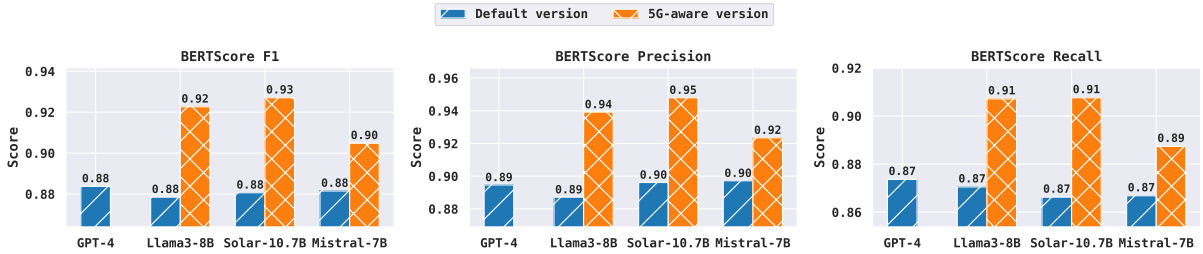
Fig. 7.9 shows the Perplexity [155] metric for the GPT-4 LLM and both the default and 5G-aware open-source LLMs, calculated using the generated outputs from the previous benchmark questions. Perplexity is a standard metric used to evaluate language models by measuring how well a model predicts a sample of text. It quantifies the model's uncertainty when generating or interpreting a sequence of words. Specifically, perplexity is the exponentiation of the average negative log-likelihood of the correct word sequence under the model's predicted probability distribution:

$$\mathrm{PPL}(p) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p(w_i|w_1, w_2, \ldots, w_{i-1})\right) \tag{7.4}$$

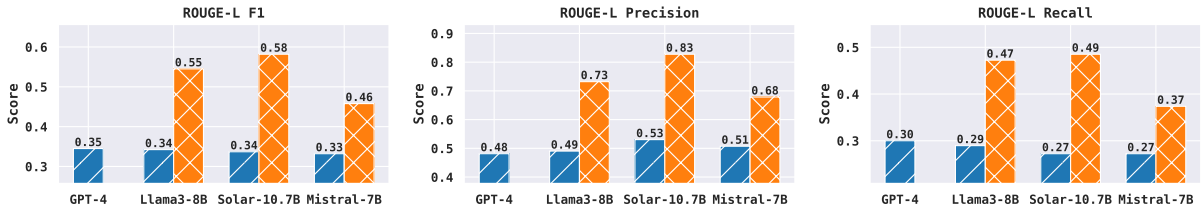
where N is the total number of words in the sequence, and $p(w_i|w_1, w_2, \ldots, w_{i-1})$ represents the conditional probability of word w_i given the preceding words in the sequence. Lower perplexity indicates that the model is more confident and accurate in its predictions. For open-source LLMs, we used the corresponding tokenizer to calculate the score, whereas, for GPT-4, we used the GPT-2 tokenizer[20], as it is open-source. From the figure, we can see that the Perplexity score of the newly trained LLMs is lower than that of the default version. This means that 5G-aware LLMs were more confident in generating 5G-related responses, demonstrating that the training was successful (as a compelling example, Solar's perplexity decreased from 878.31 to 21.30 after training).

Fig. 7.10 shows the mean generation time for GPT-4, the default, and 5G-aware versions of open-source LLMs for responses regarding the previous evaluation. From the figure, we can see that, despite the added internet latency, GPT-4 ranks first with an average time of 1.52 seconds compared to the default versions of the LLMs. However, open-source LLMs are as fast despite their smaller size. Moreover, we can see that our proposed fine-tuning does not affect generation

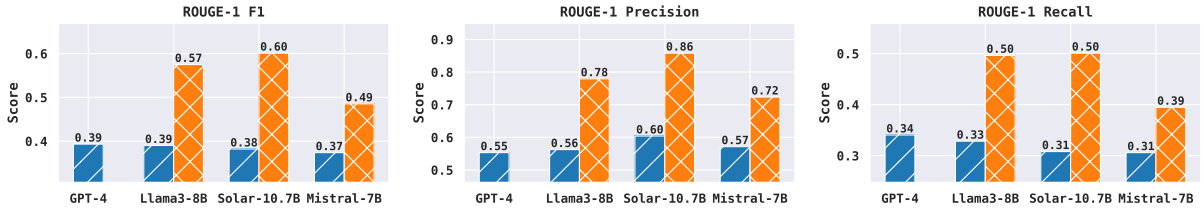
[19] gpt-4-0125-preview
[20] https://huggingface.co/openai-community/gpt2



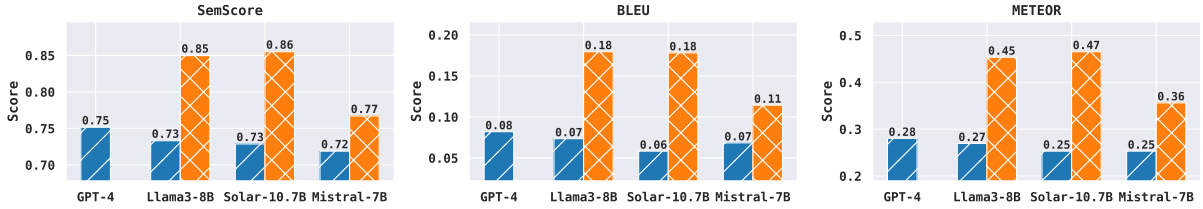
(a) BERTScore metrics.



(b) ROUGE-L metrics.



(c) ROUGE-1 metrics.



(d) SemScore, BLEU and METEOR metrics.

Figure 7.8: Comparison of LLMs metrics. Bars colored in blue represent the Default versions of the models, while bars colored in orange represent the 5G-aware versions.

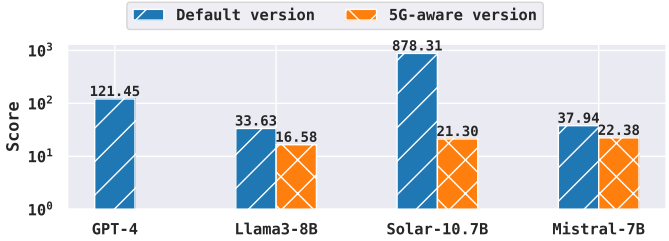


Figure 7.9: Perplexity score.

time significantly, as the difference between the default versions and the 5G-aware versions is minimal. Nevertheless, since these times are longer than GPT-4's (except for 5G-aware Mistral), the research community should investigate inference speed techniques, such as [156], so

that these LLMs can be used in 5G decision-making problems, which require fast decision speeds.

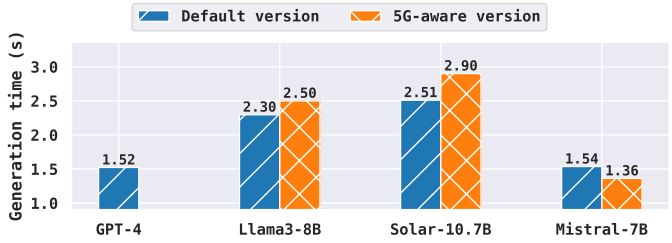


Figure 7.10: Mean generation time on the Q/A evaluation dataset.

In addition, in Fig. 7.11, we aimed to precisely gauge the understanding of the resulting 5G-aware LLMs on the 5G exam from OAI Instruct. This method allowed us to directly measure each model's comprehension of the material rather than just its ability to generate plausible text. The results from this examination reveal significant distinctions in model performance, particularly highlighting the superior understanding of 5G topics by 5G-aware LLMs compared to GPT-4, which scored 156 correct answers. Fine-Tuned Llama3, Solar-10.7B, and Mistral all demonstrated a higher number of correct responses, i.e., 186, 176, and 180, respectively, suggesting that the fine-tuning process has effectively enhanced their ability to grasp and accurately answer questions about specialized and technical content. This indicates not only an improved familiarity with the specific language and 5G concepts but also an enhanced capability to discriminate between closely related information. The Solar model initially struggled, not due to a lack of 5G knowledge but because of its difficulty in following instructions and handling multiple-choice questions. Post fine-tuning, Solar10.7B showed marked improvement in instruction adherence and began successfully answering such questions, thereby doubling its efficacy by integrating new 5G knowledge. Mistral, while proficient in following instructions, showed a noticeable deficiency in 5G-specific knowledge compared to newer models, reflecting its training focus on different domains.

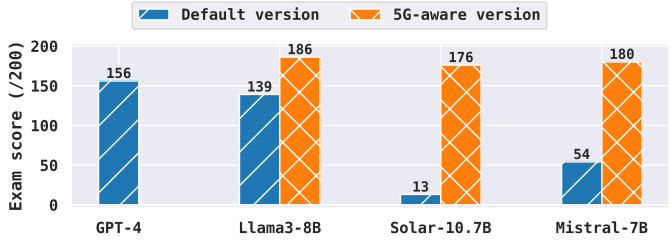


Figure 7.11: Exam results of different LLMs.

Expert satisfaction

To enhance the evaluation of LLMs and gain more detailed insights into the quality of generated responses, we enlisted experts to verify the completions. These experts rated the quality of the LLMs' answers on a scale of 1 to 5. As illustrated in Table 7.3, the overall results showed significant improvements in the Q/A parts of the 5G-aware LLMs compared to their default versions. Specifically, Llama3 ranks first with an improvement of ≈108%, followed by Solar with ≈100%, and Mistral with ≈90%. This showcases that trained LLMs can respond to 5G-related Q/A questions better than their default versions. This step, a critical part of our assessment

methodology, provides an empirical measure of the LLMs' performance in generating accurate and relevant content within the field.

Table 7.3: Improvement in Expert Satisfaction.

LLM	Default version (/5)	5G-aware version (/5)	Improvement (%)
Llama3-8B	1.55	3.22	107.74%
Solar-10.7B	1.25	2.50	100%
Mistral-7B	2.00	3.80	90%

7.4.3 Discussions

The specialized LLMs developed in our study demonstrate strong potential for advanced future network management applications, such as making informed decisions in self-healing systems, detecting anomalies in logs, and managing self-regulating systems. These capabilities are crucial as we move towards 6G technologies, highlighting the importance of specialized LLMs in the future of telecommunications. Below, we provide some conclusions on: (i) the effect of dataset size on LLM quality, (ii) LLM cost-effectiveness, and (iii) the generalizability of newly created LLMs.

Dataset size efficiency

After fine-tuning three LLMs on our proof-of-concept dataset, OAI Instruct, we evaluated their ability to learn new 5G-specific knowledge and compared their performance to the more general but less specialized GPT-4 model. Our results show that our method effectively enables LLMs to acquire 5G knowledge, even with a relatively small training dataset of approximately 80 MB. Increasing the dataset size could further enhance the LLMs' specialization, potentially allowing them to cover the entire 5G domain. However, due to high costs, the current focus in both industry and research is on developing specialized rather than general LLMs [157].

LLM cost-effectiveness

Using 5G Instruct Forge, the first step is dataset generation, which was conducted with OpenAI's GPT-4 and Llama3 LLMs. To minimize costs, we can rely solely on free open-source LLMs to generate this dataset. In this regard, we utilized a single GPU with quantization techniques for an open-source LLM that is not inherently specialized in 5G. By adopting a RAG approach, we efficiently generated a domain-specific 5G dataset. Data generation can be accomplished in just a few minutes using a local open-source LLM running on a single GPU, making it a resource-efficient process. The second step is fine-tuning, which also requires only one GPU and takes approximately 15 hours. With a minimal investment of a few minutes for data generation and 15 hours for fine-tuning on one GPU, we developed a powerful domain-specific LLM with fewer than 10 billion parameters. For inference, the created LLM is compact and efficient, designed to run on a single GPU while consuming significantly less energy than larger LLMs. It is also local and free compared to GPT-4, outperforming it in 5G-related tasks. Thus, for 5G applications, we can confidently utilize these small, cost-effective models for various use cases, ensuring privacy and security, such as in local anomaly detection and critical 5G information management on private networks. We firmly believe that creating domain-specific LLMs offers the most cost-effective solution, as described in [157].

LLM generalizability

It should be noted that the 5G-aware LLMs are trained on a set of 3GPP TSs using the freeze-tuning method, meaning only knowledge from these TSs is injected into the LLMs. They retain their previous capabilities while incorporating this additional knowledge. However, since LLMs can reason, if there is information in other TSs that can be deduced from the TSs used to train the LLM, we believe that the LLM can infer that information. Conversely, for new knowledge, these LLMs will not be able to provide a response. To adapt them, we need to use the 5G Instruct Forge again to create the embedding database from the new TSs and initiate fine-tuning to inject the new information into the LLM's knowledge. This way, they will retain both their old knowledge and acquire the new knowledge. Depending on the new TSs, the fine-tuning time will vary. As a reference, in our fine-tuning process, we used 22 TSs, which took approximately 15 hours; thus, the time required will differ based on the new set of TSs.

7.5 Conclusion

In this chapter, we introduced the "5G Instruct Forge," a cutting-edge data engineering pipeline designed to improve LLM training using domain-specific datasets from 3GPP TSs. Our evaluations show that LLMs trained with our OAI Instruct dataset outperform conventional models like GPT-4 in 5G-specific tasks, demonstrating significant performance improvements. This work advances the use of LLMs in telecommunications and provides a framework that can be extended to other technological domains. Looking ahead, our research has important implications for the development of future networks like 6G, which will rely on technologies such as self-healing systems and zero-touch management systems. These advanced networks will depend on automation capabilities to enable dynamic and efficient network management without human intervention.

In next-generation 6G networks, a dense set of language tasks will emerge, ranging from intent translation to assurance, which the management system will request LLMs to handle. Given limited resources, dedicating a single LLM to each task is not feasible. Therefore, these tasks will share a set of LLMs, and designing a scheduler that can efficiently route all IBN tasks to the available models is essential for their effective deployment. This aspect is explored in the next chapter.

