Computer Science and Information Systems 23(1):535–560 https://doi.org/10.2298/CSIS250416082T

Enhancing Network Engineering Capabilities through LLM Fine-Tuning with Automatically Generated Datasets ⋆

Claudiu Trăistaru[1], Florin Pop[2], Costin Bădică[3], Cătălina Mancaş[3], and Ionuţ Murareţu[3]

[1] University of Science and Technology POLITEHNICA Bucharest, University of Craiova, Craiova, Romania
claudiu.traistaru@edu.ucv.ro
[2] University of Science and Technology POLITEHNICA Bucharest, National Institute of Research and Development in Informatics (ICI), and Academy of Romanian Scientists, Bucharest, Romania
florin.pop@upb.ro
[3] Computer and Information Technology Department, University of Craiova, Craiova, Romania
{costin.badica,catalina.mancas, ionut.muraretu}@edu.ucv.ro

Abstract. The paper presents a method for automatically generating domain-specific datasets to fine-tune open-source LLMs in network engineering. Our objective is to address the increasingly complex nature of network configuration and management jobs by supplying LLMs with high-quality training data. We evaluated datasets generated using open-source LLMs, including DeepSeek-R1 671B, LLaMA 3.1 70B, Qwen 2.5 72B, and Mixtral 8x7B, analyzing the quality of unprocessed knowledge data and the efficacy of cleaning and deduplication methods. The resulting dataset addresses various subjects related to routing, security, and network services. Afterward, we fine-tuned smaller LLaMA 3.2 1B, LLaMA 3.2 3B and Qwen 2.5 1.5B models using Low-Rank Adaptation, thereby minimizing computational demands while maintaining the quality of domain knowledge.

1. Introduction

Network engineers are essential to the management and functioning of computer networks, as their choices directly impact the transmission and security of data within these complex systems. Historically, computer networks have facilitated diverse functions, ranging from basic communications like emailing to intricate processes such as cloud computing, online gaming, and the seamless integration of Internet of Things (IoT) devices. The main goal of these systems is to efficiently direct data traffic from source to destination, which is accomplished by properly setting up network devices and services.

Various network components, such as switches, routers, network interfaces, and network services, are part of the configuration process. Current computer networks have grown more complex, requiring a deep understanding of topologies, protocols, and the specific demands of each network. To guarantee strong data security, network engineers must deploy and maintain essential tools, including intrusion detection systems, firewalls,

⋆ This paper is a significant extension of our preliminary work presented in [22]

and safe Access Control Lists (ACLs). Recent research shows how natural language processing (NLP) can be used to automate and validate network designs by employing network parsers as specialized tools that convert network configurations into structured intermediate representations such as Abstract Syntax Trees [2].

The complexity of these systems demands novel strategies for network management. Expert systems supported by domain-specific Large Language Models (LLMs), especially open-source variants, are becoming crucial for assisting network engineers in network engineering problem solving like for example implementing automated network settings and improving network management. LLM-based approaches could allow network engineers to address the increasing difficulties of performance optimization, cybersecurity, and IoT integration while minimizing the manual burden usually linked to network configuration and management.

In this context, tuning LLMs using high-quality, domain-specific datasets proves crucial for enhancing their effectiveness in network engineering. The goal of our research effort is to develop new methodologies for the efficient creation of such datasets by leveraging existing domain-specific literature and synthetic data generation techniques. These datasets are usually structured as sets of pairs comprising questions and answers related to problem solving in network engineering. The goal of our work is to enhance LLM applications in network administration, contributing to the advancement of computer networks to ensure their security, efficiency, and ability to satisfy the requirements of a progressively networked world.

According to [4], Reinforcement Learning with Human Feedback (RLHF) is a machine learning (ML) technique that leverages human feedback to improve ML models so that they can learn more efficiently. RLHF uses rewards based on human feedback so that the model can perform tasks better aligned with human goals. RHLF is a standard technique that ensures that LLMs produce truthful, helpful and harmful content. Our research aims to create network engineering datasets by minimizing the need for RLHF, which is costly, time-consuming and error-prone. The need to use generated datasets to fine-tune LLMs for more specific fields is growing, and being able to generate this kind of dataset is our solution to this need.

