ETSI GR ENI 041 V4.1.1 (2025-12)

GROUP REPORT

Experiential Networked Intelligence (ENI); Knowledge-Enhanced Network LLMs

Disclaimer

The present document has been produced and approved by the Experiential Networked Intelligence (ENI) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGR/ENI-0041v411_NKMELMNOAM

Keywords
knowledge-enhanced, network large language model

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Experiential Networked Intelligence (ENI).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

In the field of communication networks, large language models face issues such as knowledge cutoff, high training costs, long development and training cycles, a tendency to hallucinate, difficulty in providing deep knowledge in specialized areas, and the complexity in ingesting and generating real-time network operational data. The present document describes the development of knowledge-enhanced network Large Language Models. The present document explains the motivation for developing a Knowledge-Enhanced Large Language Model, followed by recommendations for how a Large Language Model is trained and then used for network Operation, Administration, Maintenance, and Performance (OAMP) operations.

1 Scope

The present document describes the development of knowledge-enhanced network Large Language Models.

The present document will explain the motivation for developing a knowledge-enhanced network Large Language Model, followed by recommendations for how a network Large Language Model is trained and then used for network Operation, Administration, Maintenance, and Performance (OAMP) operations.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

[i.1] ETSI GR ENI 004 (V3.1.1): "Experiential Networked Intelligence (ENI); ENI terminology".

[i.2] ETSI GR ENI 016 (V2.1.1): "Experiential Networked Intelligence (ENI); Functional Concepts for Modular System Operation".

[i.3] ETSI GS ENI 005 (V3.1.1): "Experiential Networked Intelligence (ENI); ENI System Architecture".

[i.4] ETSI GS ENI 030 (V4.1.1): "Experiential Networked Intelligence (ENI); Transformer Architecture for Policy Translation; Knowledge-based Reasoning using the Functionalities of Transformers and Knowledge Graphs to Generate Policies".

[i.5] ETSI GR ENI 031 (V4.1.1): "Experiential Networked Intelligence (ENI); Construction and application of fault maintenance network knowledge graphs".

[i.6] Anwar, M. et al.: "Understanding Misunderstandings: Evaluating LLMs on Networking Questions". In Proceedings of the ACM SIGCOMM2024Conference.

[i.7] R. Wang et al.: "Role Prompting Guided Domain Adaptation with General Capability Preserve for Large Language Models". In Findings of the Association for Computational Linguistics: NAACL 2024.

[i.8] D. Edge et al.: "From Local to Global: A GraphRAG Approach to Query-Focused Summarization", April 2024 (latest version is February 2025).

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GR ENI 004 [i.1], ETSI GS ENI 005 [i.3], ETSI GS ENI 030 [i.4] and the following apply:

knowledge: analysis of data and information, resulting in an understanding of what the data and information mean

NOTE 1: Knowledge represents a set of patterns that are used to explain, as well as predict, what has happened, is happening, or is possible to happen in the future; it is based on acquisition of data, information, and skills through experience and education.

NOTE 2: While "analysis" is key, the process of gaining knowledge sometimes also involves synthesis, interpretation, and reflection.

1) **inferred knowledge:** knowledge created based on reasoning using evidence provided

2) **measured knowledge:** knowledge resulting from the analysis of data and information that was measured or reported

3) **propositional knowledge:** knowledge of a proposition, along with a set of facts that prove (or disprove) the proposition

   NOTE 1: This is available in ETSI GS ENI 005 [i.3].

   NOTE 2: The standard philosophical definition of propositional knowledge, often called the "Justified True Belief" (JTB) account, has three conditions:

      1) **Belief**: the proposition is believed;

      2) **Truth**: the proposition is true;

      3) **Justification**: there is a good reason or justification for believing it is true.

knowledge-enhanced LLM: LLM that is systematically augmented with structured external knowledge sources to improve factual accuracy, reasoning depth, and interpretability

pipeline: end-to-end construct that orchestrates a flow of events and data in response to a trigger

NOTE: This is available in ETSI GS ENI 030 [i.4].

Retrieval-Augmented Generation (RAG) pipeline: end-to-end construct comprising retrieval and generation modules that collaboratively and dynamically enhance language model outputs by leveraging external knowledge bases

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GR ENI 004 [i.1], ETSI GS ENI 005 [i.3], ETSI GR ENI 016 [i.2] and the following apply:

ChatGPT	Chat Generative Pre-trained Transformer
COT	Chain-Of-Thought
CPT	Continual Pre-Training
DAPT	Domain Adaptive Pre-Training
GPT	Generative Pre-Trained transformer
ICL	In-Context Learning
IRCOT	Interleaved Retrieval with Chain-Of-Thought

LLM	Large Language Model
KELLM	Knowledge-Enhanced Large Language Model
NER	Named Entity Recognition
NLP	Natural Language Processing
NKELLM	Network Knowledge-Enhanced Large Language Model
RAFT	Retrieval-Augmented Fine-Tuning
RAG	Retrieval-Augmented Generation
SFT	Supervised Fine-tuning
TSFT	Task-Specific Fine-Tuning

4 Background and Motivation

4.1 Background

In the field of communication networks, Large Language Models (LLMs) typically have a "knowledge cutoff," which occurs as a consequence of static training datasets and the prohibitive cost of continuous retraining. This is the point in time up to which the model has been trained with data. Any information or events that happened after the cutoff date are unknown to the model. The knowledge cutoff exists due to high training costs and long development and training cycles. In addition, additional work needs to be done to mitigate hallucinations, provide deep knowledge in specialized areas, and ingest and generate real-time network operational data.

There are a number of approaches to addressing the knowledge cutoff problem. One approach is to use a hybrid fine-tuning method such as Domain Adaptive Pre-Training (DAPT) with task-specific tuning using curated datasets to preserve generalizability. However, DAPT is typically cost-prohibitive.

Another approach is to use In-Context Learning (ICL) with Retrieval-Augmented Generation (RAG). During inference, a combination of ICL and an advanced RAG solution will provide appropriate up-to-date domain-specific knowledge. This combination directly addresses the need for "up-to-date domain-specific knowledge" without the high cost of retraining the entire model.

Typical challenges include:

1) training data pipelines often include outdated documents, causing discrepancies between reported and effective cutoffs; and

2) semantic duplicates during pretraining can skew knowledge recency.

4.2 Motivation

The rapid growth of LLMs has brought significant advancements in natural language understanding, generation, and decision-making. Networking environments, characterized by rapid changes, diverse data sources, and complex topologies, require models that can effectively manage and utilize knowledge relevant to these systems. However, deploying LLMs in specialized fields like networking often highlights the limitations of these models in leveraging domain-specific knowledge. For example, while domain adaptation mechanisms such as Continual Pre-Training (CPT) and Retrieval-Augmented Fine-Tuning (RAFT) work well on most domains, studies show the performance of LLMs in the networking domain is often inconsistent and unreliable for professional use without enhancement [i.6]. This stems from their tendency to prioritize statistical patterns over causal reasoning - a critical requirement in root-cause analysis.

A fundamental limitation arises during domain adaptation. When a general-purpose LLM is fine-tuned on a narrow domain like networking, it often loses its broad linguistic capabilities. This is a fundamental challenge because the process of improving specialized performance can degrade general performance, challenging the viability of a single model for hybrid tasks ([i.7]). This is called "catastrophic forgetting." More specifically, this is caused by the model's parameters being updated to get better at that specific domain. This in turn causes the model to overwrite (or "forget") the information it previously learned, leading to a degradation in its general capabilities (like summarization, general conversation, etc.). This trade-off challenges the viability of single-model solutions for hybrid tasks requiring both technical and linguistic proficiency [i.7].