Chapter 8

Optimizing Telecom-aware LLMs

8.1 Introduction

Recently, with the rapid expansion of GenAI, LLMs, advanced 6G systems capable of understanding and generating human-like text, have attracted significant attention for their applications in networking problems [137]. These models, such as OpenAI's closed-source GPT series and the open-source Llama and Mistral series [158], excel in various tasks, including coding, reasoning, and language processing. This makes them particularly well-suited for networking tasks that involve, for example: (i) coding, for generating network code from 3GPP standards [159]; (ii) reasoning, for autonomous anomaly resolution based on 6G KPIs to enable ZSM [104]; and (iii) language processing, for intent-based human-network interactions to enable IBN [109]. However, LLMs face challenges due to their large size and high computational demands, which complicate their deployment in 6G networks. To address this, LLMs must be shared across multiple 6G tasks. By leveraging in-context learning, where different prompts (inputs) are used without altering the LLMs' weights, these models can handle diverse tasks concurrently [137].

While the sharing of a single LLM across applications provides an intuitive motivating scenario, practical 6G deployments are expected to rely on a pool of heterogeneous LLMs, each exhibiting different strengths in terms of reasoning capability, language proficiency, and inference latency. In this generalized multi-LLM setting, each task must be completed with a high score within a specified deadline, defined as part of the SLOs. Consequently, selecting the most suitable LLM for each task, while accounting for task deadlines and the dynamic load on each model, becomes essential. This leads to a multi-objective optimization problem that aims to maximize task scores while minimizing task latencies. The problem is further complicated by unpredictable task arrival times, heterogeneous task types, and the fact that task scores can only be evaluated after response generation. To address these challenges, we propose a DRL-based task scheduling framework that optimizes task allocation across multiple LLMs. The DRL agent observes the load on the LLM queues, as well as the type and deadline of arriving tasks within a given time window, and dynamically routes each task to the most appropriate LLM. The main contributions of this chapter are manifold:

- We model the multi-objective optimization problem and the objective function to maximize task scores and minimize task latencies.
- We introduce a DRL-based solution to address the aforementioned objective function of the optimization problem. The state space includes the task load on each LLM's queue, as

well as the type and deadline of the arriving task. The action involves routing the arriving task to one of the LLMs, and the reward is formulated according to the objective function.

- The proposed DRL solution is designed to handle multiple tasks with varying types and arrival patterns, i.e., the DRL agent is trained only once and then deployed, regardless of the number of tasks and their arrival patterns.
- We evaluate the solution in a realistic setting by deploying four open-source LLMs on two Nvidia A100 GPUs with 40GB of vRAM. The realistic tasks and their evaluations, i.e., score computation, are sourced from [160], and task arrivals are simulated using a Poisson distribution.

8.2 Related Works

The topic of LLM serving is relatively new, with recent research exploring various aspects of the field. For instance, *Liu et al.* in [161] address the assignment of tasks to appropriate closed-source LLMs as a multi-objective optimization problem, aiming to minimize costs while maximizing performance. They employ a heuristic approach to route tasks to specific LLMs; however, their reliance on closed-source LLMs and focus on minimizing API costs presents a notable limitation in the context of SLO-aware 6G networks, as they do not consider task deadlines. In another study, researchers in [162] propose Aladdin, a heuristic-based scheduler designed to route tasks to a set of LLMs to meet task LLMs. However, this method depends on predicting the output length of LLMs before scheduling, which is challenging in fast-changing 6G networks. Generally, heuristic-based schedulers struggle with dense task arrivals, making them less suitable for the high-density environments of 6G networks. As an alternative, DRL has shown promise in rapidly changing environments. In [163], DRL is used to route tasks to a set of LLMs. However, this approach faces difficulties when applied to LLM serving in 6G networks, as it requires an estimate of the task arrival rate in the state space, an estimation that is particularly challenging in the demanding and heterogeneous applications of 6G networks. Additionally, their reward computation is based on model accuracy, which is not an effective metric for evaluating LLMs. In contrast, our DRL approach differs from the aforementioned methods by considering different types of tasks and remaining agnostic to task arrival rates. Moreover, we adopt a more realistic reward computation methodology for each type of task based on the framework proposed in [160].

8.3 System Model

We consider a system composed of a set of m LLMs, denoted as $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_m\}$. Each LLM ℓ_j has an associated queue with a capacity $\mathcal{C}_j$ and a token generation speed δ_j (tokens per second). The system receives a set of n tasks, denoted by $\mathcal{K} = \{k_1, k_2, \ldots, k_n\}$. Each task k_i arrives at time t_i, has a specific deadline τ_i, and can generate different output token lengths $\mathcal{P}_{i,j}$ while achieving varying scores $\Phi_{i,j}$ depending on which LLM ℓ_j processes it. The assignment of tasks to LLMs is represented by the binary decision variable $x_{i,j}$, where $x_{i,j} = 1$ if task k_i is assigned to LLM ℓ_j, and $x_{i,j} = 0$ otherwise.

The objective of the system is to maximize the total score obtained from processing the tasks while minimizing penalties associated with tasks that miss their deadlines. The objective func-

tion is initially formulated as follows:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} \Phi_{i,j} - \sigma \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} \max \left(0, t_i + \frac{\mathcal{P}_{i,j}}{\delta_j} - \tau_i\right) \tag{8.1}$$

Here, the first term maximizes the total score achieved by the system, while the second term imposes a penalty for tasks that exceed their deadlines, with σ being the penalty factor.

To eliminate the use of the max function, we introduce an auxiliary variable $z_{i,j}$. The new objective function is formulated as follows:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} \Phi_{i,j} - \sigma \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} z_{i,j} \tag{8.2}$$

Here, the auxiliary variable $z_{i,j}$ is defined through the following constraints:

$$\begin{aligned} &z_{i,j} \geq t_i + \frac{\mathcal{P}_{i,j}}{\delta_j} - \tau_i \quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in \{1, 2, \ldots, m\}, \\ &z_{i,j} \geq 0 \quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in \{1, 2, \ldots, m\} \end{aligned} \tag{8.3}$$

This optimization problem is subject to several constraints: First, the binary assignment constraint enforces the nature of the decision variable $x_{i,j}$, ensuring that it takes only binary values:

$$x_{i,j} \in \{0, 1\} \quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in \{1, 2, \ldots, m\} \tag{8.4}$$

This constraint ensures that each task is either assigned to a specific LLM or not, preventing fractional assignments and ensuring clear decision-making in the task assignment process.

Second, the task assignment constraint ensures that each task is assigned to exactly one LLM. This is mathematically expressed as:

$$\sum_{j=1}^{m} x_{i,j} = 1 \quad \forall i \in \{1, 2, \ldots, n\} \tag{8.5}$$

This constraint guarantees that no task is left unassigned or assigned to multiple LLMs, ensuring that each task has a clear and unique processing route.

Thrid, the queue capacity constraint restricts the number of tasks assigned to any LLM at any given time to be within its queue capacity:

$$\sum_{i=1}^{n} x_{i,j} \leq \mathcal{C}_j \quad \forall j \in \{1, 2, \ldots, m\} \tag{8.6}$$

This constraint ensures that the number of tasks allocated to a particular LLM does not exceed that LLM's queue capacity.

Finally, the deadline constraint requires that the time taken to complete each task respects its deadline. This can be expressed as follows:

$$\begin{cases} t_i + \frac{\mathcal{P}_{i,j}}{\delta_j} \leq \tau_i & \text{if } x_{i,j} = 1, \\ \text{No constraint imposed} & \text{if } x_{i,j} = 0. \end{cases} \tag{8.7}$$

To eliminate the explicit conditional logic, we introduce a large constant $\mathcal{M}$. The updated constraint becomes:

$$t_i + \frac{\mathcal{P}_{i,j}}{\delta_j} \leq \tau_i + (1 - x_{i,j})\mathcal{M}, \quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in \{1, 2, \ldots, m\} \tag{8.8}$$

Here, $\mathcal{M}$ is a large constant that effectively deactivates the constraint when a task is not assigned to an LLM ($x_{i,j} = 0$).

From this formalization, we can classify the task scheduling optimization problem for LLM serving as a Mixed-Integer Linear Programming (MILP) problem. However, predicting key variables such as output token length $\mathcal{P}_{i,j}$, task arrival times t_i, deadlines τ_i, and scores $\Phi_{i,j}$, is particularly challenging in fast-varying 6G networks. This unpredictability highlights the need for adaptive scheduling strategies that can respond to dynamic conditions rather than relying solely on static optimization techniques.

8.4 System Design

As mentioned earlier, solving the optimization problem efficiently without prior knowledge of task information and output token length is challenging. For this reason, we propose a DRL approach as it abstracts the complexity and stochastic nature of the environment, allowing for efficient and quick decision-making that adapts to task arrival patterns and changes in the set of LLMs. Furthermore, DRL develops the ability to learn over time and adapt to various unseen situations. In the remainder of this section, we will describe the key components of RL, i.e., state space, action space, and reward function, followed by an explanation of the DRL approach employed, i.e., DQN.

8.4.1 RL key parts

As seen in Fig. 8.1, the DRL agent interacts with the environment to identify the appropriate task scheduling tactics. At each time t, the agent observes a state s_t and executes the appropriate action a_t, allocating task k to LLM ℓ according to the policy π. The state-action function, also known as the Q-function $Q(s_t, a_t)$, which a deep neural network can approximate, can decide this policy. The environment's state is transited to s_{t+1}, and each agent receives a reward of r_t. In our situation, the reward function is established by maximising each task score while satisfying tasks deadlines.

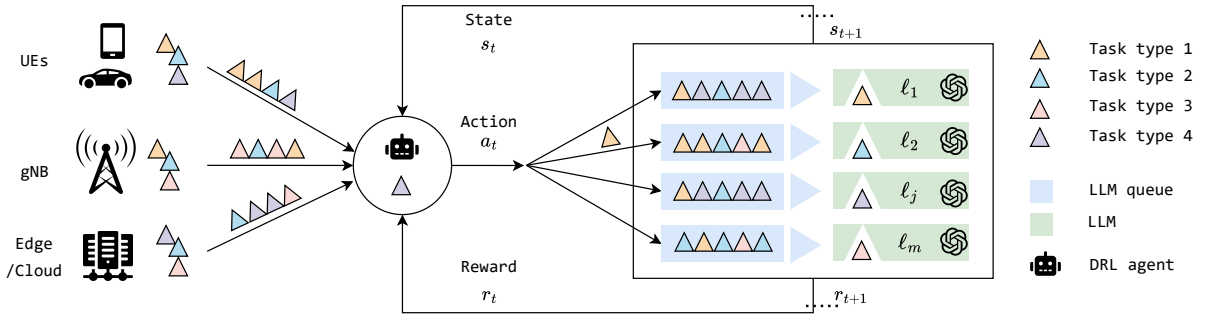


Figure 8.1: DRL-enabled task scheduling for LLMs serving in 6G networks design.

State

The state s_t at time t includes: (i) The queue sizes q_j for each LLM ℓ_j; (ii) The information about the newly arriving task, i.e., its type κ_i; and (iii) The task deadline τ_i. Formally:

$$s_t = (q, \kappa_i, \tau_i) \mid q = (q_1, q_2, \ldots, q_m) \tag{8.9}$$

This state design enables the DRL agent to remain agnostic to variations in task arrival rates and the set of LLMs in the system (token generation speeds). This flexibility ensures that task routing remains efficient for realistic scenarios in 6G networks, where LLMs can be updated or replaced, and task arrival rates can change rapidly.

Action

At time t, the action a_t involves assigning the incoming task k_i to one of the LLMs ℓ_j. The action is defined as:

$$a_t = j \quad \text{for } j \in \{1, 2, \ldots, m\} \tag{8.10}$$

Where $a_t = j$ indicates that the task k_i is assigned to LLM ℓ_j, meaning $x_{i,j} = 1$ for the selected LLM ℓ_j, and $x_{i,k} = 0$ for all other LLMs ℓ_v where $v \neq j$.

Reward

The reward r_t at time t is computed based on the completion of tasks assigned at previous time steps that finish exactly at time t. It accounts only for tasks that complete at this time, excluding those whose rewards were computed in earlier steps. The reward can be expressed as:

$$r_t = \sum_{i=1}^{n} \sum_{j=1}^{m} \Phi_{i,j} \mathbb{I}\left(t = t_i + \frac{\mathcal{P}_{i,j}}{\delta_j}\right) x_{i,j} \tag{8.11}$$

Here, $\mathbb{I}\left(t = t_i + \frac{\mathcal{P}_{i,j}}{\delta_j}\right)$ is an indicator function that equals 1 if the task k_i assigned at time t_i to LLM ℓ_j finishes at time t and 0 otherwise. The variable $x_{i,j}$ indicates that task k_i was assigned to LLM ℓ_j at time t_i. Thus, r_t ensures that only tasks completing at the current time contribute to the reward. Once a task's reward has been computed at t, it is excluded from future time step rewards. This reward design enables the DRL agent to adaptively learn and optimize its routing strategy in response to the dynamic conditions of 6G networks, where task completion times and scores may vary significantly based on the LLM used.

8.4.2 DRL Approach

DQN is ideal in the context of this problem due to its ability to effectively manage large state and action spaces while ensuring stability and convergence through experience replay and target networks [164]. In DQN, the objective is to find the policy π^* that maximizes cumulative rewards, represented as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right], \tag{8.12}$$

where γ is the discount factor, starting from any initial state s_0. The Q-function $Q(s, a; \theta)$ estimates the expected cumulative reward for taking action a in state s and subsequently following

the optimal policy. DQN addresses this by approximating the Q-function using a neural network with parameters θ, which is trained by minimizing the loss function:

$$\begin{aligned} L(\theta) &= \mathbb{E}\left[(y_t - Q(s_t, a_t; \theta))^2\right], \\ y_t &= r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-). \end{aligned} \tag{8.13}$$

Here, the target y_t includes the immediate reward r_t and the discounted maximum future reward predicted by a target network with parameters θ^-. This target network, which is periodically updated, helps stabilize training and enables the Q-function to converge more effectively.