We envisage a network engineering assistant acting like a human expert in the field. To develop such an expert assistant in network engineering using publicly available models (i.e., whose weights can be freely downloaded and fine-tuned), we will investigate the feasibility of generating training datasets based on a sizable corpus of texts and also the use of these datasets to fine-tune smaller versions of open-weight models, such as LLaMA 1B and LLaMA 3B.

For example, the expert assistant can assist network engineers in designing configurations of switches and routers that control the routes along which data packets travel over the network, guaranteeing effective and safe data transfer. Among other things, these configurations must consider fault tolerance, traffic load distribution, and network topology. Modern networks can virtualize network functions to provide services like firewalls, load balancers, or intrusion detection systems, but these must be correctly set up.

The development of open-source LLM models has significantly fueled the availability of advanced artificial intelligence technologies, stimulating innovation and fostering diversity across various sectors. Unlike their proprietary equivalents, these models allow the scientific community and developers to investigate, alter, and enhance them. Open-source

LLMs boost research and development and allow us to improve them to achieve better results.

In the quest for Artificial General Intelligence (AGI), the growing availability of both closed and open-source LLMs represents a significant turning point. LLMs have progressed gradually toward reaching AGI by passing through several evolutionary stages. This development has resulted in abundant publicly available and proprietary LLMs, adding to the diverse environment of AI-driven language creation. In this paper, we propose a new method to automatically produce a varied and relevant dataset that can be used to fine-tune open-source LLMs to increase their performance in the field of network engineering.

Our study investigates how open-source models may be improved by fine-tuning using specialized data previously selected and organized into distinct, well-chosen and well-designed datasets.

We extended the data generation methodology presented in [22] by the usage of the following models: DeepSeek-R1 671B, LLaMA 3.1 70B, Qwen 2.5 72B, and Mixtral 8x7B, representing the junction of innovation, accessibility, and adaptation. By tackling fundamental difficulties like computing efficiency, scalability, and domain-specific adaptation, these models enable academics and practitioners to create transformational NLP applications. As the field evolves, the contributions of these models will indeed affect the course of future improvements, ensuring that the advantages of LLMs are both broad and inclusive. The iterative nature of their creation assures that these models are always current, constantly evolving in response to new challenges and possibilities. Afterward, we used the cleaned generated data to fine-tune LLaMA 3.2 1B, LLaMA 3.2 3B and Qwen 2.5 1.5B models.

The paper is structured as follows. In Section 2, we provide an overview of the domain presenting related work of LLM in the network engineering domain. Section 3 depicts the data collection and processing methodology, the evaluation methods and the used metrics. The experimental setup and the results are discussed in Section 4. Finally, we present some conclusions in Section 5.

2. Related Work

2.1. Open-Source Large Language Models

The Open Pre-trained Transformer (OPT) design efficiently uses transformer-based pre-training to provide coherent, contextually appropriate outputs, making it a key component in generalizable NLP applications. A model's emphasis on open-source accessibility promotes repeatability and community-driven innovations, resulting in significant gains in adaptability and applicability across diverse areas. [25].

The LLaMA [20] models aim to reduce the size of the parameter space without sacrificing performance. LLaMA produces high-quality results while being computationally accessible using powerful tokenization approaches and stringent dataset filtering. This model's capacity to thrive in zero-shot and fine-tuning scenarios demonstrates its adaptability, making it ideal for academics and institutions with limited computational resources. Furthermore, LLaMA's efficient architecture design allows it to scale successfully across applications that require dynamic adaptation, such as real-time translation

systems and adaptive content production [21]. Its parameter efficiency also makes it a good choice for inclusion in systems that require scalability and quick deployment.

Mixtral is a Sparse Mixture of Experts (SMoE) language model created by Mistral AI. The design comprises eight unique expert networks, amounting to 47 billion parameters, of which only 13 billion are utilized during inference to enhance computational efficiency, in this case it means only 2 experts are selected per token during inference, minimizing the memory usage.

This approach enables Mixtral to surpass models such as Llama 2 70B and GPT-3.5 across multiple benchmarks, particularly excelling in mathematics, code creation, and multilingual activities. [9].

Phi-4 is a small language model (SLM) with 14 billion parameters, developed by Microsoft, intended to excel in intricate reasoning tasks, including mathematics and coding. In contrast to numerous models that predominantly depend on organic data sources, Phi-4 deliberately integrates synthetic data during its training, resulting in improved performance in STEM-oriented question-answering abilities. Significantly, Phi-4 exceeds its instructor model in these domains, illustrating the efficacy of its data-generation and post-training methodologies. [1].