The motivation for developing a "Knowledge-Enhanced LLM" (KELLM) stems from the unique challenges posed by modern networking environments and the limitations of traditional LLMs in handling domain-specific and complex scenarios. Key motivating factors include:

1) Temporal Dynamics

Networking environments evolve at millisecond timescales, with traffic patterns, device states, and security threats changing dynamically. Current LLM architectures, which rely on static pre-training corpora and batch-oriented updates, struggle to maintain situational awareness. For example:

a) Latency Sensitivity: Edge-deployed LLMs are required to process network telemetry within 50 ms to 100 ms to support real-time decisions like traffic rerouting. However, even optimized models like GPT-3.5-turbo exhibit median inference latencies of 320 ms on standard GPU hardware, exceeding acceptable thresholds for time-sensitive operations.

b) Concept Drift: Network configurations and threat landscapes change unpredictably. Without continuous online learning, LLM accuracy degrades by 2 % to 4 % weekly in production environments.

2) Security Concerns

Deploying LLMs in critical network infrastructure introduces novel attack vectors:

a) Adversarial Prompting: Malicious actors can exploit LLM vulnerabilities to generate harmful configurations. In controlled tests, researchers induced BGP route leaks 37 % of the time by crafting subtle prompt variations.

b) Data Leakage Risks: LLMs processing network logs sometimes inadvertently memorize and expose sensitive information like IP addresses or authentication tokens. Differential privacy methods reduce this risk but increase model perplexity by 30 % to 40 %, harming task performance.

3) Multi-source knowledge integration

Networking involves specialized protocols, configurations, and problem-solving methods. Typical training recipes for existing LLMs do not include network telemetry, logs, and other OAMP examples. Since existing LLMs have not seen enough examples of these types of data, the LLM has not learned the statistical patterns, relationships, and vocabulary of the networking domain. Therefore, it cannot generate accurate or reliable responses for tasks like fault detection, routing optimization, or security analysis. Enhancing an LLM with networking domain-specific knowledge bridges this gap, enabling it to address complex technical challenges effectively. However, this can be enhanced using multi-agent frameworks like Microsoft's AutoGen and LangChain, which enable collaborative problem-solving by delegating subtasks to specialized agents. For example, one agent might parse network logs, another analyse traffic anomalies, and a third generate mitigation strategies. This also enables data fusion to holistically combine the results of these tasks to look for related knowledge. These systems often leverage RAG to ground responses in real-time data from vector databases.

4) Reduce the hallucination of LLMs

Hallucinations are incorrect or fabricated outputs. They pose significant risks in networking, where an erroneous configuration suggestion or a misdiagnosed fault can disrupt operations. Generic LLMs, often trained on data that lacks specialized context for networking operations, can generate ungrounded (i.e. not based on facts) responses. To mitigate this, RAG pipelines can be used to ground LLM responses to a set of comprehensive knowledge sources. Each knowledge source contains essential static information such as device configurations, network topology documents, and architectural standards as well as dynamic data that need to be continuously updated with the outputs from network monitoring tools. By retrieving relevant, up-to-date information before generating a response, the LLM's outputs become anchored in fact, demonstrably reducing hallucinations. Another complementary technique is Chain-of-Thought (CoT) prompting, see ETSI GS ENI 030 [i.4] or similar mechanisms, which guide the model to break down a problem and generate a sequence of reasoning steps. While these steps are not typically verified by external tools in real-time, they make the model's logical pathway transparent and can improve the quality of the final output. For true interactive verification, AI Agent-based systems can be used, where guardrails cross-reference proposed actions (e.g. generated CLI commands) against device APIs or operational policies before execution.

5) Improve the accuracy rate of LLMs

Networking problems often require solutions tailored to specific conditions, such as topology, traffic patterns, and historical data. An LLM enhanced with knowledge can provide more context-aware, precise, and actionable insights, improving operational outcomes.

To improve learning effectiveness, a structured training approach, sometimes called curriculum learning, can be beneficial. This involves first exposing the model to foundational concepts, such as the principles of individual network protocols or the functions of basic device types. However, a simplistic hierarchical strategy is insufficient on its own. True network intelligence requires understanding the complex, non-hierarchical interdependencies between different domains. For example, how a security policy can influence routing decisions, or how application performance needs dictate quality of service configurations. Therefore, the training process needs to progress from foundational knowledge to modelling these intricate, cross-layer and cross-technology relationships, which are often better represented as a dense knowledge graph rather than a simple hierarchy.

6) Enhancing human-computer interaction

Networking teams face challenges in interpreting and acting upon dense technical data. An LLM can serve as an intermediary, simplifying complex information, enabling better decision-making, and improving collaboration between human operators and automated systems. By analysing the intent of user input in depth, it is possible to understand exactly what the requirements are. Whether it is a simple query or a complex task description, the model captures key information better and provides a more tailored response to the user's needs. For example, in the intelligent customer service scenario, can quickly understand the user enquiry product problems, and give targeted answers.

4.3 Enhancing LLMs with External Knowledge

4.3.1 Characteristics

A Network Knowledge-Enhanced Large Language Model (NKELLM) is characterized by its ability to:

1) access and reason over external knowledge sources beyond its parametric memory;
2) maintain up-to-date information without retraining; and
3) perform domain-specific tasks with contextual accuracy.

Certain types of NKELLMs provide the ability to mitigate hallucinations through evidence-based generation and/or types of logical reasoning, see ETSI GS ENI 030 [i.4]. The most effective systems employ hybrid retrieval from multiple validated sources while implementing rigorous consistency checks.

4.3.2 Types of External Knowledge Sources

The most common external source is a Knowledge Graph, for the descriptions of Knowledge Graph can see ETSI GS ENI 005 [i.3] and ETSI GS ENI 030 [i.4], since this provides a structured and efficient way to represent and utilize knowledge. Other forms of external knowledge sources include structured knowledge bases (e.g. Wikidata and DBPedia), unstructured or semi-structured corpora (e.g. academic paper repository or technical documentation), multimedia archives (e.g. video transcripts or image-text pairs, such as LAION-5B), and dynamic knowledge sources (e.g. network telemetry and trouble tickets). However, the dynamic knowledge sources listed cannot be directly ingested by any type of LLM. These types of sources, though very valuable, require specialized preprocessing and infrastructure, including protocol-specific decoders for raw data normalization, stream processing engines for temporal alignment/aggregation, and domain-optimized Natural Language Processing (NLP) pipelines for ticket analysis.

The present document focusses on different types of knowledge, including knowledge base and knowledge graph to serve as the external knowledge source. The present document will also examine mechanisms to increase knowledge enhancement when using RAG, such as in-context learning. Examples of using a Knowledge Graph to create an NKELLM are defined in ETSI GS ENI 030 [i.4] and are not covered in the present document.

4.3.3 RAG Systems

There are a large number of RAG systems, each with different architectures optimized for different problems:

1) Auto-RAG use LLMs to autonomously decide when/what to retrieve using reinforcement learning.
2) Modular RAG decouples retrieval, reranking, and generation into interchangeable components, enabling customized pipelines for domain-specific needs.