At each time step t, the action a_t is chosen based on an ϵ-greedy strategy. With a probability ϵ, a random action is selected to encourage exploration, while with a probability $1 - \epsilon$, the action maximizing the Q-value is chosen to exploit learned knowledge. ϵ will decrease over time during the learning pushing the agent to explore the environment at the beginning of the training and driving it to exploitation over time. After executing the action, the reward r_t is obtained, and the transition (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer. This buffer allows the agent to sample previous transitions randomly during training, helping to break correlations between samples and stabilize learning. This overall approach promotes convergence and enables DQN to learn optimal policies in complex environments.

8.5 Performance Evaluation

The section is structured into three subsections: *Experimentation setup,* which details the experimental setup; *Experimentation results,* which presents and analyzes the performance of the DRL approach; and *Discussions,* which offers additional insights related to the experiment.

8.5.1 Experimentation Setup

Our experimental setup includes three machines hosting the DRL-DQN agent and the LLMs. The first machine, equipped with a 12th Gen Intel(R) Core(TM) i7-12700, is dedicated to the training and inference of the DRL agent. The two remaining machines, equipped with an Intel(R) Xeon(R) Gold 6240R CPU and an NVIDIA A100 GPU (40GB vRAM), are dedicated to the LLMs environment. Each GPU machine is used to deploy two LLMs, resulting in a total of four LLMs for evaluation. These four LLMs use the Instruct versions of: *phi3-3.8b*, *gemma2-9b*, *qwen2-7b*, and *llama3-8b*. The DRL-DQN agent interacts with the LLMs using API calls. This interaction is facilitated by the Ollama[1] framework, which serves the LLMs, while the Langchain[2] library is used for the interaction. Meanwhile, the DRL-DQN agent is trained using the Stable Baselines library [165]. Realistic tasks are derived from [160] and their arrival pattern is simulated using the Poisson distribution with a parameter λ. This latter indicates the mean number of tasks arriving per second. From [160], five types of tasks were considered: 'reasoning', 'coding', 'math', 'data_analysis', and 'language'. Table 8.1 present the parameters regarding DRL-DQN, LLMs and tasks information.

8.5.2 Experimentation Results

Fig. 8.2 presents the convergence evaluation of the DRL-DQN agent throughout the training process. The x-axis denotes the training steps corresponding to various episodes, with each

[1]https://ollama.com
[2]https://www.langchain.com

Table 8.1: Setup parameters.

Parameter	Value
Replay buffer size	50,000
Minibatch size	32
Hidden layers	2
No. of neurons	[128, 128]
Activation function	[ReLU, ReLU]
Optimizer	Adam
Learning rate	0.001
Discount factor γ	0.99
Epsilon decay	0.1 (initial) to 0.05 (final)
Step per Episode	30
Number of total steps	10,000
Time step	1s
LLM temperature	0
LLM top_p	0.95
LLM repeat_penalty	1.15
LLM queue size	10
Task deadlines	10s
Training Task arrival rate (λ)	1
Testing Task arrival rate (λ)	$0.5, 1, 2, 4, 8$

episode consisting of 50 steps, resulting in a total of 334 episodes. The y-axis represents the score, defined as the cumulative rewards obtained during each episode. Analysis of the reward curve indicates a gradual increase over time, converging after approximately 5,000 training steps. These results show that the DRL-DQN agent successfully identified a policy that outperforms earlier iterations in training, ultimately achieving a high reward score.

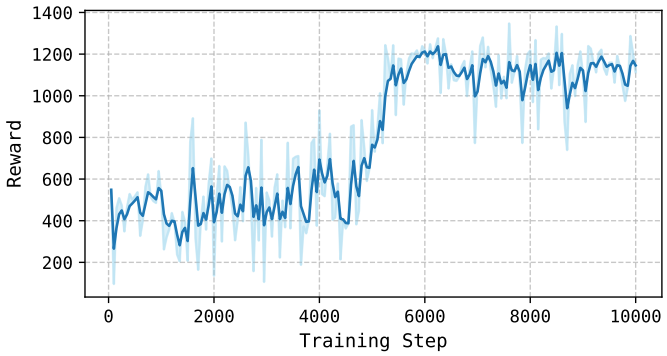
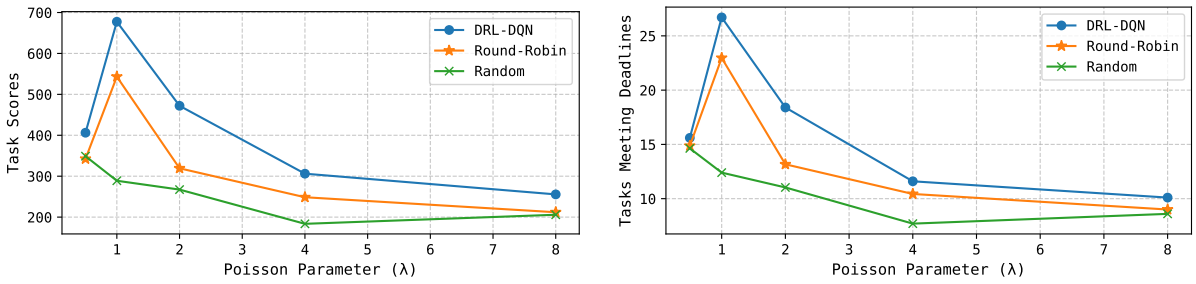


Figure 8.2: Convergence evaluation of DRL-DQN during training.

The resulting policy from the aforementioned training is evaluated against two baselines: *(i) RR,* a well-known and powerful scheduling baseline that optimizes resource utilization by allocating tasks in a cyclic manner. This approach ensures that all LLMs receive equitable task allocation; and *(ii) Random,* which selects LLMs randomly without any strategic consideration. We con-

duct experiments with the three approaches over 30 episodes while varying the task arrival rate over time using five different λ values: 0.5, 1, 2, 4, and 8. We record the average reward (sum of the task scores) across all 30 episodes. Additionally, we track the average number of tasks for which deadlines were satisfied across all 30 episodes. The results are shown in Fig. 8.3.

In subfigure 8.3(a), we present the average rewards achieved by each approach over the 30 episodes. The DRL-DQN agent consistently outperforms both the RR and Random methods, demonstrating its effectiveness in learning a superior scheduling policy. While the RR method shows stable performance, it does not reach the reward levels of the DRL-DQN agent. This is because the RR method ensures equitable routing among LLMs, but the varying generation times of the LLMs limit overall performance. The DRL-DQN agent surpasses RR by learning the token generation speeds of the LLMs and identifying faster models based on its experience, even when the set of LLMs is updated. In contrast, the Random method exhibits variability in performance due to its lack of strategic task allocation. On the other hand, subfigure 8.3(b) illustrates the average number of tasks for which deadlines were satisfied across the 30 episodes. Here, the DRL-DQN agent significantly outperforms the baselines in deadline satisfaction. For instance, for $\lambda = 1$, it achieves 89%, compared to 76% (RR) and 41% (Random). Similarly, for $\lambda = 2$, it achieves 61%, versus 43% (RR) and 36% (Random). The RR method performs reasonably well; however, it does not reach the performance level of the DRL-DQN agent. The Random method, in contrast, results in a notably lower rate of deadline satisfaction, highlighting its inefficiency in task scheduling.



(a) Task scores for different task arrival rates (λ). (b) Satisfied deadlines for different task arrival rates (λ).

Figure 8.3: Performance comparison among DRL-DQN, RR, and Random scheduling methods.

8.5.3 Discussions

Task scheduling for LLMs is a relatively new area that has recently gained attention in research. This is not a simple task that traditional heuristics can easily resolve in the rapidly evolving landscape of 6G networks. Therefore, as experimental results have shown, DRL-based solutions are effective in optimizing task scheduling for LLMs. It can learn the dynamic nature of the environment, such as task arrival patterns and LLMs token generation speed, and adapt to these factors to develop an effective scheduling policy. However, several considerations need to be addressed in future works. These include optimizing energy consumption, as LLMs require significant energy for inference and serving due to their computational demands. Additionally, strategies for balancing the use of Small Language Models (SLMs) at the far edge with larger LLMs at the edge or in the cloud should be considered. Although SLMs may have lower performance scores, they consume less energy and have faster execution times.

8.6 Conclusion

As 6G networks evolve to become more intelligent, the integration of LLMs will be crucial in achieving these capabilities. However, their computational demands necessitate efficient task scheduling strategies to optimize performance across multiple 6G use cases. To this end, this chapter introduced a DRL-based scheduler, aiming to maximize task scores while adhering to latency constraints. Our evaluations, conducted under real-world conditions, demonstrate that the proposed approach outperforms traditional scheduling methods such as RR and Random. These results highlight the potential of DRL in enhancing the efficiency and effectiveness of LLMs within 6G networks. Future research can focus on tuning the DRL agent to simultaneously optimize execution time, task scores, and energy consumption, as well as extending the DRL environment to incorporate SLMs at the far edge and LLMs at the edge and cloud.

At this stage of the thesis, we presented our contributions in designing GenAI-based systems for IBN. Part I addressed intent translation, Part II focused on intent assurance, and in this part, we tackled GenAI operations. In the next part, we summarize all contributions and identify future directions and potential research avenues.

Part IV

Conclusion and Perspectives

Chapter 9

Conclusion

This thesis has explored how GenAI, particularly LLMs, can fundamentally reshape IBN in 6G networks. Motivated by the growing complexity of managing heterogeneous infrastructures and the need for autonomous, intent-driven operations, we investigated three interconnected challenges: intent translation, intent assurance, and GenAI operations.

First, we addressed the challenge of intent translation, bridging the gap between high-level, human-centric intents and the technical configurations required to realize them across diverse network domains. We proposed LLM-enabled frameworks that allow users to express intents directly in natural language, eliminating the complexity of structured formats such as JSON or YAML. Our approach demonstrated that LLMs can accurately generate deployable service descriptors, significantly lowering the barrier for non-expert users. We then extended the framework to support multiple OSS-level tasks beyond service configuration, including configuration, assurance, and monitoring through APIs, forming the OSS-GPT framework. These contributions confirm the feasibility of using natural language as the primary medium for expressing and managing network intents.

Second, we examined the problem of intent assurance, ensuring that network behaviors remain aligned with user-defined objectives under dynamic conditions. Our research began with the design of an NWDAF-based architecture for real-time traffic anomaly detection, combining ML models with 3GPP-compliant microservices. Building on this, we proposed a trustworthy closed-loop assurance pipeline that integrates AI, XAI, and LLMs. This framework not only detects and resolves anomalies autonomously but also explains decisions and corrective actions in natural language, increasing transparency and operator trust. Our findings demonstrate that intent assurance can be achieved in a way that is both autonomous and interpretable, advancing the vision of ZSM.

Finally, we turned our attention to GenAI operations, recognizing that the effectiveness of LLMs in IBN depends on domain adaptation and efficient utilization. We introduced 5G INSTRUCT Forge, a data engineering pipeline for transforming complex 3GPP technical specifications into structured datasets, enabling the creation of Telecom-aware LLMs. These models significantly outperformed general-purpose LLMs in Telecom-specific tasks, confirming the necessity of domain specialization. To address scalability, we proposed a DRL-enabled scheduling framework that optimally routes tasks to shared LLMs under SLO constraints. This ensures efficient resource utilization without compromising performance, paving the way for practical deployment of LLMs in large-scale network management systems.

Taken together, the contributions of this thesis demonstrate that GenAI can serve as a cornerstone technology for 6G IBN. By enabling natural language interfaces, autonomous assurance, and scalable AI operations, our work brings intent-driven management closer to real-world deployment, reducing operational complexity and empowering vertical users to interact seamlessly with next-generation networks.

Chapter 10

Future Perspectives

10.1 Introduction

In this chapter, we explore future perspectives building on the work presented in this thesis, which investigates the integration of GenAI into IBN. We organize the chapter into two main sections, each addressing perspectives related to IBN and GenAI, respectively, while highlighting their interplay in network management and automation.

Regarding IBN, we discuss potential enhancements to LCM, including feasibility checks, intent conflict resolution, and dynamic negotiation mechanisms. We also consider federated and collaborative IBN for multi-operator environments, as well as security and resilience enhancements through zero-trust architectures.

Concerning GenAI, we explore three directions: sustainability and energy-efficient AI strategies for greener network operations; faster inference and decision-making to optimize LLM-based tasks; and the potential role of advanced Agentic AI and early Artificial General Intelligence (AGI) capabilities to support autonomous intent understanding, reasoning, and network orchestration at scale.

10.2 Future Directions in IBN

10.2.1 LCM Enhancements

Our work has demonstrated the feasibility of automatic intent decomposition and translation. However, achieving a fully robust and scalable E2E LCM system requires additional enhancements, particularly in feasibility checking, intent activation, and large-scale conflict resolution. One key challenge is that LLM-generated configurations may not always align with the available infrastructure resources, especially as the number and complexity of concurrent intents grow. To address this, an ML-based feasibility check should be integrated into the IBN-enabled management framework. Once an intent is deemed feasible, it can be automatically activated within the infrastructure. If the intent fails the feasibility check, an agent-based negotiation mechanism can be triggered to interact with users. Following our contributions, this negotiation can be facilitated through a chatbot interface (using GenAI), allowing users to resolve conflicts and adjust requests using natural language. Preliminary works, such as [96], have explored this direction, but further research is needed to develop scalable and fully autonomous negotiation mechanisms capable of handling large numbers of simultaneous intents. Moreover, intent

assurance mechanisms can continuously monitor the system to detect conflicts among already activated intents. As intent density increases, large-scale conflict resolution becomes a critical scalability challenge. In cases where multiple intents compete for shared resources, priority-aware mechanisms should determine precedence, while negotiation agents may resolve conflicts by coordinating with users or automatically adapting lower-priority intents. In addition, the growth of monitoring data generated by intent assurance functions introduces scalability challenges related to data collection, storage, and real-time analytics, which remain open research directions.