DeepSeek-R1 is an open-source large language model created by DeepSeek, a Chinese artificial intelligence firm. Launched in January 2025 under the MIT License, it is engineered to excel at logical inference, mathematical reasoning, and real-time problem-solving tasks. DeepSeek-R1 utilizes reinforcement learning methodologies to augment its reasoning abilities. Significantly, it was created at a substantially lower cost than its rivals, highlighting China's increasing competitiveness in AI development. [5].

Qwen 2.5 is an extensive collection of large language models (LLMs) created by Alibaba Cloud, intended to address various requirements. Qwen 2.5 has undergone substantial enhancements in both the pre-training and post-training phases compared to earlier versions. [18].

Together, these models represent the open-source LLM ecosystem's rising diversity and specialization. Mixtral and Phi-4 address domain-specific difficulties with specialized innovations. This variety of capabilities demonstrates open-source models' ability to serve various applications, from broad, generalist tasks to highly specialized sectors. Furthermore, the community-driven collaborative mindset around these models has reduced barriers to entry for researchers globally, creating an atmosphere in which innovation is driven by shared information and collective achievement. This culture of openness has sped progress, ensuring breakthroughs are quickly communicated and implemented in academic and industry settings.

High-performing models have sparked a global research movement, democratizing cutting-edge NLP technology. The repeatability and accessibility of open-source models have opened up new avenues for research, resulting in achievements in fields ranging from computational linguistics to real-time machine learning applications. These models offer a solid framework for multidisciplinary research, enabling applications in social science, legal analysis, and biological research [25].

2.2. Role of Datasets in Fine-Tuning

Datasets are necessary for fine-tuning LLMs, acting as the key connection between their extensive pretraining and the specific requirements of target applications. Pretraining on

extensive, general-purpose corpora equips open-source models like LLama, Qwen, and Phi with exceptional generalization abilities. At the same time, fine-tuning facilitates their adaptation to domain-specific tasks, enabling applications in healthcare, finance, and legal services.

The success rate of fine-tuning can be significantly affected by the dataset's properties, such as size, quality, and diversity. The quantity of the dataset is essential as it dictates the extent of language patterns and domain-specific knowledge that a model can assimilate. Larger datasets often enhance coverage and generalization; nevertheless, overly extensive datasets may result in redundancy, causing decreasing returns and heightened computing expenses. Achieving a good equilibrium between dataset size and task relevance is crucial for enhancing efficiency and performance.

In general, high-quality datasets are essential for ensuring that models acquire accurate and consistent patterns. Dataset quality is defined by low noise, consistent annotations, and applicability to the specific area. A well-curated dataset enables the model to concentrate on obtaining task-specific knowledge, free from the influence of extraneous or erroneous information. The preprocessing stages, including deduplication, noise filtration, and context enhancement, are essential for meeting these quality goals. The influence of datasets on fine-tuning results is significant. Well-curated datasets facilitate models in attaining enhanced accuracy, equity, and interpretability [21]. The success of LLaMA underscores the significance of dataset quality and diversity in enhancing fine-tuning effectiveness.

While the method of collecting and selecting high-quality training data is somewhat complicated, the basic ideas of creating a dataset are relatively straightforward. The first step is to choose the problems that you want our model to solve. After that, you must create a dataset demonstrating these behaviors. Many teams are increasingly designating specific responsibilities to handle the procurement of acceptable datasets while maintaining compliance with privacy standards as they recognize the importance of data.

Pre-training requires different and considerably larger datasets than instruction fine-tuning or preference fine-tuning. Even with these modifications, the fundamental concepts of dataset design, i.e., ensuring quality, obtaining acceptable coverage, and preserving sufficient quantity, stay the same across all training stages.

2.3. Challenges in Dataset Design for Fine-Tuning Large Language Models

Using RLHF to fine-tune language models demonstrates that iterative, human-centered approaches significantly enhance dataset quality and model alignment with real-world applications. This methodology leverages the value of integrating human judgment directly into the data creation pipeline [16]. Including clear and precise instructions for annotators proved instrumental in reducing inconsistencies and streamlining the annotation process. By embedding iterative feedback mechanisms, datasets can undergo continuous refinement, ensuring they remain relevant, reliable, and effective in enabling LLMs to meet specific task requirements.