3) Retrieval-Augmented Fine-Tuning (RAFT) fine-tunes LLMs on datasets where answers depend on retrieved documents, teaching models to identify irrelevant contexts and synthesize multi-document evidence.

4) Recursive RAG retrieves hierarchical document summaries before drilling into granular chunks, ensuring comprehensive context capture.

5) Iterative RAG systems improve upon standard RAG by performing multiple cycles of refinement. It iteratively improves the quality of output by repeating the retrieval process in a loop, refining the search criteria each time based on the previous results until the desired outcome is achieved.

6) Interleaved Retrieval with Chain-of-Thought (IRCOT) dynamically integrates external information retrieval with structured reasoning. IRCOT transforms RAG from a static 'retrieve-then-answer' process into a dynamic, multi-step reasoning loop. Instead of retrieving all potentially relevant information at the start, the model follows a 'think-search-assimilate' cycle. It begins reasoning (the 'Chain-of-Thought'), identifies a specific unknown, triggers a targeted retrieval for just that piece of information, and then integrates the new fact to continue its reasoning process. This iterative approach mimics how a human expert solves a problem, leading to more focused and efficient information gathering.

For RAG systems enhanced with networking knowledge, it is recommended to **start** with a Modular RAG architecture due to its flexibility. For specific, high-value application scenarios like automated fault diagnosis or advanced network optimization, it is recommended implementing IRCOT-style reasoning loops **within** that modular framework. These systems can effectively handle advanced strategy design, network architecture consulting, and similar tasks.

Every RAG system needs to have the following functions:

1) Query parsing: Tokenization, Named Entity Recognition (NER), keyword extraction, and multi-turn context tracking.

2) Retrieval: Variations include sparse retrieval, dense vector retrieval, and optionally hybrid retrieval.

3) Result Processing: Reranking of retrieved documents, context assembly with prompt templating.

4) Generation: Response generation via LLMs (e.g. ChatGPT, LLaMA, Gemini), source attribution, and answer formatting.

5) Data pipeline: Document upload, chunking, cleaning, embedding generation, and managing vector/index databases.

5 Creating Network Knowledges to Enhance LLMs

5.1 Overview

Creating a KELLM requires external knowledge, which often comes from operation manuals, regulations, equipment manuals, papers, third-party databases, etc. One way to do this is to develop a knowledge graph-enhanced LLM (or transformer), which incorporates the knowledge graph during the pre-training and inference phases of the LLM (or transformer). For this to work, knowledge needs to be stored in a format that can be understood by the LLM (or transformer). Therefore, extracting knowledge from an LLM (or transformer) to construct a knowledge graph or database has become a common practice.

The LLM can be pre-trained by using the entity and relationship information in the knowledge graph ETSI GS ENI 030 [i.4]. This is called the "knowledge injection" approach, which puts semantic information into the model's architecture or input. While the knowledge becomes part of the model's core reasoning, this is not an optimal approach. This is because it is extremely expensive, inflexible (if the knowledge graph changes, the LLM needs to be retrained), and suffers from catastrophic forgetting.

A better approach is to use a modified RAG system that is *integrated* with a knowledge graph. In this approach, the LLM first understands the user query and identifies key entities and relationships. Then, instead of a simple vector search, it does a *graph-based retrieval*, that returns a **subgraph**. The subgraph is then passed to the LLM as context. This approach is known as GraphRAG [i.8].

The knowledge in the knowledge graph (e.g. nodes, edges, properties of both, and constraints), when grounded by a set of ontologies see ETSI GS ENI 030 [i.4], can be used as the basis and reference for large language model reasoning. An ontology provides:

1) The schema and vocabulary for the knowledge graph to use. For example, it defines what types of entities can exist (e.g. Router, Firewall, VLAN), what properties they can have (ipAddress, firmwareVersion), and what types of relationships can link them (connectsTo, isMonitoredBy). Without this, the graph does not have consistent semantics.

2) It enables reliable querying: For an LLM to "call on the relevant knowledge," it needs to know how to ask for it. The ontology provides the consistent vocabulary for querying. Without it, the LLM would have to guess whether to ask for a firewall, a security_appliance, or a packet_filter. An ontology ensures there is one, unambiguous term.

3) It enables logical inference: This is the most critical point. A good ontology includes axioms and constraints (e.g. "a Router is a subclass of NetworkDevice"; "every Interface needs to be connectedTo something"). This enables the system to infer new facts that are not explicitly stated in the graph, which is the foundation of true reasoning.

The LLM can call on the relevant knowledge in the knowledge base for logical analysis and deduction, so as to get a more reasonable answer. This is typically done using either RAG, GraphRAG, and/or prompt augmentation, where relevant facts from the knowledge base are retrieved and supplied as context for the LLM's response generation. According to different application scenarios and user needs, a specific knowledge base can be constructed, so that the LLM can be customized in specific areas. For example, in an intelligent customer service scenario, a knowledge base is built for product knowledge and common problems of users, so that the model can better provide accurate help for users; in the field of education, a subject knowledge base is built that supports models for teaching and answering questions.

NOTE: An LLM is still a statistical model, even when using a knowledge graph as described above. For example, it does not perform causal reasoning. However, with targeted training on causal rules or with new causal-aware architectures, an LLM learns to approximate some aspects of causal reasoning and apply causal principles in limited contexts.

5.2 Creating a Knowledge Graph

The construction process of knowledge graph refers to ETSI GR ENI 031 [i.5] on network fault maintenance knowledge graph construction. For content related to knowledge graph function management and capability orchestration, refer to ETSI GS ENI 030 [i.4].The construction process is as follows:

1) Data acquisition and processing

Data sources include two categories: knowledge data and network operation data.

The scope of knowledge data includes but is not limited to:

a) World knowledge, such as general web pages, books, encyclopaedias, code and other data.

b) Industry data related to network, such as network-related standards and specifications, papers, patents, books, etc.

c) Work order data for networks, such as customer complaint data, fault work orders, activation work orders, service incidents, etc.

Network operation data includes but is not limited to:

a) Network operation status data, such as configuration data, alarm data, fault data, performance data, log data, DPI data, network topology data, command data; etc.

b) Inspection and testing-type data, such as operation data, drive test data, etc.

The data above needs to be regularly collected, processed (e.g. perform tasks such as data filtering, correlation, cleansing, and deduplication) and normalized. After processing, the data is stored for further processing.

2) Knowledge Extraction

Knowledge extraction transforms unstructured or semi-structured network data into structured knowledge suitable for inclusion in a knowledge graph. This process requires sophisticated techniques to handle the domain's inherent complexity and consists of three main subtasks: entity extraction, relationship extraction, and attribute extraction.

Entity Extraction involves identifying and categorizing key entities from diverse documents and data sources. The primary goal is to recognize concepts critical to network operations. Exemplary entities include network devices (e.g. routers, switches, firewalls), logical constructs (e.g. VLANs, VRFs), protocols (e.g. OSPF, BGP, SNMP), technical concepts (e.g. Quality of Service, MPLS), and operational artifacts (e.g. trouble tickets, configuration files). A significant challenge in the networking domain is ambiguity that requires deep contextual understanding. For example, the term "port 22" needs to be disambiguated to determine if it refers to TCP/22 (SSH), UDP/22, or a physical interface labelled '22' on a device. Similarly, an alert like "link down" requires disambiguation to distinguish between an administrative shutdown (intentional) and a physical layer failure (unintentional). Standard NER models need to be fine-tuned with network-specific data to resolve these cases correctly. Once extracted and disambiguated, entities need be linked to a canonical entry in the knowledge graph to ensure consistency and prevent data duplication.