10.2.2 Federation and Collaboration

With 6G and beyond, multiple MNOs will increasingly need to federate their resources to efficiently provide services for advanced applications. While this thesis primarily focuses on scalability with respect to intent complexity and AI-driven management, cross-operator federation represents an additional and largely unexplored scalability dimension. Future IBN systems should therefore support collaborative intent handling, enabling the simultaneous deployment and configuration of services across multiple MNOs. This transforms the mobile ecosystem into a "network of networks," where operators interconnect their infrastructures to collectively fulfill user intents. A key capability for such collaborative IBN systems is the use of intelligent agents that can determine which MNOs are best suited to fulfill a given intent at scale. These agents would analyze the intent's requirements, such as QoS, latency, and geographic coverage, alongside the available resources, policies, and SLAs of different operators. By doing so, the system can automatically select the optimal combination of MNOs, plan the necessary configuration steps for each domain, and coordinate their execution in a consistent manner. Furthermore, federation introduces challenges related to monitoring data growth, cross-domain observability, and distributed decision-making. Leveraging shared insights and cross-operator learning to manage these challenges represents an important direction for future research. Investigating scalable, agent-based collaborative IBN mechanisms for multi-operator deployment is therefore essential to realize autonomous, resilient, and large-scale 6G networks.

10.2.3 Security and Resilience

Security will be a critical enabler for future IBN systems. Zero-trust architectures ensure that every intent, component, and transaction is verified before execution, eliminating implicit trust. In the context of 6G, adaptive zero-trust frameworks combined with AI-driven anomaly detection can provide continuous authentication, context-aware access control, and self-healing capabilities that automatically isolate or remediate threats. Integrating such mechanisms into the intent translation and execution pipeline will help prevent unauthorized actions, detect anomalies in real time, and strengthen intent assurance across disaggregated and dynamic infrastructures. Equally important is resilience, as systems must guarantee service continuity under failures or unexpected conditions. Techniques such as redundancy strategies, fallback mechanisms, predictive fault detection, and AI-assisted self-healing orchestration will be crucial. Future research should explore cross-layer automation, blockchain-assisted integrity, and federated AI for lightweight intrusion detection at the edge. By combining security and resilience with AI-enabled orchestration, systems can evolve into trustworthy, autonomous, and highly reliable network fabrics capable of sustaining dynamic and collaborative operations across multi-operator environments.

10.3 Future Directions in GenAI

10.3.1 Sustainability and Energy Efficiency

From Chapter 6, we observed that the best-performing open-source LLM for our use case has 70B parameters, which, while delivering strong performance, results in extremely high power consumption. This underlines a key limitation in relying solely on very large models for real-world network management deployments, where sustainability is as important as accuracy. A promising research direction is the development of SLMs with fewer parameters (e.g., around 1B), which aim to retain much of the efficiency and reasoning capacity of larger counterparts while drastically reducing energy consumption. Such models are inherently more lightweight, environmentally sustainable, and better suited for deployment at the edge or in resource-constrained environments. However, while progress in SLMs has been significant, they are not yet fully capable of replacing large models in terms of robustness and generalization, making continued research and investment in their evolution essential. Beyond reducing model size, the power consumption of LLMs can be further optimized through advanced energy-efficient techniques [166]. Approaches such as model pruning, quantization, knowledge distillation, and adaptive inference can substantially lower the computational footprint while maintaining task performance. Furthermore, integrating energy-awareness into RL frameworks used in AI-driven network operations, such as DRL,offers an opportunity to explicitly balance performance with sustainability objectives. This calls for a paradigm shift where models and management frameworks are co-designed not only for accuracy and scalability but also for green AI principles, ensuring that the future of autonomous 6G networks is both intelligent and environmentally responsible.

10.3.2 Inference and Decision-Making Optimization

One of the most critical limitations of LLMs for network automation is their long generation time, which becomes particularly problematic in low-level decision-making scenarios, such as URLLC, where near-zero latency is required. In such cases, even small delays can compromise service guarantees. While several approaches have been proposed in the literature to accelerate LLM inference [156], these techniques need to be adapted and validated for real-time network management, where both speed and reliability are essential. Another bottleneck lies in the reliance of current LLMs on extensive ICL, which further increases inference latency. Developing Telecom-aware LLMs offers a promising solution, as they can reduce the need for detailed contextual prompts and directly capture domain-specific reasoning. By embedding Telecom-specific knowledge into their training, these models can better understand anomalies and dependencies across heterogeneous technological domains, a task where generic LLMs often fail. Initial steps have been taken in this direction in Chapter 7 of this thesis; however, further research is needed to enhance their efficiency, domain specialization, and the reliability of their structured outputs. Our evaluations in Chapter 6 confirm a clear trade-off between efficiency and reasoning capability. While SLMs are more energy-efficient, their ability to perform reasoning remains limited. This highlights that reducing inference time cannot come at the cost of reasoning depth. Additionally, given the central role of LLMs in intent translation, service planning, and decision support, the risk of hallucinations, where models produce incorrect or inconsistent outputs, remains a concern. Although current mitigation mechanisms, such as validation agents, rule-based checks, and human-in-the-loop controls, help address these risks, further research should explore structured mitigation strategies to ensure trustworthy decision-making. Future work should therefore focus on enhancing SLMs reasoning capabilities, integrating structured reasoning frameworks, developing hybrid inference pipelines that combine low latency with ro-

bust reasoning, and systematically addressing hallucinations to support reliable, AI-native 6G network management.

10.3.3 Next-Generation Agentic AI Towards AGI

In modern 6G networks, intents will become increasingly complex, as autonomous management must support highly abstracted, multi-domain, and continuously evolving requirements. With this rising complexity, current LLM-based planning systems (e.g., Chapter 4) reveal significant limitations: as tasks grow longer and require multiple reasoning and decision-making steps, error rates and deviations from optimal execution tend to increase. This highlights a fundamental challenge for intent-driven orchestration, as today's LLMs remain largely reactive rather than proactive and are not yet fully capable of reliably executing multi-step reasoning in dynamic and large-scale network environments. To overcome these limitations, research is now moving towards Agentic AI, where models evolve from passive reasoning engines into autonomous agents capable of perceiving, planning, and acting within their environment. In telecom orchestration, Agentic AI can enable intent-driven systems that not only interpret user intents but also autonomously decompose them, coordinate multi-domain actions, verify execution outcomes, and iteratively improve their strategies through feedback. Achieving this vision requires integrating hierarchical intent decomposition, contextual cross-domain understanding, and robust multi-step reasoning mechanisms into the AI pipeline. Furthermore, new training and interaction paradigms are needed to support agentic behaviors, combining domain-specific knowledge, cross-domain correlations, and real-world operational feedback. RL methods, such as vinePPO [167], offer a promising foundation for enabling persistent learning loops, self-correction, and adaptive decision-making, essential for network-level autonomy. While transformer-based architectures dominate current research, emerging alternatives such as RWKV [168] and Structured State Space Models (SSMs) [169] could provide more efficient or specialized representations for agentic orchestration. Exploring these architectures in tandem with advanced reinforcement and planning techniques could lead to the emergence of telecom-oriented agentic systems, capable of executing workflows, reasoning over anomalies, and autonomously resolving conflicts within intent assurance frameworks. Ultimately, the evolution from LLM-based orchestration to Agentic AI marks a pivotal step towards AGI-like capabilities for networking, where systems can autonomously manage diverse, evolving, and mission-critical tasks with minimal supervision and near-zero error. Such progress will be fundamental to realizing real-time, scalable, and trustworthy intent orchestration in 6G and beyond.

Chapitre 11

Résumé en français

11.1 Contexte de la Thèse

Les réseaux cellulaires ont continuellement évolué au cours des dernières décennies, poussés par la nécessité de supporter une demande de trafic toujours croissante, de nouveaux types de services et des exigences de performance plus strictes. À la base de ces réseaux se trouve l'infrastructure physique et virtuelle, composée des réseaux d'accès radio (RANs), des réseaux de transport (TNs) et des coeurs de réseau (CNs) qui assurent la connectivité. Au-dessus de cette infrastructure, les frameworks de gestion de réseau sont responsables de l'orchestration, de l'allocation et de l'optimisation des ressources afin de garantir la continuité et l'efficacité des services. À chaque nouvelle génération, de la 2G à la 5G, la couche de gestion est devenue progressivement plus sophistiquée, passant d'une configuration manuelle et de politiques statiques vers l'automatisation et le contrôle piloté par logiciel [4]. Cette évolution prépare le terrain pour l'étape suivante : des frameworks de gestion autonomes et intelligents, nécessaires pour exploiter les infrastructures complexes et dynamiques des systèmes au-delà de la 5G et de la 6G.

L'évolution des réseaux de la 5G vers la 6G se caractérise non seulement par des débits plus élevés et une latence plus faible, mais également par un changement de paradigme vers des infrastructures réseau plus intelligentes, flexibles et autonomes. Alors que la 5G a introduit des catégories de services clés telles que le haut débit mobile amélioré (eMBB), les communications ultra-fiables à faible latence (URLLC) et les communications massives de type machine (mMTC), la 6G vise à étendre ces capacités et à les pousser à de nouveaux extrêmes [5]. En particulier, la 6G ambitionne de supporter des services de communication immersifs tels que la téléprésence holographique et la réalité étendue (XR), d'intégrer des fonctionnalités de communication et détection conjointes (JCAS), de permettre la création de jumeaux numériques à grande échelle des systèmes physiques et d'implémenter l'intelligence artificielle (AI) comme une fonctionnalité native du réseau [6]. De plus, la 6G devrait fournir une connectivité ubiquitaire dans des environnements divers, des zones urbaines denses aux régions éloignées et mal desservies, garantissant que les utilisateurs et les dispositifs puissent accéder de manière transparente à des services de haute qualité à tout moment et en tout lieu. Cette évolution souligne la complexité croissante des opérations réseau et met en évidence le besoin urgent de frameworks de gestion avancés capables de faire face à l'échelle, à l'hétérogénéité et au dynamisme sans précédent des infrastructures 6G [7].

Les futurs services 6G, tels que les communications holographiques, la XR, les systèmes autonomes et l'automatisation industrielle, exigent des garanties de performance strictes, une

résilience élevée et une qualité de service adaptative (QoS). Les frameworks de gestion traditionnels, principalement réactifs et basés sur des règles, ne sont pas adéquats pour soutenir la complexité et les exigences de tels services. La communauté converge plutôt vers des frameworks de gestion avancés, proactifs et pilotés par l'AI, capables de raisonner sur l'état du réseau, de prédire les anomalies et d'adapter les ressources de manière autonome [6]. Dans ce contexte, le réseau basée sur l'intention (IBN) a émergé comme un paradigme prometteur pour la gestion des réseaux. Plutôt que de configurer manuellement des paramètres bas niveau, les opérateurs et les verticales peuvent spécifier des intentions de haut niveau, telles que les SLOs ou des objectifs de performance [1]. Le système de gestion interprète ensuite ces intentions, les traduit en configurations exploitables et les applique à l'infrastructure sous-jacente. De manière cruciale, ce processus inclut également des mécanismes d'assurance continue qui vérifient si le réseau satisfait les intentions exprimées en temps réel, fournissant des boucles de rétroaction capables de déclencher des actions correctives en cas d'écarts [8]. Cette approche réduit non seulement la complexité opérationnelle, mais garantit également l'alignement avec les exigences métier tout en assurant le respect des engagements de performance et de fiabilité de bout en bout (E2E).

11.2 Motivation

À l'heure actuelle, les technologies d'intelligence artificielle générative (GenAI), telles que les grands modèles de langage (LLMs), permettant leur déploiement dans des scénarios complexes de gestion de réseau. Par conséquent, GenAI peut jouer un rôle central dans le soutien à la gestion autonome et basée sur les intentions dans les réseaux au-delà de la 5G et de la 6G. En tirant parti de la GenAI, les réseaux peuvent mieux interpréter des objectifs de haut niveau, s'adapter dynamiquement aux conditions changeantes et garantir que les services respectent des exigences strictes de performance et de fiabilité. La maturation de GenAI ouvre la voie à des implémentations pratiques et à grande échelle de frameworks basés sur les intentions qui étaient auparavant uniquement théoriques, ce qui rend l'exploration de ces technologies à la fois opportune et cruciale pour améliorer fondamentalement l'IBN [9].

Premièrement, la traduction des intentions vise à combler l'écart entre les intentions de haut niveau et les configurations techniques nécessaires à leur mise en œuvre au sein de l'infrastructure [1]. Bien que les standards définissent les intentions comme des constructions lisibles par l'homme, généralement exprimées dans des formats tels que JSON ou YAML pour faciliter la communication des objectifs, ces intentions restent complexes et nécessitent souvent une expertise pour être correctement formulées. Des erreurs d'interprétation ou de configuration peuvent entraîner des dégradations ou des échecs de service. Cela motive notre recherche pour proposer des mécanismes capables de simplifier l'expression des intentions et d'utiliser des techniques GenAI pour les traduire automatiquement en configurations réseau précises et bas niveau. Ainsi, nous visons à réduire considérablement la charge opérationnelle des opérateurs humains tout en maintenant précision, fiabilité et conformité aux objectifs exprimés.