A critical role of task-specific instructions and illustrative examples in guiding annotators to provide explicit objectives and contextually relevant examples ensures a shared understanding of goals. This is especially vital for complex domains, such as networking tasks, where precision and contextual sensitivity are paramount. Extending these prac-

tices to dataset design for LLM fine-tuning facilitates the creation of high-quality datasets capable of supporting nuanced and domain-specific applications.

The design of datasets for fine-tuning LLMs is a complex but indispensable process that requires addressing significant challenges, including annotation complexity and privacy compliance. By leveraging human feedback mechanisms and aligning to ethical principles [16], researchers and practitioners can create datasets that are high-quality, reliable and aligned with societal values. These advancements are essential for harnessing the full potential of LLMs in a manner that respects both technical rigor and ethical integrity.

2.4. Applications of Large Language Models in Network Engineering

Incorporating refined LLMs into network engineering signifies a fundamental change and actionable knowledge. LLMs have become essential instruments in contemporary network engineering by connecting human purpose with technological execution.

A primary use of LLMs is in automated configuration management, where their capacity to comprehend human-readable configuration requests and convert them into accurate, deployable network configurations has been transformational. [8] elucidates the capacity of LLMs to optimize network configuration procedures, significantly reducing the need for manual involvement. This not only reduces mistakes but also expedites deployment time-frames. Moreover, [15] illustrates how LLMs may generate precise and resilient router configurations from abstract, high-level requirements. These qualities are especially beneficial in extensive and diverse networks, where uniformity and scalability are essential. By generalizing across many networking circumstances, LLMs may adjust to new technologies, suggest optimum setups, and improve the overall agility of network operations.

Security diagnostics constitute another transformational area in which LLMs have demonstrated considerable potential. One approach involves using context-aware, iterative prompting to help LLMs identify and diagnose router misconfigurations [10]. Misconfigurations, a popular source of network vulnerabilities, can diminish performance or render systems susceptible to possible security breaches. LLMs can effectively discover abnormalities and possible dangers by evaluating logs, configurations, and traffic patterns. The incorporation of predictive analytics into LLM procedures facilitates the forecast of potential threats through historical data patterns, allowing for preventative measures to strengthen network defenses and proactively minimize risks.

A popular task is the creation of network management scripts. LLMs can generate task-specific scripts for functions like automatic improvement of device configurations, dynamic traffic routing, and diagnostic assessments [14]. This functionality improves operational processes by automating redundant jobs and enabling quick reactions to changing network circumstances. LLMs may dynamically modify routing pathways to mitigate bottlenecks or execute load-balancing algorithms in real time, dramatically enhancing network speed. Furthermore, by including LLMs in continuous integration and deployment (CI/CD) pipelines, developers can guarantee that created scripts undergo thorough testing and validation, minimizing downtime and improving operational dependability.

Telecommunication networks, known for their complexity and scale, have also gained advantages from the particular characteristics of LLMs. As introduced by [13], Tele-LLMs is a collection of fine-tuned models tailored for telecommunication-specific functions, including performance monitoring, problem diagnosis, and capacity planning. These

models utilize domain-specific data to detect network irregularities and forecast possible failures, enabling engineers to take proactive remedial actions. The predictive skills of Tele-LLMs encompass the optimization of 5G deployments and edge computing infrastructures, where real-time decision-making and agility are essential. By examining historical performance trends and real-time measurements, Tele-LLMs empower telecom operators to improve service dependability and efficiency.

LLMs' adaptability to various networking tasks is further demonstrated in [24], which introduces the NetLLM framework. This framework customizes LLMs for networking applications, enabling them to comprehend and produce domain-specific instructions and settings. NetLLM demonstrates the advantages of domain-specific fine-tuning by integrating networking terminology and contextual knowledge into its models. This guarantees that NetLLM can adeptly manage intricate scenarios with accuracy, including the orchestration of multi-vendor systems and the administration of hybrid network environments. The modular architecture facilitates incremental enhancements, guaranteeing sustained flexibility and efficacy.