Relationship Extraction identifies and classifies the semantic relationships between the extracted entities. There are two types of relationships: explicit and implicit. Explicit relationships are stated clearly in text, such as in design documents ("Router-A is connected to Switch-B") or technical specifications ("BGP is a type of exterior gateway protocol"). Transformer-based workflows are effective at identifying these explicit links. Implicit relationships are not directly stated but are *implied by the data*. For instance, a specific CLI command in a configuration file (interface GigabitEthernet0/1; ip address 10.1.1.1 255.255.255.0) implicitly creates a hasIPAddress relationship between the interface and the IP address. Likewise, a sequence of log messages can imply a causal relationship between a power spike and a subsequent device reboot. Extracting these implicit links requires advanced natural language processing (NLP) models capable of understanding command syntax, log semantics, and operational cause-and-effect, moving beyond simple text analysis.

Attribute Extraction retrieves descriptive properties or characteristics of entities, populating the knowledge graph with detailed, queryable data. It starts with finding specific attribute-value pairs associated with an entity. For a network device, this can include its model number, serial number, firmware version, or number of ports. For a protocol, it can be its administrative distance or timer values. Then, a combination of pattern matching (using regular expressions for structured data like IP addresses or MAC addresses) and NLP-based slot filling is typically used to extract attributes from both structured configuration files and unstructured text. Then, extracted attributes are normalized to a consistent format (e.g. standardizing units, date formats) and validated against schema constraints or authoritative sources to maintain data quality within the knowledge graph.

3) Knowledge Fusion

Knowledge fusion is a crucial step in constructing a knowledge graph related to network knowledge. It integrates information and knowledge from multiple, often heterogeneous, sources into a single, unified and consistent representation, such as a knowledge graph. It involves resolving conflicts and redundancies between sources to create a coherent dataset, which can then be used to infer new knowledge and generate a more holistic understanding of a domain. The main processes are as follows:

1. **Schema Integration**

   Before merging data, it is necessary to align the schemas of the different sources. This involves mapping equivalent classes, properties, and constraints to ensure that entities and relationships from different sources are interpreted consistently within the unified knowledge graph. The use of a formal ontology is critical here, as it provides a canonical schema and a consensual vocabulary that serves as the "ground truth" for integration, preventing ambiguity in the unified knowledge graph.

2. **Entity Alignment**

   Entity alignment (also known as entity resolution) identifies and merges records from different sources that refer to the same real-world entity across sources. Methods include:

   a) Name-based Similarity: The challenge with naive methods like Name-based Similarity is that they are notoriously unreliable in the networking domain. Device hostnames (e.g. lon-core-rtr-01, london.router.core.01) can vary significantly while referring to the same device.

b) A Robust Approach: A robust alignment strategy needs to prioritize unique, stable identifiers as primary keys for fusion. These include hardware MAC addresses, device serial numbers, or unique management IP addresses.

c) Using Secondary Signals: Softer signals, such as attribute-based similarity (e.g. matching model number and OS version) or context-based similarity (e.g. identifying two routers connected to the same switch), can then be used as secondary, lower-confidence evidence to resolve ambiguities or link entities that lack a primary key.

Once similarity is scored using this weighted approach, a threshold can be applied, with manual verification reserved for ambiguous or critical conflicts.

3. **Instance Matching:** This consists of detecting duplicates or near-duplicates and consolidating them into a single entity representation.

4. **Relationship Alignment and Integration**

Relationship alignment maps and merges equivalent or related relationships from different sources. This includes:

a) Relationship Type Mapping: Standardizing relationship types that are named or represented differently across sources. For example, one source uses the term "connected to" to describe the relationship between a network device and a network, while another source uses "linked with". Mapping these different relationship type expressions to a unified set of relationship types in the knowledge graph is important. An ontology is the preferred method for standardizing relationship types (e.g. ensuring connected to and linked with are both mapped to a single, canonical connectsTo relationship).

b) Relationship Merging: All valid relationships for aligned entities are integrated, ensuring the graph is semantically consistent and complete as possible.

c) Relationship Conflict Detection and Resolution: Identifying and resolving contradictory relationships by evaluating source reliability (e.g. data from the device's running configuration is more authoritative than a stale inventory spreadsheet), data recency, or by flagging the conflict for expert review.

5. **Attribute Aggregation and Conflict Resolution**

For each unified entity, all relevant attributes from different sources are collected and merged. When attribute values conflict, resolution strategies include:

a) Source Confidence: Preferring data from more reliable or recent sources.

b) Voting or Consensus: Using majority or weighted voting among sources.

c) Manual Verification: Resorting to expert review for ambiguous or critical conflicts.

6. **Knowledge Enrichment and Reasoning**

A key goal of knowledge fusion is to enable the creation of new knowledge that does not explicitly exist in any single source by combining information from multiple sources, such as discovering new relationships or deducing higher-level concepts. This includes:

a) Logical Inference: By leveraging the axioms and rules defined in the ontology, the system can infer new facts. For example, if the graph knows Router-A is connected to Switch-B, and Switch-B is in the London_Data_Centre, it can infer that Router-A is located in the London_Data_Centre.

b) Completing Missing Data: Gaps in one data source can be filled by leveraging overlapping information from another.

7. **Provenance Tracking and Metadata Management**

To ensure trust and maintainability, it is crucial to maintain metadata about the origin, timestamp, and confidence level of every piece of knowledge in the graph. This enables traceability, auditability, and future updates to the knowledge graph.

4) Knowledge Graph Storage

A knowledge graph is a type of knowledge base. The more meaningful distinction is between structured knowledge (graphs, relational databases) and unstructured knowledge (document stores indexed for vector search). A comprehensive NKELLM will likely use a hybrid approach: a knowledge graph for structured facts and reasoning, and a vector database for semantic search over raw documents (manuals, RFCs, tickets). There are two ways to store the knowledge graph:

a) Databases: If the information needs to be queried, analysed, and other operations in the future, it can be structured and stored in a database. For example, use a relational database (such as MySQL, SQLite, etc.) to design an appropriate data table structure (such as entity tables, relationship tables, attribute tables, etc.) and store entity, relationship, attribute, and other information in the corresponding data tables. For unstructured text information, text fields can be used for storage.

b) Knowledge Graph Databases: A knowledge graph can also be stored using a graph database. This allows for easy querying and reasoning operations on the knowledge graph.

5) Creating a Solution Based on GraphRAG

GraphRAG (Graph Retrieval-Augmented Generation) is a knowledge-enhanced generation approach that combines graph structures with language models. It builds a knowledge graph by organizing entities and their relationships, enabling more structured and efficient retrieval of relevant information. During the retrieval phase, GraphRAG leverages the graph to identify relevant subgraphs related to a query. In the generation phase, the language model uses the retrieved graph-based context to produce more accurate and coherent responses. This method improves the model's ability to understand and reason over complex knowledge, making it well-suited for tasks like question answering and dialogue systems.