Ensuite, l'assurance des intentions se concentre sur la vérification que le comportement du réseau respecte de manière constante les intentions exprimées et sur la détection et la résolution autonomes des anomalies [1]. La mise en place d'un système en boucle fermée capable de surveiller, d'analyser et de corriger les écarts sans intervention humaine est essentielle pour garantir que les objectifs de haut niveau soient atteints. Cela nécessite des frameworks capables de détecter les anomalies, d'en identifier les causes racines et de déclencher des actions correctives, tout en maintenant la transparence et la confiance pour les opérateurs humains. Cette problématique

motive notre recherche à explorer des approches pilotées par la GenAI permettant de garantir que la prise de décision autonome soit fiable et interprétable, renforçant la confiance dans ces systèmes tout en améliorant l'efficacité opérationnelle.

Enfin, ces approches pilotées par la GenAI soulignent que la GenAI est un facteur clé tant pour la traduction que pour l'assurance, mais elles introduisent également de nouveaux défis liés à la gestion des modèles GenAI. Le déploiement à grande échelle de la GenAI dans des opérations réseau critiques nécessite une attention particulière au cycle de vie des modèles GenAI, incluant l'entraînement, la mise à jour, la surveillance et l'intégration [10]. Sans une gestion efficace des modèles, les performances peuvent se dégrader, les décisions peuvent devenir peu fiables et l'évolutivité peut être compromise. Cela motive notre recherche à considérer des frameworks et des stratégies assurant que les modèles GenAI restent robustes, dignes de confiance et évolutifs, soutenant le fonctionnement fiable des réseaux 6G futurs.

11.3 Défis de la Thèse

Cette thèse explore l'avancement des frameworks IBN dans les réseaux 6G grâce à l'utilisation de la GenAI. Ces technologies ont récemment gagné en popularité et ont permis de simplifier des applications centrées sur l'humain qui utilisent le langage naturel, comme la traduction, la compréhension, l'analyse de sentiments ou la réponse à des questions. La traduction et l'assurance des intentions font partie de ces tâches, puisque notre objectif est d'exprimer les intentions réseau en langage naturel et de garantir leur exécution à l'aide de modèles GenAI, tout en expliquant les décisions en langage naturel. La Fig. 11.1 résume mon parcours doctoral.

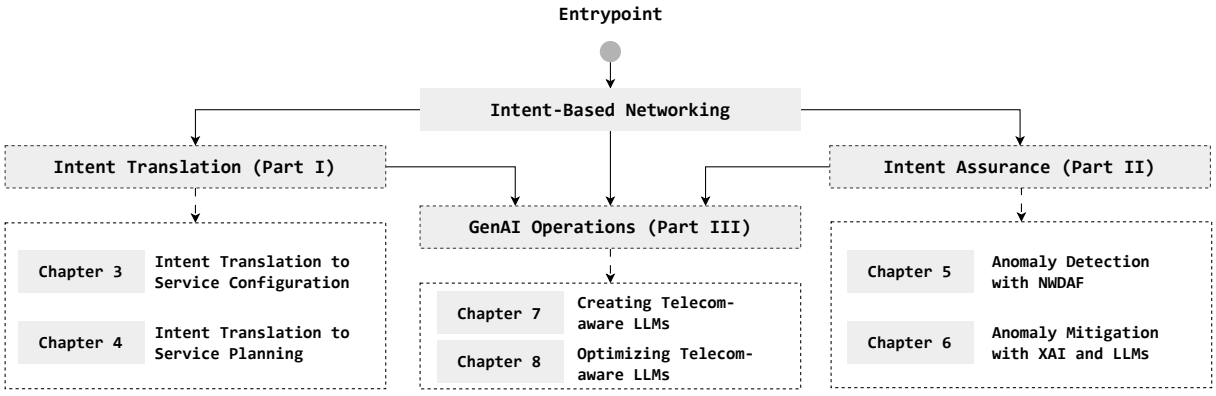


FIGURE 11.1 : Parcours doctoral.

La thèse aborde d'abord le problème de la traduction des intentions, qui devient plus complexe lorsqu'elles sont exprimées en langage naturel. Traduire des intentions en configurations structurées et bas niveau est difficile en raison de l'ambiguïté et de la flexibilité du langage humain. Pour y faire face, nous nous appuyons sur des LLMs, efficaces pour comprendre du texte non structuré et offrant une flexibilité totale dans la définition des intentions [11]. Nous nous concentrons initialement sur la configuration des services dans plusieurs domaines technologiques, incluant les RANs, CNs et Edge/Cloud, ce qui complique la traduction puisque chaque intention doit être appliquée sur plusieurs domaines. Ensuite, nous étendons la traduction au-delà de la configuration de services pour couvrir toutes les tâches de gestion, telles que la récupération de l'état des intentions, la configuration et la suppression de ressources sur plusieurs domaines. Cela

nécessite de décomposer les intentions de haut niveau en tâches exploitables au niveau des OSSs.

Ensuite, la thèse traite du problème de l'assurance des intentions, qui inclut la détection et la résolution autonome des anomalies. Nous commençons par la détection d'anomalies dans le trafic des UEs, avec une architecture combinant des modèles GenAI, comme les autoencodeurs, et NWDAF pour détecter des motifs de trafic anormaux [12]. Cette détection alimente les autres fonctions du système en boucle fermée, permettant l'assurance automatisée des intentions. Nous abordons ensuite la boucle complète d'assurance, consistant à détecter les anomalies, identifier leurs causes racines et les résoudre de manière autonome [13]. Ceci est réalisé via une combinaison d'AI, de XAI et de modèles GenAI, avec un accent particulier sur l'explication des décisions et des actions correctives en langage naturel grâce aux LLMs.

Enfin, pour améliorer la qualité et l'efficacité de la traduction et de l'assurance, nous montrons que des LLMs spécifiques au domaine sont nécessaires, notamment pour les réseaux 5G/6G. Pour cela, nous proposons un pipeline qui extrait et structure les standards 3GPP afin de créer des LLMs experts en télécommunications, surpassant les modèles génériques pour la compréhension d'intentions spécifiques au domaine. Étant donné les besoins GPU importants, utiliser un modèle distinct pour chaque tâche IBN n'est pas pratique. Nous proposons donc un mécanisme de routage qui dirige efficacement les tâches vers un ensemble de LLMs à l'aide du DRL, optimisant l'utilisation des ressources tout en maintenant des performances élevées pour l'IBN.

11.4 Contributions pour la traduction des intentions

11.4.1 Traduction de l'intention en configuration de service

Problème :

Les solutions IBN actuelles reposent sur des formats structurés tels que JSON ou YAML, souvent encapsulés dans des configurations bas niveau (ILIs), tel que les descripteurs de service réseau (NSDs) pour le domaine Edge/Cloud, conformes aux standards de l'ETSI [15]. Bien que ces formats assurent précision et interopérabilité, leur création reste complexe, chronophage et sujette aux erreurs, en particulier pour les utilisateurs non experts. La dépendance à ces spécifications techniques constitue une barrière à l'adoption, car les utilisateurs doivent maîtriser des détails techniques plutôt que de se concentrer sur leurs besoins métiers. Une évolution naturelle consiste à permettre la spécification d'intentions directement en langage naturel, supprimant ces barrières structurelles et rendant la technologie accessible à un public plus large [11]. Cependant, traduire des intentions en langage naturel en configurations exploitables devient encore plus complexe lorsqu'elles couvrent plusieurs domaines technologiques hétérogènes, tels que les RANs, CNs et Edge/Cloud. Les principaux défis incluent : (i) l'ambiguïté et la variabilité du langage, (ii) la décomposition des intentions de haut niveau en tâches exploitables et cohérentes à travers différents domaines, et (iii) la prise en charge complète du cycle de vie des intentions (LCM), incluant la traduction, la négociation, l'activation, la surveillance et l'assurance, de manière automatisée et harmonisée [16].

Solution proposée :

Pour relever ces défis, nous proposons une architecture LLM-centrée couvrant l'ensemble du cycle de vie des intentions multi-domaines. Le système exploite les capacités avancées des LLMs pour interpréter les intentions exprimées en langage naturel, les décomposer en tâches spécifiques

à chaque domaine technologique, négocier avec l'utilisateur si nécessaire pour clarifier les ambiguïtés, traduire ces tâches en ILIs, et activer les services sur les infrastructures. De plus, le système assure un suivi continu et la vérification de l'exécution des intentions afin de garantir leur conformité aux objectifs spécifiés. Une boucle d'apprentissage par retour humain (HF) permet au système de s'améliorer de manière itérative en capitalisant sur les retours et corrections des utilisateurs, enrichissant ainsi la base de connaissances (KB) sans nécessiter de réentraînement complet des modèles. Le prototype implémenté dans la 5G facility d'EURECOM [3] a démontré la faisabilité et l'efficacité de cette approche dans des scénarios réels multi-domaines, confirmant le potentiel des LLMs pour simplifier et automatiser la gestion de réseaux de prochaine génération tout en réduisant la complexité pour les utilisateurs finaux.

Démonstration :

Une démonstration en ligne présentant le prototype de traduction d'intentions est disponible[1].

Publications :

LLM-enabled Intent-Driven Service Configuration for Next-Generation Networks
Abdelkader Mekrache ; Adlen Ksentini
2024 IEEE 10th International Conference on Network Softwarization (NetSoft)

Intent-Based Management of Next-Generation Networks : an LLM-Centric Approach
Abdelkader Mekrache ; Adlen Ksentini ; Christos Verikoukis
IEEE Network, Volume 38, Issue 5, September 2024

11.4.2 Traduction de l'intention en planification de service

Problème :

La gestion des réseaux 6G multi-domaines impose des exigences considérables aux OSSs, qui doivent assurer une gestion E2E et coordonner des tâches hétérogènes à travers plusieurs domaines technologiques tels que le RAN, le CN, le TN et l'Edge/Cloud. Dans ce contexte, les utilisateurs doivent souvent formuler des intentions de haut niveau impliquant la coordination simultanée de ressources de calcul, de stockage et de réseau, tout en respectant des exigences strictes en matière de QoS et de QoE. Bien que l'approche basée sur les LLMs, présentée précédemment, ait démontré sa capacité à automatiser la configuration des services, elle se limite généralement à une seule fonctionnalité, comme la génération d'un unique appel API ou la configuration d'un service dans un domaine isolé. Les utilisateurs sont ainsi confrontés à plusieurs difficultés : (i) la nécessité de maîtriser de multiples endpoints API et leurs structures spécifiques, souvent hétérogènes selon les standards et les domaines ; (ii) la coordination manuelle de plusieurs appels API pour atteindre un objectif unique, ce qui augmente la probabilité d'erreurs et de délais ; (iii) l'absence de mécanismes intégrés pour superviser l'exécution, garantir l'assurance de service et gérer les anomalies de manière automatisée ; et (iv) la complexité d'adaptation aux nouvelles fonctionnalités ou aux évolutions des APIs. Ces limitations rendent l'interaction avec le système peu intuitive pour les utilisateurs non experts et freinent l'adoption d'une gestion IBN véritablement naturelle et fluide. Une solution est donc nécessaire pour simplifier l'interaction utilisateur, couvrir l'ensemble des tâches OSS et gérer de manière autonome le cycle de vie complet des intentions multi-domaines.

[1] `https://www.youtube.com/watch?v=SDyBge8WMt0`

Solution proposée :

Pour répondre à ces défis, nous proposons OSS-GPT, une architecture multi-agent basée sur des LLMs qui étend notre approche centrée sur la gestion des intentions afin de couvrir l'ensemble des tâches OSS multi-domaines. Le système interprète les intentions en langage naturel des utilisateurs, planifie et exécute de manière autonome une séquence d'appels API à travers tous les domaines 6G, génère les corps de requêtes et les paramètres nécessaires, et surveille l'exécution pour assurer l'accomplissement des objectifs et la garantie de service E2E. OSS-GPT repose sur quatre agents LLM spécialisés : l'*assistant*, qui sert d'interface avec l'utilisateur et transmet les intentions pertinentes au *planner* ; le *planner*, qui transforme les intentions en séquences d'appels API à exécuter ; l'*executor*, qui exécute chaque endpoint API conformément à la planification ; et le *reporter*, qui fournit un retour en langage naturel sur l'état et les résultats de l'exécution. Grâce à cette collaboration et à une planification hiérarchique, OSS-GPT s'adapte automatiquement aux nouvelles fonctionnalités et aux nouveaux endpoints API, nécessitant seulement la mise à jour des spécifications existantes. L'interface de type chatbot simplifie l'interaction utilisateur, masque la complexité des multiples APIs et réduit le besoin de compétences techniques approfondies. Des expérimentations réelles sur l'OSS d'EURECOM ont montré l'efficacité du système pour gérer l'ensemble des tâches OSS via le langage naturel, permettant ainsi une interaction IBN véritablement fluide et naturelle.

Démonstration :

Une démonstration en ligne présentant la gestion du réseau avec OSS-GPT est disponible[2].

Publications :

OSS-GPT : An LLM-Powered Intent-Driven Operations Support System for 5G Networks
Abdelkader Mekrache ; Adlen Ksentini ; Christos Verikoukis
2025 IEEE 11th International Conference on Network Softwarization (NetSoft)

Next-Generation 6G Network Management with OSS-GPT
Abdelkader Mekrache ; Adlen Ksentini ; Christos Verikoukis
Proceedings of the ACM SIGCOMM 2025 Posters and Demos, 158-160

11.5 Contributions pour l'assurance des intentions

11.5.1 Détection des anomalies avec NWDAF

Problème :

La détection d'anomalies constitue la première étape de l'assurance des intentions, qui consiste à identifier et résoudre les anomalies afin de satisfaire les attentes définies par les intentions. À cet effet, le NWDAF joue un rôle central dans les réseaux 5G/6G, en collectant, traitant et analysant des volumes massifs de données provenant de sources hétérogènes telles que les fonctions réseau (NFs), le RAN et l'infrastructure [12]. Parmi les cas d'usage critiques figure la détection d'anomalies, essentielle pour permettre aux systèmes d'assurance des intentions de résoudre automatiquement les problèmes sans intervention humaine. Cependant, la mise en œuvre pratique du NWDAF rencontre plusieurs défis majeurs : (i) la scalabilité, car les

[2] `https://www.youtube.com/watch?v=A1tTyHhyT80`

données réseau sont volumineuses et distribuées, nécessitant un traitement en temps réel ; (ii) la complexité des modèles ML requis pour analyser ces données et détecter des comportements anormaux précis ; (iii) la diversité des sources de données et des interfaces standardisées, rendant l'intégration et la collecte de données non triviales ; et (iv) le manque de prototypes opérationnels conformes aux standards 3GPP, limitant l'expérimentation de nouvelles méthodes [115, 117]. Ces défis rendent la détection d'anomalies réseau à la fois critique et difficile à réaliser dans des conditions réelles.

Solution proposée :

Nous proposons une architecture NWDAF basée sur des microservices, offrant flexibilité et extensibilité grâce à l'implémentation de chaque cas d'usage 3GPP comme un microservice indépendant et plug-and-play [19]. L'architecture est organisée en trois couches : (*i*) la couche *Exposure*, fournissant une interface nord conforme aux spécifications 3GPP via les openAPIs ; (*ii*) la couche *Analytics*, qui intègre des modèles ML pour analyser les données collectées et générer des analyses exploitables ; et (*iii*) la couche *Monitoring*, qui collecte les données des NFs via leurs interfaces standardisées, du RAN via des xApps O-RAN, et de l'infrastructure via le VIM. Pour le cas spécifique de la détection de trafic anormal des UEs, nous avons conçu un modèle autoencodeur LSTM, entraîné sur des données réelles du dataset Milano [20]. Ce modèle ML est déployé en tant que microservice dans le NWDAF et intégré à une plateforme 5G expérimentale OAI incluant CN et RAN [21]. Le système collecte en temps réel les données réseau réelles via des interfaces conformes 3GPP, applique l'analyse par l'autoencodeur et détecte avec précision les motifs de trafic anormaux générés par de vrais UEs. Les résultats expérimentaux démontrent la capacité du NWDAF à fournir des services analytiques fiables, flexibles et évolutifs, tout en respectant les standards 3GPP et en facilitant l'intégration de nouveaux cas d'usage.

Démonstration et ressources open source :

Une démonstration en ligne présentant la détection d'anomalies de trafic UE avec le NWDAF est disponible[3]. De plus, le code source complet du NWDAF a été rendu publiquement disponible[4].

Publication :

Combining Network Data Analytics Function and Machine Learning for Abnormal Traffic Detection in Beyond 5G
Abdelkader Mekrache ; Karim Boutiba ; Adlen Ksentini
GLOBECOM 2023 - 2023 IEEE Global Communications Conference

11.5.2 Atténuation des anomalies avec XAI et LLMs

Problème :

L'assurance des anomalies inclut la détection et la résolution d'anomalies sans intervention humaine. Elle est également désignée par le terme ZSM, qui constitue un pilier fondamental des réseaux 6G en visant une automatisation complète des processus de gestion [13]. Pour atteindre cet objectif, les systèmes de gestion de réseau (NMSs) s'appuient sur des modèles AI/ML capables de détecter et de résoudre automatiquement les anomalies. Toutefois, ces modèles présentent plusieurs limitations critiques : (i) leur nature de "boîtes noires", rendant leurs

[3] `https://www.youtube.com/watch?v=kI9GuJeW0es`
[4] `https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-nwdaf`

décisions opaques et difficilement interprétables par les opérateurs ; (ii) l'absence de mécanismes intégrés permettant d'expliquer de manière compréhensible les causes racines des anomalies ; et (iii) la difficulté pour les utilisateurs non experts d'interagir avec les résultats, ce qui réduit la confiance et limite l'adoption de ces systèmes. Ce déficit d'explicabilité et de transparence pose un problème majeur de fiabilité, de responsabilité et de confiance, notamment dans des environnements où les décisions prises par l'AI peuvent impacter directement la performance des services, comme dans la gestion des violations de SLA liées à la latence, au débit ou à l'allocation des ressources. Le défi central consiste donc à concevoir des solutions ZSM autonomes combinant efficacité opérationnelle et interprétabilité, afin de garantir la fiabilité et l'acceptabilité de la gestion réseau pilotée par l'AI.

Solution proposée :

Pour relever les défis mentionnés précédemment, nous étendons le travail de [23] en ajoutant une brique LLM après les modules AI et XAI, aboutissant à un pipeline inédit composé de AI|XAI|LLMs, conçu pour permettre un ZSM fiable. Premièrement, le module AI (XGBoost [24]) est chargé de détecter les anomalies, par exemple les violations de latence d'un SLA dans une application cloud-native. Deuxièmement, le module XAI (SHAP [25]) identifie la cause racine de l'anomalie en attribuant une importance quantitative aux différentes caractéristiques réseau (CPU, RAM, bande passante, etc.). Troisièmement, un LLM (Llama2 [26]) convertit ces explications numériques en explications accessibles en langage naturel, renforçant ainsi la transparence et la compréhension pour l'opérateur humain. En plus de fournir des explications, le LLM est également capable de proposer, voire d'exécuter directement, des actions correctives adaptées (par exemple, le réajustement des ressources CPU/RAM d'un microservice afin de restaurer le SLA). Cette combinaison permet d'allier précision de la détection, interprétabilité des décisions et autonomie opérationnelle. L'efficacité du pipeline est illustrée à travers un cas d'usage réel de mise à l'échelle dynamique de ressources dans un cluster Edge/Cloud, démontrant la capacité du système à résoudre de manière autonome des anomalies tout en fournissant des explications compréhensibles et dignes de confiance.

Démonstration :

Une démonstration en ligne présentant la détection et la résolution d'anomalies avec le pipeline proposé est disponible[5].

Publication :

On Combining XAI and LLMs for Trustworthy Zero-Touch Network and Service Management in 5G
Abdelkader Mekrache ; Mohamed Mekki ; Adlen Ksentini ; Bouziane Brik ; Christos Verikoukis
IEEE Communications Magazine, Volume 63, Issue 4, April 2025

[5] `https://www.youtube.com/watch?v=1CoDNVWJcqA`

11.6 Contributions pour les opérations GenAI dans l'IBN

11.6.1 Construction des LLMs spécialisés télécom

Problème :

Les LLMs ont montré des capacités impressionnantes pour comprendre et générer du texte généraliste, mais leur efficacité dans des domaines spécialisés tels que les télécommunications reste limitée. Les réseaux 5G et 6G reposent sur des standards très techniques, tels que les TSs du 3GPP, qui contiennent des informations denses sur les protocoles, la configuration des services et les exigences de SLA. Ces documents sont souvent non structurés, référencent d'autres documents ou paragraphes, et utilisent des formats complexes (figures, tableaux, annexes), rendant leur traitement automatique très difficile. Par conséquent, l'entraînement ou le fine-tuning des LLMs directement sur ces sources sans transformation préalable est inefficace, ce qui limite leur capacité à répondre correctement aux requêtes liées aux réseaux et à assister les développeurs ou opérateurs télécoms dans leurs tâches. L'absence de datasets structurés spécialisés constitue donc un goulot d'étranglement majeur pour la création de LLMs experts en Telecom, capables de comprendre et générer des informations fiables sur les réseaux mobiles.

Solution proposée :

Pour résoudre ce problème, nous introduisons 5G INSTRUCT Forge, un pipeline d'ingénierie des données avancé qui transforme les TSs du 3GPP en datasets structurés et optimisés pour l'entraînement des LLMs. Ce pipeline inclut plusieurs étapes clés : collecte et sélection des TSs pertinents, nettoyage et homogénéisation des documents, et génération de paires prompt/réponse exploitables par les LLMs. Grâce à l'utilisation de modèles LLMs existants pour structurer et enrichir les données, le pipeline produit un dataset cohérent et prêt pour le fine-tuning. À titre de preuve de concept, nous avons créé le dataset OAI-Instruct à partir de 22 TSs du 3GPP, fournissant une base de données riche et structurée pour la formation de LLMs spécialisés en Telecom. Ces modèles fine-tunés, à partir de datasets produits par le pipeline, ont démontré des performances supérieures à celles de GPT-4 pour des tâches spécifiques aux Telecom, telles que l'analyse de protocoles, la génération de configurations réseau et la réponse à des questions techniques complexes [28]. Ce pipeline illustre ainsi la faisabilité et l'efficacité d'une approche systématique pour créer des LLMs experts en Telecom, comblant le fossé entre les standards télécoms complexes et les capacités d'assistance intelligente basées sur LLMs.

Ressources open source :

Le code source complet de 5G INSTRUCT Forge a été rendu publiquement disponible[6]. De plus, la dataset OAI-Instruct est disponible en ligne[7].

Publication :

5G INSTRUCT Forge : An Advanced Data Engineering Pipeline for Making LLMs Learn 5G
Azzedine Idir Ait Said ; Abdelkader Mekrache ; Karim Boutiba ; Kostas Ramantas ; Adlen Ksentini ; Moufida Rahmani

[6] `https://gitlab.eurecom.fr/netsoft/5g-instruct-forge`
[7] `https://huggingface.co/datasets/Netsoft/oai-instruct`

IEEE Transactions on Cognitive Communications and Networking, Volume 11, Issue 2, April 2025

11.6.2 Optimisation des LLMs spécialisés télécom

Problème :

L'intégration des LLMs dans les réseaux 6G apporte de nouvelles capacités, notamment le raisonnement autonome, la génération de code et l'interprétation de requêtes en langage naturel. Ces fonctionnalités facilitent la gestion de services avancés tels que l'IBN et le ZSM. Cependant, le déploiement pratique de ces modèles rencontre plusieurs obstacles majeurs. Tout d'abord, le coût computationnel élevé des LLMs nécessite d'importantes ressources GPU et CPU, ainsi que de la mémoire. Les réseaux 6G ne peuvent pas se permettre de dédier un modèle par tâche, ce qui impose le partage d'un même modèle entre plusieurs applications et utilisateurs. Ensuite, tous les LLMs ne sont pas équivalents selon le type de tâche. Certains modèles excellent dans le raisonnement complexe, tandis que d'autres sont plus efficaces pour la génération de texte ou la traduction de standards 3GPP. Le choix du modèle pour chaque tâche influence directement la performance et le respect des délais. Par ailleurs, les tâches arrivent de manière aléatoire avec des priorités et types différents, rendant les méthodes de planification statiques, comme le round-robin ou la répartition aléatoire, inadaptées. Enfin, chaque tâche doit respecter un délai strict et atteindre un score minimal de performance, imposant une optimisation multi-objectif simultanée : maximiser la performance tout en minimisant la latence. En résumé, la planification des tâches LLM dans un réseau 6G partagé constitue un problème complexe avec des contraintes dynamiques et une incertitude sur l'impact des décisions en temps réel.

Solution proposée :

Pour relever ces défis, nous proposons un cadre de planification basé sur le DRL capable de gérer la distribution dynamique des tâches sur plusieurs LLMs partagés. L'agent DRL observe en temps réel l'état du réseau, notamment la charge des files d'attente de chaque LLM, le type de tâche et la date limite SLO pour chaque nouvelle tâche. Sur la base de ces informations, il décide quel LLM doit traiter chaque tâche, en tenant compte de ses capacités et de l'état actuel du réseau, afin de maximiser la probabilité de succès et de minimiser la latence. La fonction de récompense est formulée pour combiner la performance des tâches et le respect des délais, guidant l'agent vers un compromis optimal entre score et latence. Une fois entraîné, l'agent DRL peut gérer différents types de tâches et des arrivées dynamiques sans nécessiter de réentraînement, offrant ainsi une solution autonome et scalable. Pour valider cette approche, nous avons déployé quatre LLMs open-source sur deux GPUs et simulé les arrivées de tâches selon une loi de Poisson. Les résultats expérimentaux montrent que le planificateur DRL dépasse les méthodes classiques, notamment en termes de taux de réussite des tâches, respect des délais et efficacité globale du réseau. Ainsi, ce framework DRL permet une allocation dynamique et optimale des LLMs pour les réseaux 6G, garantissant des performances élevées tout en respectant les contraintes temporelles strictes imposées par les SLO.

Publication :

DRL-enabled SLO-aware Task Scheduling for Large Language Models in 5G Networks
Abdelkader Mekrache ; Adlen Ksentini ; Christos Verikoukis
IEEE ICC - 2025 IEEE International Conference on Communications

11.7 Conclusion

Cette thèse a exploré comment la GenAI, et en particulier les LLMs, peuvent transformer fondamentalement l'IBN dans les réseaux 6G. Motivés par la complexité croissante de la gestion d'infrastructures hétérogènes et par le besoin d'opérations autonomes orientées par l'intention, nous avons étudié trois défis interconnectés : la traduction des intentions, l'assurance des intentions et les opérations basées sur la GenAI.

Tout d'abord, nous avons abordé le défi de la traduction des intentions, en comblant le fossé entre les intentions de haut niveau, et les configurations techniques nécessaires pour les réaliser. Nous avons proposé des frameworks basés sur les LLMs permettant aux utilisateurs d'exprimer directement leurs intentions en langage naturel, éliminant ainsi la complexité des formats structurés tels que JSON ou YAML. Notre approche a montré que les LLMs peuvent générer avec précision des descripteurs de services déployables, réduisant la barrière pour les utilisateurs non experts. Nous avons ensuite élargi la portée pour englober l'ensemble du cycle de vie des intentions, y compris la configuration, l'assurance et la supervision via les APIs OSS. Ces contributions confirment la faisabilité de l'utilisation du langage naturel comme moyen principal pour exprimer et gérer les intentions réseau.