After completing the construction of the knowledge graph, the application or system orchestrates the retrieval of needed knowledge by interpreting the user's natural language query, translating it (often with the help of an LLM) into a graph query (e.g. using Cypher or SPARQL). It then retrieves the relevant subgraph and then provides this context to the LLM for answer generation. This solution can iteratively query for supplementary information to optimize results when needed.

5.3 Creating a Knowledge Base

A knowledge base is an information system used to store, manage, and apply knowledge. Its core purpose is to enable computers to "understand" and "use" knowledge to assist humans in decision-making, question answering, or reasoning. In the context of large language models, it serves to support the model in making decisions, answering questions, or performing reasoning tasks.

A knowledge graph is one type of knowledge base. A state-of-the-art knowledge base for a complex domain like networking is not a single entity but a hybrid system that combines structured and unstructured knowledge stores to provide a complete informational picture for the LLM.

This hybrid approach acknowledges that some network knowledge consists of discrete facts and relationships (ideal for a graph), while other knowledge is contained within vast amounts of text (ideal for semantic search).

As detailed in clause 5.2, the knowledge graph serves as the structured backbone of the knowledge base. It stores entities (devices, protocols, VLANs), their attributes (IP addresses, firmware versions), and the explicit relationships between them (connectsTo, isMemberOf). It excels at answering factual questions, performing logical inference, and understanding the relational topology of the network. It is the source for "what is connected to what".

The second critical component is a database designed for unstructured data, which handles the vast corpus of text-based network knowledge. This is typically a vector store. It stores and indexes large volumes of documents, such as vendor manuals, RFCs, best-practice guides, standard operating procedures, and historical incident tickets. To do this, it uses two different mechanisms:

1) Chunking & Embedding: Documents are broken down into manageable text chunks. Each chunk is then processed by an embedding model, which converts the semantic meaning of the text into a numerical vector.

2) Vector Storage: These vectors are stored in a specialized vector database, indexed for efficient similarity searching.

NOTE 1: Most vector stores use cosine similarity or other similar geometric measures. For example, cosine similarity calculates the cosine of the angle between two vectors. It has no inherent understanding of language or meaning. The "semantic" part of "semantic search" comes entirely from the embedding model. The job of the embedding model is to convert text into vectors in such a way that semantically similar pieces of text have vectors that point in a similar direction. Hence, the quality of the search result depends almost entirely on how well the embedding model did its job of creating the vector space in the first place.

NOTE 2: The vector store is essentially a very fast geometric calculator, not a semantic reasoner. It is not possible to swap out the geometric similarity calculation. Instead, a two-phased approach is recommended:

a) Stage 1: Candidate Retrieval: Use the vector store's fast, built-in similarity search to get a large set of potentially relevant candidates (e.g. the top 100 results). This is a broad, "good enough" first pass.

b) Stage 2: Re-ranking: This is where the "real" semantic logic is applied. The top candidates are passed to a more sophisticated and computationally expensive model, often called a cross-encoder or a re-ranker. This model does not just compare two vectors; it looks at the raw text of the user's query and the raw text of each candidate document together to produce a much more accurate relevance score.

However, this is beyond the scope of the present document.

Primary Use Case: When a user asks a conceptual question or a query that requires deep contextual knowledge (e.g. "What is the recommended procedure for upgrading the OS on a vendor-specific network device?"), the system converts the query into a vector and retrieves the most semantically similar text chunks from the document store.

The true power of this hybrid approach is realized when both stores are used together within a single RAG workflow. This allows the system to move from simple fact retrieval to complex problem-solving.

Consider the query: "Why is latency high on the primary link to our London office?":

1) Step 1 (Graph Query): The system first queries the knowledge graph to resolve the ambiguous terms. It identifies the specific devices, interfaces, and circuits that constitute the "primary link to the London office".

2) Step 2 (Targeted Vector Search): Using the precise device models and interface types retrieved from the graph (e.g. "INET-_GTW_01," "interface Te0/0/0"), the system performs a highly targeted semantic search in the vector database for relevant documents, such as known bug reports, performance tuning guides, or past incident tickets related to those specific components.

3) Step 3 (Context Synthesis): The LLM is provided with a rich, multi-faceted context: the structured entity and relationship data from the graph and the relevant procedural or descriptive text from the vector database.

By combining these approaches, the LLM is grounded in both factual, relational truth and deep, contextual expertise, enabling it to generate responses that are far more accurate, comprehensive and actionable.

6 Training of Knowledge-Enhanced LLMs

6.1 Introduction

The training phase for LLMs typically involves two key stages: pre-training followed by fine-tuning. This approach leverages self-supervised learning for foundational knowledge acquisition and supervised learning for task-specific adaptation.

Pre-training utilizes self-supervised learning on vast, unlabelled datasets (e.g. web corpora) to build general linguistic and factual knowledge. This enables it to produce a base model capable of broad language understanding suitable for generic tasks but not specialized for specific tasks.

The fine-tuning phase adapts the pre-trained model using smaller, labelled datasets tailored to downstream applications (e.g. network management). Common strategies include:

1) Incremental Pre-training: Further pre-training on domain-specific data (e.g. telecom RFCs).

2) Supervised Fine-Tuning (SFT): Task-specific training (e.g. log analysis).

3) Instruction Tuning: Aligning model outputs with structured prompts.

4) Parameter-Efficient Fine-Tuning (or one of its variants): training time adaptation using minimal data.

5) Human Alignment: Techniques like Reinforcement Learning from Human Feedback (RLHF).

6) AI Alignment: Automated optimization via methods like Proximal Policy Optimization (PPO).

The key difference between the two approaches is that fine-tuning datasets are orders of magnitude smaller than pre-training corpora while achieving task specialization.

The following three deployment strategies are used by providers of network management LLM inference services based on resources and needs:

1) Full Pipeline Development: Pre-train a base model from scratch, then perform Task-Specific Fine-Tuning (TSFT).

2) Internal Model Customization: Select a pre-existing base model from an internal repository and fine-tune it.

3) Third-Party Model Adaptation: License an external base model (e.g. GPT-4, LLaMA-3) and utilize cloud platforms/services for fine-tuning.

6.2 Training Objectives

The primary objective is to develop a telecom-specialized NKELLM that is capable of automated OAMP tasks. Specific goals include:

1) Real-time telemetry: enable sub-10 ms ingestion and contextualization of streaming data. Examples include:

   a) SNMP/YANG traps (e.g. linkDown, bgpBackwardTransition).

   b) NetFlow/IPFIX records for traffic anomaly detection.

   c) PM counters.

2) Support question-answer and other applications for help desk and customer care solutions.

6.3 Data Engineering

6.3.1 Data Sources

The scope of network knowledge data includes but is not limited to:

a) General world network knowledge, such as general web pages, books, encyclopaedias, codes and other network knowledge data, etc.

b) Network operation management related industry data, such as network operation related standards and specifications, papers and patents, books, etc.

c) Network operation management work order data, such as customer complaint data, Breakdown Work Order, opening work order, service events, etc.

The following are some of the data engineering tasks for Telecom Knowledge-Enhanced LLMs:

1) Definition of model training and inference tools:

   a) Define the tooling to ingest structured, semi-structured, and unstructured data for training. Examples of these data include network configuration files (e.g. YANG, CLI), and 3GPP TSs), 3GP and IETF specifications in PDF, Word, XML, and JSON), and customer service call transcripts and network device manuals, respectively.

   b) Define the different tooling of inference requires, e.g. streaming data engineering pipelines.