Ensuite, nous avons étudié le problème de l'assurance des intentions, garantissant que les comportements réseau restent alignés avec les objectifs définis par l'utilisateur dans des conditions dynamiques. Nos recherches ont débuté par la conception d'une architecture basée sur NWDAF pour la détection d'anomalies de trafic en temps réel, combinant des modèles ML avec des microservices conformes aux standards 3GPP. Sur cette base, nous avons proposé un pipeline d'assurance en boucle fermée fiable intégrant AI, XAI et les LLMs. Ce framework permet non seulement de détecter et résoudre les anomalies de manière autonome, mais aussi d'expliquer les décisions et les actions correctives en langage naturel, renforçant la transparence et la confiance des opérateurs. Nos résultats montrent que l'assurance des intentions peut être réalisée de manière autonome tout en restant interprétable, faisant progresser la vision du ZSM.

Enfin, nous nous sommes intéressés aux opérations basées sur la GenAI, reconnaissant que l'efficacité des LLMs dans l'IBN dépend de l'adaptation au domaine et de l'utilisation efficace des ressources. Nous avons présenté un pipeline d'ingénierie des données destiné à transformer les TSs du 3GPP en datasets structurés, permettant la création de LLMs experts en Telecom. Ces modèles ont largement surpassé les LLMs généralistes pour les tâches spécifiques aux télécoms, confirmant la nécessité d'une spécialisation par domaine. Pour répondre aux enjeux de scalabilité, nous avons proposé un framework de planification basé sur DRL, qui oriente de manière optimale les tâches vers des LLMs partagés sous contrainte de SLO. Cela garantit une utilisation efficace des ressources sans compromettre les performances, ouvrant la voie à un déploiement pratique des LLMs dans les systèmes de gestion réseau à grande échelle.

Pris dans leur ensemble, les contributions de cette thèse démontrent que la GenAI peut constituer une technologie clé pour l'IBN. En permettant des interfaces en langage naturel, une assurance autonome et des opérations d'AI à grande échelle, nos travaux rapprochent la gestion orientée par l'intention d'un déploiement réel, réduisant la complexité opérationnelle et permettant aux utilisateurs d'interagir de manière fluide avec les réseaux de prochaine génération.

Bibliography

[1] Engin Zeydan and Yekta Turk. "Recent advances in intent-based networking: A survey". In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE. 2020, pp. 1–5.

[2] Yupeng Chang et al. "A survey on evaluation of large language models". In: *arXiv preprint arXiv:2307.03109* (2023).

[3] Sagar Arora et al. "A 5G Facility for Trialing and Testing Vertical Services and Applications". In: *IEEE Internet of Things Magazine* (2022).

[4] Qazi Kamal Ud Din Arshad, Ahsan Ullah Kashif, and Ijaz Mansoor Quershi. "A review on the evolution of cellular technologies". In: *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. IEEE. 2019, pp. 989–993.

[5] Zoran Bojkovic et al. "6G ultra-low latency communication in future mobile XR applications". In: *Advances in Signal Processing and Intelligent Recognition Systems: 6th International Symposium, SIRS 2020, Chennai, India, October 14–17, 2020, Revised Selected Papers 6*. Springer. 2021, pp. 302–312.

[6] Muhammad K Shehzad et al. "Artificial intelligence for 6G networks: Technology advancement and standardization". In: *IEEE Vehicular Technology Magazine* 17.3 (2022), pp. 16–25.

[7] Estefania Coronado et al. "Zero touch management: A survey of network automation solutions for 5G and 6G networks". In: *IEEE Communications Surveys & Tutorials* 24.4 (2022), pp. 2535–2578.

[8] ETSI ISG ZSM. *Zero-touch Network and Service Management (ZSM); Intent-driven Autonomous Networks; Generic Aspects*. Tech. rep. GR ZSM 011 V2.1.1. ETSI, 2024. URL: `https://www.etsi.org/deliver/etsi_gr/ZSM/001_099/011/02.01.01_60/gr_ZSM011v020101p.pdf`.

[9] 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.

[10] Satyadhar Joshi. "LLMOps, AgentOps, and MLOps for Generative AI: A Comprehensive Review". In: (2025).

[11] Jieyu Lin et al. "AppleSeed: Intent-Based Multi-Domain Infrastructure Management via Few-Shot Learning". In: *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*. IEEE. 2023, pp. 539–544.

[12] 3GPP. *Technical Specification Group Core Network and Terminals; 5G System; Network Data Analytics Services;* tech. rep. 29.520. Version 17.10.0. 2023.

[13] ETSI ISG ZSM. *Zero-touch Network and Service Management (ZSM); Intent-driven Closed Loops*. Tech. rep. GS ZSM 016 V1.1.1. ETSI, 2024. URL: `https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/016/01.01.01_60/gs_ZSM016v010101p.pdf`.

[14] Kai Arulkumaran et al. "Deep reinforcement learning: A brief survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[15] ETSI. *Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; Network Service Descriptor File Structure Specification*. ETSI Group Specification GS NFV-SOL 007. 2019.

[16] Yanbo Song et al. "Full-life cycle intent-driven network verification: Challenges and approaches". In: *IEEE Network* 37.5 (2022), pp. 145–153.

[17] Anwer Al-Dulaimi, Xianbin Wang, and I Chih-Lin. *5G networks: fundamental requirements, enabling technologies, and operations management*. John Wiley & Sons, 2018.

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

[19] Kapil Bakshi. "Microservices-based software architecture and approaches". In: *2017 IEEE aerospace conference*. IEEE. 2017, pp. 1–8.

[20] Gianni Barlacchi et al. "A multi-source dataset of urban life in the city of Milan and the Province of Trentino". In: *Scientific Data* 2 (Oct. 2015), p. 150055. DOI: 10.1038/sdata.2015.55.

[21] Florian Kaltenberger et al. "OpenAirInterface: Democratizing innovation in the 5G Era". In: *Computer Networks* 176 (2020), p. 107284.

[22] David Gunning et al. "XAI—Explainable artificial intelligence". In: *Science robotics* 4.37 (2019), eaay7120.

[23] Mohamed Mekki et al. "XAI-Enabled Fine Granular Vertical Resources Autoscaler". In: *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*. IEEE. 2023, pp. 161–169.

[24] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[25] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017).

[26] Hugo Touvron et al. "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (2023).

[27] Nhu-Ngoc Dao et al. "A review on new technologies in 3GPP standards for 5G access and beyond". In: *Computer Networks* (2024), p. 110370.

[28] Josh Achiam et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

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

[30] Yiming Wei, Mugen Peng, and Yaqiong Liu. "Intent-based networks for 6G: Insights and challenges". In: *Digital Communications and Networks* 6.3 (2020), pp. 270–280.

[31] Lina Bariah et al. "Large Language Models for Telecom: The Next Big Thing?" In: *arXiv preprint arXiv:2306.10249* (2023).

[32] Zakria Qadir et al. "Towards 6G Internet of Things: Recent advances, use cases, and open challenges". In: *ICT express* 9.3 (2023), pp. 296–312.

[33] Chathurika Ranaweera et al. "4G to 6G: Disruptions and drivers for optical access". In: *Journal of Optical Communications and Networking* 14.2 (2022), A143–A153.

[34] Marco Giordani et al. "Toward 6G networks: Use cases and technologies". In: *IEEE communications magazine* 58.3 (2020), pp. 55–61.

[35] O-RAN Alliance. *O-RAN: Towards an Open and Smart RAN*. 2018. URL: https://www.o-ran.org/.

[36] Gabrial Brown et al. "Service-based architecture for 5g core networks". In: *Huawei White Paper* 1 (2017).

[37] Meghna Khaturia et al. "Service-based architecture evolution: Towards enhanced signaling in beyond 5G/6g networks". In: *2024 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2024, pp. 1–6.

[38] Sumbal Malik et al. "Implanting Intelligence in 5G Mobile Networks—A Practical Approach". In: *Electronics* 11.23 (2022), p. 3933.

[39] Liqiang Zhao et al. "Open-source multi-access edge computing for 6G: Opportunities and challenges". In: *IEEE Access* 9 (2021), pp. 158426–158439.

[40] Yushan Siriwardhana et al. "AI and 6G security: Opportunities and challenges". In: *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE. 2021, pp. 616–621.

[41] Luis Velasco et al. "End-to-end intent-based networking". In: *IEEE communications Magazine* 59.10 (2021), pp. 106–112.

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

[43] Alexander Clemm et al. *Intent-Based Networking - Concepts and Definitions*. 2022. URL: https://www.rfc-editor.org/info/rfc9315.

[44] TM Forum. *Intent Toolkit*. URL: https://www.tmforum.org/toolkits/intent/.

[45] Benoît Claise et al. *Service Assurance for Intent-Based Networking Architecture*. 2023. URL: https://www.rfc-editor.org/info/rfc9417.

[46] Benoît Claise et al. *A YANG Data Model for Service Assurance*. 2023. URL: https://www.rfc-editor.org/info/rfc9418.

[47] TM Forum. *Intent Common Model v3.6.0 (TR290)*. 2024. URL: https://www.tmforum.org/resources/introductory-guide/intent-common-model-v3-6-0-tr290/.

[48] TM Forum. *Intent Management API TMF921-v5.0*. 2024. URL: https://www.tmforum.org/oda/open-apis/directory/intent-management-api-tmf921/v5-0/.

[49] Feng-Hsiung Hsu. *Behind Deep Blue: Building the computer that defeated the world chess champion*. Princeton University Press, 2022.

[50] Reha Kılıçhan and Mustafa Yılmaz. "Artificial intelligence and robotic technologies in tourism and hospitality industry". In: *Erciyes Üniversitesi Sosyal Bilimler Enstitüsü Dergisi* 50 (2020), pp. 353–380.

[51] Ravil Muhamedyev. "Machine learning methods: An overview". In: *Computer modelling & new technologies* 19.6 (2015), pp. 14–29.

[52] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Supervised learning". In: *Machine learning techniques for multimedia: case studies on organization and retrieval*. Springer, 2008, pp. 21–49.

[53] M Emre Celebi and Kemal Aydin. *Unsupervised learning algorithms*. Vol. 1. Springer, 2016.

[54] Abdelkader Mekrache et al. "Deep reinforcement learning techniques for vehicular networks: Recent advances and future trends towards 6G". In: *Vehicular Communications* 33 (2022), p. 100398.

[55] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3 (1992), pp. 279–292.

[56] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3 (1992), pp. 229–256.

[57] Samira Pouyanfar et al. "A survey on deep learning: Algorithms, techniques, and applications". In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–36.

[58] Samanwoy Ghosh-Dastidar and Hojjat Adeli. "Spiking neural networks". In: *International journal of neural systems* 19.04 (2009), pp. 295–308.

[59] Zewen Li et al. "A survey of convolutional neural networks: analysis, applications, and prospects". In: *IEEE transactions on neural networks and learning systems* 33.12 (2021), pp. 6999–7019.

[60] Yong Yu et al. "A review of recurrent neural networks: LSTM cells and network architectures". In: *Neural computation* 31.7 (2019), pp. 1235–1270.

[61] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[62] Yuxi Li. "Deep reinforcement learning: An overview". In: *arXiv preprint arXiv:1701.07274* (2017).

[63] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[64] Dor Bank, Noam Koenigstein, and Raja Giryes. "Autoencoders". In: *Machine learning for data science handbook: data mining and knowledge discovery handbook* (2023), pp. 353–374.

[65] Rudresh Dwivedi et al. "Explainable AI (XAI): Core ideas, techniques, and solutions". In: *ACM computing surveys* 55.9 (2023), pp. 1–33.

[66] Jürgen Dieber and Sabrina Kirrane. "Why model why? Assessing the strengths and limitations of LIME". In: *arXiv preprint arXiv:2012.00093* (2020).

[67] David Foster. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play.* 2nd. O'Reilly Media, 2023. ISBN: 978-1098134181.

[68] Jing Xiong et al. "Autoregressive models in vision: A survey". In: *arXiv preprint arXiv:2411.05902* (2024).

[69] Gokul Yenduri et al. "Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions". In: *arXiv preprint arXiv:2305.10435* (2023).

[70] Aaron Van Den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* 12 (2016), p. 1.

[71] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. "Normalizing flows: An introduction and review of current methods". In: *IEEE transactions on pattern analysis and machine intelligence* 43.11 (2020), pp. 3964–3979.

[72] Yann LeCun et al. "A tutorial on energy-based learning". In: *Predicting structured data* 1.0 (2006).

[73] Florinel-Alin Croitoru et al. "Diffusion models in vision: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* 45.9 (2023), pp. 10850–10869.

[74] Gary Marcus, Ernest Davis, and Scott Aaronson. "A very preliminary analysis of DALL-E 2". In: *arXiv preprint arXiv:2204.13807* (2022).

[75] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[76] Gregory Grefenstette. "Tokenization". In: *Syntactic wordclass tagging*. Springer, 1999, pp. 117–133.

[77] Yangyi Chen et al. "Scaling laws for predicting downstream performance in llms". In: *arXiv preprint arXiv:2410.08527* (2024).

[78] Wayne Xin Zhao et al. "A survey of large language models". In: *arXiv preprint arXiv:2303.18223* (2023).

[79] Colin Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *Journal of machine learning research* 21.140 (2020), pp. 1–67.

[80] Mike Lewis et al. "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension". In: *arXiv preprint arXiv:1910.13461* (2019).

[81] Aohan Zeng et al. "Glm-130b: An open bilingual pre-trained model". In: *arXiv preprint arXiv:2210.02414* (2022).

[82] Yi Tay et al. "Transcending scaling laws with 0.1% extra compute". In: *arXiv preprint arXiv:2210.11399* (2022).

[83] Jie Huang and Kevin Chen-Chuan Chang. "Towards reasoning in large language models: A survey". In: *arXiv preprint arXiv:2212.10403* (2022).

[84] Tim Pearce and Jinyeop Song. "Reconciling Kaplan and Chinchilla scaling laws". In: *arXiv preprint arXiv:2406.12907* (2024).

[85] Zhichao Wang et al. "A comprehensive survey of llm alignment techniques: Rlhf, rlaif, ppo, dpo and more". In: *arXiv preprint arXiv:2407.16216* (2024).

[86] Banghao Chen et al. "Unleashing the potential of prompt engineering in large language models: a comprehensive review". In: *arXiv preprint arXiv:2310.14735* (2023).

[87] IBM Developer. *Preparing Data for Fine-Tuning LLMs*. `https://developer.ibm.com/learningpaths/get-started-data-prep-kit/dpk-llm-applications/dpk-prepare-data-fine-tuning-llms/`.

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

[89] Qingxiu Dong et al. "A survey on in-context learning". In: *arXiv preprint arXiv:2301.00234* (2022).

[90] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive NLP tasks". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 9459–9474.

[91] Juyong Jiang et al. *A Survey on Large Language Models for Code Generation*. 2024. arXiv: `2406.00515 [cs.CL]`. URL: `https://arxiv.org/abs/2406.00515`.

[92] Bingxi Zhao et al. *LLM-based Agentic Reasoning Frameworks: A Survey from Methods to Scenarios*. 2025. arXiv: `2508.17692 [cs.AI]`. URL: `https://arxiv.org/abs/2508.17692`.

[93] Xiang Chen et al. *An Empirical Study on Challenges for LLM Application Developers*. 2024. arXiv: `2408.05002 [cs.AI]`. URL: `https://arxiv.org/abs/2408.05002`.

[94] Baptiste Rozière et al. "Code Llama: Open Foundation Models for Code". In: *arXiv preprint arXiv:2308.12950* (2023).

[95] Celso H Cesila et al. "Chat-IBN-RASA: Building an Intent Translator for Packet-Optical Networks based on RASA". In: *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*. IEEE. 2023, pp. 534–538.

[96] Yogesh Sharma et al. "Intent Negotiation Framework for Intent-Driven Service Management". In: *IEEE Communications Magazine* 61.6 (2023), pp. 73–79.

[97] Xiaoang Zheng and Aris Leivadeas. "Network assurance in intent-based networking data centers with machine learning techniques". In: *2021 17th International Conference on Network and Service Management (CNSM)*. IEEE. 2021, pp. 14–20.

[98] Jingyu Wang et al. "Network Meets ChatGPT: Intent Autonomous Management, Control and Operation". In: *Journal of Communications and Information Networks* 8.3 (2023), pp. 239–255.

[99] Chao Feng, Xinyu Zhang, and Zichu Fei. "Knowledge solver: Teaching llms to search for domain knowledge from knowledge graphs". In: *arXiv preprint arXiv:2309.03118* (2023).

[100] Ali Maatouk et al. "Teleqna: A benchmark dataset to assess large language models telecommunications knowledge". In: *arXiv preprint arXiv:2310.15051* (2023).

[101] Abdelkader Mekrache, Adlen Ksentini, and Christos Verikoukis. "Machine Reasoning in FCAPS: Towards Enhanced Beyond 5G Network Management". In: *IEEE Communications Surveys & Tutorials* (2024).

[102] ETSI. *Zero Touch Network and Service Management (ZSM) Means of Automation*. 2018.

[103] 3rd Generation Partnership Project (3GPP). *Study on Artificial Intelligence (AI) and Machine Learning (ML) for NR Mobility*. Tech. rep. TR 38.744. Release 19. 3GPP, 2024.

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

[105] 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.

[106] OpenAPI Initiative. *OpenAPI Specification Version 3.1.0*. `https://github.com/OAI/OpenAPI-Specification`. Accessed: 2024-12-03. 2021.

[107] 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.

[108] Arthur S Jacobs et al. "Hey, lumi! using natural language for {intent-based} network management". In: *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 2021, pp. 625–639.

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

[110] Yongliang Shen et al. "Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face". In: *Advances in Neural Information Processing Systems* 36 (2024).

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

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

[113] Mohamed Mekki, Sagar Arora, and Adlen Ksentini. "A scalable monitoring framework for network slicing in 5g and beyond mobile networks". In: *IEEE Transactions on Network and Service Management* 19.1 (2021), pp. 413–423.

[114] 3GPP. *Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS)*. Tech. rep. 23.288. Version 17.8.0. 2023.

[115] Dimitrios Michael Manias, Ali Chouman, and Abdallah Shami. "An NWDAF Approach to 5G Core Network Signaling Traffic: Analysis and Characterization". In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 2022.

[116] Yachao Yuan et al. "Insight of Anomaly Detection with NWDAF in 5G". In: *2022 International Conference on Computer, Information and Telecommunication Systems (CITS)*. 2022.

[117] Ali Chouman, Dimitrios Michael Manias, and Abdallah Shami. "Towards supporting intelligence in 5G/6G core networks: NWDAF implementation and initial analysis". In: *2022 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE. 2022.

[118] 3GPP. *3GPP 5G Open API, Release 17*. URL: `https://forge.3gpp.org/rep/all/5G_APIs/-/tree/REL-17`.

[119] Open RAN alliance. *O-RAN specifications*. 2023. URL: `https://www.o-ran.org/specifications`.

[120] Robert Schmidt, Mikel Irazabal, and Navid Nikaein. "FlexRIC: An SDK for next-Generation SD-RANs". In: CoNEXT '21. 2021.

[121] Sepp Hochreiter. "The vanishing gradient problem during learning recurrent neural nets and problem solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[122] Redouane Niboucha et al. "Zero-touch security management for mMTC network slices: DDoS attack detection and mitigation". In: *IEEE Internet of Things Journal* (2022).

[123] Karim Boutiba, Miloud Bagaa, and Adlen Ksentini. "Radio Link Failure Prediction in 5G Networks". In: *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2021, pp. 1–6.

[124] Anon. *Zero-touch network and service management (ZSM); Requirements based on documented scenarios*. Accessed 29 March 2021. 2021. URL: `https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/001/01.01.01_60/gs_ZSM001v010101p.pdf`.

[125] Song Wang et al. "Machine learning in network anomaly detection: A survey". In: *IEEE Access* 9 (2021), pp. 152379–152396.

[126] Dan Niu et al. "A Network Traffic anomaly Detection method based on CNN and XGBoost". In: *2020 Chinese Automation Congress (CAC)*. IEEE. 2020, pp. 5453–5457.

[127] Xuansheng Wu et al. "Usable XAI: 10 Strategies Towards Exploiting Explainability in the LLM Era". In: *arXiv preprint arXiv:2403.08946* (2024).

[128] Jing Su et al. “Large Language Models for Forecasting and Anomaly Detection: A Systematic Literature Review”. In: *arXiv preprint arXiv:2402.10350* (2024).

[129] Mohamed Mekki, Nassima Toumi, and Adlen Ksentini. “Microservices configurations and the impact on the performance in cloud native environments”. In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE. 2022, pp. 239–244.

[130] A. Gokaslan and V. Cohen. *OpenWebText Corpus*. Skylion007.github.io. 2019. URL: http://Skylion007.github.io/OpenWebTextCorpus.

[131] Christian Buck, Kenneth Heafield, and Bas Van Ooyen. “N-gram Counts and Language Models from the Common Crawl.” In: *LREC*. Vol. 2. 2014, p. 4.

[132] Luca Soldaini et al. “Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research”. In: *arXiv preprint arXiv:2402.00159* (2024).

[133] Erik Cambria and Bebo White. “Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]”. In: *Computational Intelligence Magazine, IEEE* 9 (May 2014), pp. 48–57. DOI: 10.1109/MCI.2014.2307227.

[134] Hyung Won Chung et al. *Scaling Instruction-Finetuned Language Models*. 2022. DOI: 10.48550/ARXIV.2210.11416. URL: https://arxiv.org/abs/2210.11416.

[135] Gemma Team et al. “Gemma: Open models based on gemini research and technology”. In: *arXiv preprint arXiv:2403.08295* (2024).

[136] Yiheng Liu et al. “Understanding llms: A comprehensive overview from training to inference”. In: *arXiv preprint arXiv:2401.02038* (2024).

[137] Hao Zhou et al. “Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities”. In: *arXiv preprint arXiv:2405.10825* (2024).

[138] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[139] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[140] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. “Data augmentation for low-resource neural machine translation”. In: *arXiv preprint arXiv:1705.00440* (2017).

[141] Heereen Shim et al. “Synthetic Data Generation and Multi-Task Learning for Extracting Temporal Information from Health-Related Narrative Text”. In: *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*. Ed. by Wei Xu et al. Online: Association for Computational Linguistics, Nov. 2021, pp. 260–273. DOI: 10.18653/v1/2021.wnut-1.29. URL: https://aclanthology.org/2021.wnut-1.29.

[142] Mitesh Mangaonkar and Venkata Karthik Penikalapati. “Enhancing Production Data Pipeline Monitoring and Reliability through Large Language Models (LLMs)”. In: *Eduzone International peer reviewed/refereed academic multidisciplinary journal* 13 (Jan. 2024), pp. 51–56.

[143] Ajay Patel, Colin Raffel, and Chris Callison-Burch. “DataDreamer: A Tool for Synthetic Data Generation and Reproducible LLM Workflows”. In: *arXiv preprint arXiv:2402.10379* (2024).

[144] Nihal V Nayak et al. “Learning to generate instruction tuning datasets for zero-shot task adaptation”. In: *arXiv preprint arXiv:2402.18334* (2024).

[145] Hang Zou et al. “TelecomGPT: A Framework to Build Telecom-Specfic Large Language Models”. In: *arXiv preprint arXiv:2407.09424* (2024).

[146] Yaowei Zheng et al. “LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models”. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Bangkok, Thailand: Association for Computational Linguistics, 2024. URL: http://arxiv.org/abs/2403.13372.

[147] Tianyi Zhang et al. "Bertscore: Evaluating text generation with bert". In: *arXiv preprint arXiv:1904.09675* (2019).

[148] Ansar Aynetdinov and Alan Akbik. "SemScore: Automated Evaluation of Instruction-Tuned LLMs based on Semantic Textual Similarity". In: *arXiv preprint arXiv:2401.17072* (2024).

[149] Ning Ding et al. "Parameter-efficient fine-tuning of large-scale pre-trained language models". In: *Nature Machine Intelligence* 5.3 (2023), pp. 220–235.

[150] Dan Hendrycks et al. "Measuring massive multitask language understanding". In: *arXiv preprint arXiv:2009.03300* (2020).

[151] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013.

[152] Kaitao Song et al. "Mpnet: Masked and permuted pre-training for language understanding". In: *Advances in neural information processing systems* 33 (2020), pp. 16857–16867.

[153] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Ed. by Pierre Isabelle, Eugene Charniak, and Dekang Lin. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040.

[154] Alon Lavie and Abhaya Agarwal. "Meteor: an automatic metric for MT evaluation with high levels of correlation with human judgments". In: *Proceedings of the Second Workshop on Statistical Machine Translation*. StatMT '07. Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 228–231.

[155] Fred Jelinek et al. "Perplexity—a measure of the difficulty of speech recognition tasks". In: *The Journal of the Acoustical Society of America* 62.S1 (1977), S63–S63.

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

[157] Zheng Zhang et al. "Balancing specialized and general skills in llms: The impact of modern tuning and data strategy". In: *arXiv preprint arXiv:2310.04945* (2023).

[158] Kilian Carolan, Laura Fennelly, and Alan F Smeaton. "A Review of Multi-Modal Large Language and Vision Models". In: *arXiv preprint arXiv:2404.01322* (2024).

[159] Azzedine Idir Ait Said et al. "5G INSTRUCT Forge: An Advanced Data Engineering Pipeline for Making LLMs Learn 5G". In: *IEEE Transactions on Cognitive Communications and Networking* (2024).

[160] Colin White et al. "LiveBench: A Challenging, Contamination-Free LLM Benchmark". In: *arXiv preprint arXiv:2406.19314* (2024).

[161] Yueyue Liu et al. "OptLLM: Optimal Assignment of Queries to Large Language Models". In: *arXiv preprint arXiv:2405.15130* (2024).

[162] Chengyi Nie, Rodrigo Fonseca, and Zhenhua Liu. "Aladdin: Joint Placement and Scaling for SLO-Aware LLM Serving". In: *arXiv preprint arXiv:2405.06856* (2024).

[163] Siddharth Jha et al. "Learned Best-Effort LLM Serving". In: *arXiv preprint arXiv:2401.07886* (2024).

[164] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[165] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[166] Jovan Stojkovic et al. "Towards Greener LLMs: Bringing Energy-Efficiency to the Forefront of LLM Inference". In: *arXiv preprint arXiv:2403.20306* (2024).

[167] Amirhossein Kazemnejad et al. “VinePPO: Refining Credit Assignment in RL Training of LLMs”. In: *arXiv preprint arXiv:2410.01679* (2024).

[168] PENG Bo. *BlinkDL/RWKV-LM: 0.01*. Version 0.01. Aug. 2021. DOI: 10.5281/zenodo.5196577. URL: https://doi.org/10.5281/zenodo.5196577.

[169] Albert Gu et al. “Combining recurrent, convolutional, and continuous-time models with linear state space layers”. In: *Advances in neural information processing systems* 34 (2021), pp. 572–585.