2) Telecom-Specific Preprocessing:

   a) Use a semantic tokenizer (e.g. ModernBERT or SentencePiece) to tokenize telecom vocabulary).

   b) Convert heterogeneous timestamps (device logs, SNMP traps) with NTP alignment

   c) Perform entity recognition.

3) Curate Datasets:

   a) This consists of different types of datasets for wide task coverage (e.g. QA pairs, code snippets, and annotated troubleshooting logs).

   b) It is recommended that the above is augmented with synthetic data generation.

6.3.2 Data Engineering stage

The data engineering phase, which plays a key role in the training of large models for network operations management, mainly consists of the following data processing steps to prepare a dataset suitable for model training and evaluation:

1) Basic data processing steps:

   a) Data cleaning: Remove noise, errors or incomplete data, eliminate duplicate data records, improve data quality.

   b) Data selection: According to the specific conditions to screen out the required data to ensure the relevance and validity of the data.

   c) Data privacy desensitization: Processing data involving personal privacy or sensitive information to protect user privacy.

   d) Data modal transformation: The uniform transformation of different modal data into a format suitable for model processing.

2) Processing of modal-specific data:

   a) Text data annotation: Adding labels to the text data so that the model better understands and learns the meaning of the text.

   b) Audio data resampling: Resampling the audio data, adjusting the sampling rate and other parameters to meet the requirements of model training.

   c) Image data pre-processing: Various preprocessing operations such as resizing, rotating, cropping, and normalizing the image data to match the input requirements of the model.

3) Segmentation of a data set:

   a) The processed data can form a training data set. At the same time, the data set is divided according to the design requirements or preset strategy to form the training set and verification set for pre-training or parameter tuning of the large model These data sets can also be used as test sets for model evaluation capabilities.

6.4 Training of LLM

6.4.1 Model Selection

The model selection stage is driven by specific operational requirements. For a specialized application like network management, the typical approach is not to train a foundation model from scratch. Rather, an appropriate pre-trained base model is selected, either from an internal model repository, an open-source hub (like Hugging Face), or a commercial API provider, which then serves as the foundation for further fine-tuning and adaptation.

6.4.2 Model Training

LLM are typically trained in three distinct phases:

1) Phase 1: Foundation Pre-training. A base model is trained from scratch on a massive, general corpus of web data. The pre-training phase is designed to imbue a Foundation LLM with broad knowledge and linguistic capabilities. This stage uses self-supervised learning, where the model learns by predicting the next word in a sentence or filling in masked-out portions of text. The result is a foundational model with broad capabilities but no specific expertise in a specialized domain like networking. This stage is computationally intensive and is typically only performed by large AI research labs. For specialized applications like network management, this phase also includes Phase 2 below.

2) Phase 2: Domain-Adaptive Pre-training. This optional but highly recommended stage adapts a general foundation model to a specific domain. The objective is to make the base model "fluent" in the language and concepts of a specific field, such as networking. This involves familiarizing the model with specialized terminology, acronyms, and core principles. The general foundation model from Phase 1 undergoes a second round of self-supervised pre-training, but this time using a large, curated corpus of domain-specific data. For networking, this would include RFCs, vendor documentation, network engineering textbooks, technical blogs, and sanitized configuration data. The result is a domain-adapted model that has a much deeper "understanding" of networking concepts than the general base model. This model is now primed for the final stage of development: task-specific fine-tuning.

3) Phase 3: Once a model has been adapted to the networking domain through DAPT (Phase 2), the final stage is to teach it how to perform specific, practical tasks. This is achieved through task-specific fine-tuning, which uses supervised learning on a curated dataset of examples. The objective is to align the domain-adapted model's behaviour to a specific operational task, such as classifying network alerts, answering questions based on a runbook, or generating router configurations from natural language prompts. This is done by training the model on a high-quality, labelled dataset where each entry consists of an input and a desired output. For example, a specific syslog message is input, and it is desired for the LLM to determine the corresponding root cause of the alert. The result of the fine-tuning process is a specialized "expert" model that is highly proficient at its designated task.

   The recommended approach is Parameter-Efficient Fine-Tuning (PEFT), a set of modern techniques that addresses the drawbacks of full fine-tuning. The core idea is to freeze the vast majority of the pre-trained model's weights and only train a very small number of new, task-specific parameters.

   It is further recommended to use Low-Rank Adaptation (LoRA), a specific type of PEFT.

NOTE: It is recommended to start with LoRA+. This is a more robust and simpler starting point that often yields significant improvements over standard LoRA with much less tuning complexity. If further optimization is needed, then AdaLoRA is a good alternative, though it is more complex.

6.4.3 Model Evaluation and Optimization

6.4.3.1 Model Evaluation

Evaluating a Network Knowledge-Enhanced LLM (NKELLM) requires a multi-faceted approach that goes beyond standard NLP metrics. The evaluation needs to assess not only the model's performance on specific tasks but also its reliability, factual accuracy, and operational maturity.

A robust evaluation framework encompasses the following categories:

1) **Factuality and Grounding Metrics:** This is the most critical category for evaluating any model enhanced with external knowledge via RAG. These metrics measure the model's ability to base its responses on the provided source information and avoid hallucination. Exemplary metrics include:

   a) **Groundedness / Attribution:** This metric measures what percentage of the information in a generated response can be directly traced back to and supported by the retrieved source documents. A low groundedness score indicates the model is inventing information (hallucinating).

b) **Faithfulness:** This assesses whether the model's response accurately represents the information from the source documents without distortion or misinterpretation. A model can have a high groundedness score (it only used the source) but a low faithfulness score (it twisted the meaning of the source).

c) **Answer Relevance:** This measures how relevant the retrieved documents and the final answer are to the user's original query. This helps evaluate the performance of the retrieval component of the RAG pipeline.

2) **Service Quality Metrics:** These metrics evaluate the model's performance on its designated tasks. The choice of metric depends heavily on the application scenario. Some examples include:

a) For Classification Tasks (e.g. alert categorization): Standard metrics like Precision, Recall, and F1 Score are appropriate.

b) For Generative Tasks (e.g. summarization, report generation):

i) Lexical Metrics (e.g. BLEU, ROUGE): These measure the overlap of words between the generated text and a reference text. They are useful for a preliminary assessment but are limited as they do not capture semantic meaning.

ii) Semantic Similarity: Using another model to evaluate the semantic similarity between the generated output and a reference answer provides a more meaningful quality score.

c) For Question-Answering (Q&A) Tasks:

i) Exact Match (EM): Measures the percentage of answers that are identical to the ground truth. This is often too strict for complex answers.

ii) F1 Score: A more flexible metric that measures the word-level overlap between the prediction and the ground truth.

d) For Code Generation Tasks: Execution Success Rate. Is the most important metric—does the generated code (e.g. a device configuration script) execute without errors?

3) **Service Maturity Metrics:** These metrics assess the model's readiness for deployment in a production environment.

a) Robustness: How does the model perform when faced with noisy, malformed, or adversarial inputs?

b) Stability and Latency: Is the service consistently available? What is the end-to-end response time (from query to answer)? This is critical for real-time operational use cases.

c) Scalability: How does the system perform under increasing load (e.g. concurrent users or queries)?

4) **Human Evaluation:** Ultimately, automated metrics cannot fully capture the quality of an LLM's output. A structured human evaluation process remains the gold standard. This involves having domain experts (e.g. senior network engineers) rate the model's responses based on criteria such as:

a) Correctness and technical accuracy.

b) Clarity and usefulness.

c) Safety (i.e. does the model avoid suggesting dangerous or disruptive actions?).

6.4.3.2 Model Optimization

Model optimization includes model compression and the application of efficient inference frameworks. Model compression reduces the size and computational cost of the model through techniques such as pruning (removing unimportant parameters), quantization (converting parameters from high precision to lower precision), and knowledge distillation (training a smaller model to mimic a larger one). These methods help increase inference speed. At the same time, the use of efficient inference frameworks (such as TensorRT, ONNX Runtime, and OpenVINO) accelerates the inference process, optimizes computational performance, and reduces latency. Together, these measures enhance the efficiency and responsiveness of large models in real-world applications. Typical model optimization capabilities include:

1) Model Pruning

Model pruning is a technique used to reduce model size and computational load by removing unimportant parameters or neurons in a neural network. The process involves evaluating the importance of each parameter or neuron and determining which parts to remove. After pruning, the model is typically retrained to recover any potential loss in performance. This significantly improves inference efficiency and reduces memory usage, making it suitable for resource-constrained environments such as mobile devices and embedded systems.

2) Model Quantization

A technique that converts model parameters from high precision (e.g. 32-bit floating point) to lower precision (e.g. 8-bit integer). The main goal of quantization is to reduce memory consumption and computational cost while maintaining as much of the model's performance as possible, thereby accelerating the inference process.

3) Knowledge Distillation

A training method where a smaller "student" model learns from the "soft labels" (i.e. the probability distributions) output by a larger "teacher" model, rather than relying solely on hard labels (true labels). This allows the student model to capture deeper features and decision boundaries from the teacher model, thereby improving its generalization ability.

4) Inference Frameworks

Tools such as inference frameworks are used to accelerate model inference and improve the performance of large models (e.g. TensorRT, ONNX Runtime, OpenVINO). Efficient inference frameworks are tools and libraries designed to optimize the computation of machine learning and deep learning models during inference (i.e. prediction or classification). These frameworks enhance inference speed and reduce latency by optimizing computation graphs, leveraging hardware capabilities, implementing quantization, and enabling parallel processing.

6.5 Deployment and Inference: Operationalizing a RAG System

Once a model is trained, deploying it for inferencing requires a robust architecture that can provide timely, accurate, and secure responses. While RAG is the primary architectural pattern for ensuring responses are grounded in up-to-date knowledge, moving from a prototype to a production-ready system involves significant engineering challenges. A successful deployment needs to address the following operational considerations:

1) **Knowledge Lifecycle Management:** A production knowledge base is not a static asset; it is a dynamic system that is required to accurately reflect the current state of the network. This requires a comprehensive strategy for managing the entire lifecycle of knowledge, which goes far beyond simply adding new documents:

    a) **Data Ingestion and Updates:** The system needs automated pipelines to ingest data from various sources (e.g. monitoring tools, configuration management databases, document repositories). More importantly, it needs to handle updates to existing information. For example, when a router's OS is upgraded, the system is required to find and update the corresponding entity in the knowledge base.

    b) **Data Deletion and Archiving ("Unlearning"):** When a device is decommissioned or a network topology changes, the corresponding information needs to be removed or archived from the knowledge base. Simply leaving outdated information in the vector store is a common failure mode, as it can be retrieved by the RAG system and lead to incorrect answers based on a state that no longer exists. A robust deletion and versioning strategy is critical for maintaining the system's reliability.

2) **Security and Access Control:** In any enterprise environment, not all users require access to all information. Implementing effective security within a RAG system is a non-trivial architectural challenge:

   a) **Role-Based Access Control (RBAC) in Retrieval:** The system needs to enforce access controls at the retrieval stage. This means that when a user asks a query, the retrieval mechanism needs to only return documents and data that the user is authorized to see.

   b) **Implementation Strategy:** A secure approach involves embedding access control metadata directly with the data chunks in the knowledge base. The retrieval pipeline is required to be designed to be "RBAC-aware," filtering its results based on the user's authenticated role and permissions before the information is passed to the LLM. This prevents sensitive data from ever reaching the model context for an unauthorized user.

3) **User Feedback and Validation Loop:** While user feedback is invaluable for improving the system, creating a fully automated feedback loop is a significant risk:

   a) **The Risk of Poisoning:** Allowing user feedback to "trigger automatic knowledge base corrections" creates a vector for poisoning the knowledge base with incorrect information, whether accidental or malicious.

   b) **Human-in-the-Loop Validation:** A production-grade system needs to implement a human-in-the-loop validation workflow. When a user flags an answer as incorrect or suggests a correction, that feedback is routed to a queue for review by a domain expert. The correction is committed to the knowledge base only after an expert has validated it. This ensures the integrity and accuracy of the system's knowledge over time.

Addressing these operational challenges is essential for transforming a promising NKELLM prototype into a trusted, reliable, and secure tool for network operations.

7 Application scenarios

7.1 Knowledge Q&A

Network operation and maintenance managers need to understand the knowledge of network operation management and maintenance when conducting network monitoring, alarm identification, and fault handling, etc., or when customer service staff receive customer inquiries and product orders, they need to understand the relevant knowledge of products or network maintenance management. The conventional method is accomplished by consulting various online and offline technical materials or consulting experienced operation and maintenance experts. Based on the induction, organization, and generation capabilities of the large model, the knowledge Q&A service for network operation management can provide efficient, comprehensive, and accurate answers.

The trained network LLMs (or encapsulated service) can form an independent APP alone to provide services externally, or this service can be embedded in the traditional network operation management system, presenting as a menu function of the management system. When the network operation and maintenance manager invokes this function (i.e. the network operation and maintenance manager raises a question), an answer will be provided as feedback.

The following evaluation metrics can be selected to assess the performance of a knowledge Q&A system.

Table 1: Evaluation metrics for knowledge Q&A

Metric	Description
Exact Match (EM)	Whether the predicted answer exactly matches the reference answer
F1 Score	Overlap between predicted and true answers; balances precision and recall
Answer Correctness	Evaluates whether the final answer generated by the LLM is factually and semantically correct, regardless of its specific wording. It answers the question: "Is the information presented in the answer true and does it accurately address the user's question?
BLEU / ROUGE / METEOR	Measures text similarity between generated and reference answers
Recall@k / MRR / Hit@k	Evaluates retriever performance in retrieval-augmented QA
Precision@k	Measures how many retrieved documents were relevant
Hallucination Rate	Percentage of generated facts that are ungrounded or false

Metric	Description
Faithfulness	Measures whether the generated answer accurately represents the information in the retrieved context
Answer Latency	Time taken to generate a response
Human Evaluation	Ratings of answer correctness, fluency, and trustworthiness

7.2 Content recommendation

When customer service staff (marketing staff) recommend and promote business to customers, they need to understand customer characteristics and product attributes, etc. The conventional method is accomplished by customer service staff matching products or content by asking about customer needs. Based on the comprehensive perception and comprehensive analysis capabilities of the large model, the content recommendation service for network operation management can provide precise product recommendations to customers without the need for or with as little interaction with customers as possible, such as communication package recommendations.

The trained network LLMs can be embedded in the traditional customer service management system as an external service provided by the customer service management system. When customer service staff invoke this service, the system provides the recommendation results. Or this service can form an independent APP (such as "digital customer service") alone to provide services externally. This service can directly face customers and make recommendations to customers based on customer characteristics.

The following evaluation metrics can be selected to assess the performance of the content recommendation system.

Table 2: Evaluation metrics for content recommendation

Metric	Description
CTR (Click-Through Rate)	Clicks divided by impressions
Precision@k / Recall@k	Accuracy of top-k recommended items
NDCG (Normalized Discounted Cumulative Gain)	Measures ranking quality considering item positions
MAP (Mean Average Precision)	Averages precision across ranked lists
Coverage / Diversity / Novelty	Measures how broad, varied, or fresh the recommendations are
User Engagement Metrics	Includes dwell time, conversion rate, bounce rate, etc.
Conversion Rate	What percentage of recommendations lead to an actual purchase?
Average Revenue Per Recommendation	This measures the financial impact directly.
Customer Lifetime Value (CLV) Uplift	Does accepting a recommendation lead to a higher overall customer lifetime value?
Demographic Parity	Does the system recommend high-value products equally across different customer demographics (e.g. age, location), or is it biased?
Exposure Fairness	Are all relevant products (e.g. plans from different partners, or both pre-paid and post-paid options) getting a fair amount of exposure, or is the system biased towards a few popular items?

7.3 Prediction

When network operation and maintenance managers conduct network monitoring and network maintenance, they can predict the network operation status at a future time point or within a certain period of time based on the current network operation status, such as network traffic prediction and fault prediction, and prevent network deterioration or network faults in advance through early prediction. The conventional method is to make predictions based on historical data over a period of time and the corresponding prediction algorithm. Based on the comprehensive perception and comprehensive analysis capabilities of the large model, the prediction service for network operation management can provide precise predictions based on a wider range of various types of data and stronger cross-domain analysis capabilities.

The trained network LLMs (or encapsulated service) can be embedded in the traditional network operation management system. At this time, the large model presents as a background prediction algorithm of the system. Based on the comprehensive perception and comprehensive analysis of the information as needed or periodically, this algorithm provides the required prediction results.

The following evaluation metrics can be selected to assess the performance of the prediction system.

Table 3: Evaluation metrics for prediction

Metric	Description
Accuracy	Proportion of correctly predicted samples
Precision / Recall / F1 Score	Key metrics for evaluating classification performance
ROC-AUC	Measures the ability to distinguish between classes
MAE / RMSE / MAPE	Error metrics for regression/forecasting tasks
Confusion Matrix	Visualizes true vs. predicted classifications

7.4 Content generation

When network operation and maintenance managers conduct network optimization, fault handling, and network maintenance, they need to generate various specific execution plans based on the actual status such as network topology and configuration, network operation status, problem diagnosis, and surrounding environment, such as network optimization plans, network fault handling plans, and network operation and maintenance inspection plans. The conventional method is to generate execution plans based on certain rules, operation and maintenance tools, and the experience of network operation and maintenance managers. Based on the comprehensive perception, comprehensive analysis, and autonomous generation capabilities of the large model, the content generation service for network operation management can generate corresponding execution plans based on various types of data, guarantee requirements, experience and rules.

The trained network LLMs (or encapsulated service) can be embedded in the traditional network operation management system, for example, embedded in the network optimization system. At this time, the large model presents as a background optimization plan generation algorithm in the network optimization system. When a certain deterioration occurs in the network, based on the judgment of the problem, comprehensive perception of the information, comprehensive analysis, and generation capabilities, this algorithm can autonomously generate the corresponding network optimization solution.

The following evaluation metrics can be selected to assess the performance of the content generation system.

Table 4: Evaluation metrics for content generation

Metric	Description
BLEU / ROUGE / METEOR	Measure similarity to reference content (e.g. in summarization)
Perplexity	Measures how fluent or confident the model is in its generations
Diversity / Distinct-n	Assesses the variety of generated outputs
Toxicity / Bias Score	Checks for harmful, biased, or unsafe content
Human Evaluation	Judges the creativity, relevance, and readability
Code Execution Accuracy	For code generation: whether the generated code runs correctly
Groundedness/Attribution	What percentage of the claims in the generated report can be directly verified from the source data provided to the model?
Faithfulness	Does the generated content accurately represent the source data without distorting or misinterpreting it?
Constraint/Policy Adherence	If the model generates a configuration script, does it adhere to the organization's security policies, standard syntax, and operational best practices?

8 Summary and Recommendations

8.1 Summary

The present document describes Knowledge-Enhanced Network Large Language Models, focusing on their service capabilities, training processes, knowledge enhancement methods, evaluation metrics, and optimization techniques. It aims to address key challenges of general LLMs in network scenarios - such as knowledge gaps, hallucinations, and reasoning limitations - by improving domain expertise, real-time responsiveness, and practical deployment efficiency, ultimately supporting intelligent and automated network operations.

8.2 Recommendations

1) Multimodal Data Fusion: In future model training, further integrate multimodal data (such as text, audio, time series and traffic data).

2) Enhanced Fault Prediction and Prevention: Enhance the model's predictive ability on historical network data and further develop mechanisms to prevent faults in real-time. Shift from "passive response" to "proactive prevention", enhancing network stability and operational efficiency.

3) Openness: the use of open AI models, tools, and datasets. Enhance user trust in the model and promote its application in critical network operation scenarios.

4) AI Detection of Faults: Use declarative rules to process faults and alarms.

Annex B:
Change history

Date	Version	Information about changes
2023-10-13	0.0.1	The first draft
2023-10-31	0.0.2	Add some contents for clause 4 and clause 5.1
2023-12-14	0.0.3	Add some contents for clause 5 and clause 6
2023-12-14	4.0.4	Modify some contents for clause 6
2024-03-05	4.0.5	Modify some contents for clause 4, 5 and 6
2024-04-09	4.0.6	Add the contents of clause 5.2.3
2024-04-17	4.0.8	Modify some contents of clause 5.2.3
2024-06-05	4.0.9	Add more contents for clause 5.2.1 and clause 5.2.2
2024-09-10	4.0.10	Add and modify the contents for the whole document
2024-09-16	4.0.10	Modify some contents for clause 4, 5 and6
2024-10-21	4.0.11	Modify some contents for clause 4, 5 and 6
2024-10-29	4.0.12	Modify some contents for clause 4, 5 and 6
2024-11-05	4.0.12	Modify some contents for clause 4, 5 and 6
2024-12-12	4.0.13	Modify some contents for clause 4,5 and 6
2025-03-04	4.0.13	Modify some contents for clause 4, 5 and 6
2025-03-06	4.0.14	Adjusted the content of the entire manuscript section
2025-03-06	4.0.15	Modify some contents for clause 4, 5 and6 based on some comments
2025-04-19	4.0.16	Modify all contents based on the new comments
2025-05-18	4.0.16	Modify all contents based on the new comments
2025-05-27	4.0.16	Modify some contents based on the new comments
2025-06-13	4.0.17	Modify some contents based on the new comments
2025-06-17	4.0.18	Modify some contents based on the Technical_Review
2025-07-09	4.1.18	Modify some contents based on the new comments
2025-07-29	4.1.19	Modify some contents based on the Technical_Review to make it stable

History

Document history		
V4.1.1	December 2025	Publication