The integration of LLMs in network engineering presents significant advantages, although it requires the adequate addressing of specific challenges like data privacy, ethical issues, and regulatory compliance. Huang et al. (2023) underscore the necessity of resilient governance structures to alleviate risks linked to automated decision-making and to guarantee conformity with industry norms. Collaboration among stakeholders, such as researchers, policymakers, and industry executives, is crucial for formulating rules that reconcile innovation with responsibility.

3. Methodology

Before diving into the specific activities, we provide a comprehensive overview of the data generation pipeline, as illustrated in Figure 1. The procedure begins with the acquisition of technical documentation from reliable sources. The raw data is subsequently curated, chunked, vectorized, and indexed into a vector database. Thereafter, relevant portions are extracted and processed, utilizing prompting techniques to generate Q&A pairs based on chosen network engineering topics. These are further automatically checked, deduplicated and sanitized to guarantee quality and validity. Finally, the curated dataset is ultimately used for fine-tuning.

3.1. Data Collection

The initial phase of our methodology entails the acquisition of relevant textual data from various data sources for the field of network engineering, including relevant literature, websites, and repositories comprising, for example, openly accessible volumes from the Network Solution Vendor's repository, specifically the Day One publications [12], supplemented by additional document corpus K documentation. The textual database consists of 79 books.

These textual resources must undergo vectorization, indexing, and subsequent storage within a vector database for practical use. The initial stage in developing such datasets is to collect information from various reliable sources. Relevant sources for network engineering include technical documentation, online manuals with setup instructions, research papers, and community forums like Network Engineering Stack Exchange and

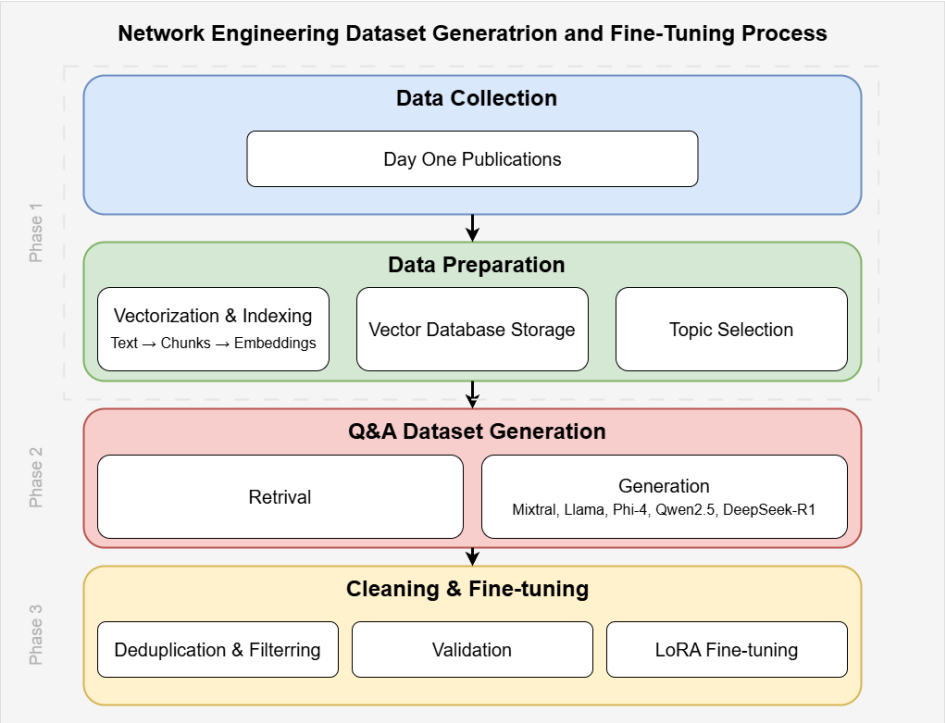

Fig. 1. Data Processing Diagram

Reddit. The development of the LLaMA model demonstrates this approach by integrating domain-relevant information acquired from sources GitHub repositories and Stack Exchange conversations to ensure high-quality and domain-specific content coverage [21]. Another similar effort is provided by NVIDIA research guidelines that encourage getting data from organized, high-quality sources generally underrepresented in common pretraining corpora [17].

Some studies emphasize the necessity of bias reduction and openness across the data process. When obtaining data from public forums or repositories, anonymizing sensitive material and documenting dataset provenance is critical for reducing bias and assuring trustworthiness. Frameworks such as ”Datasheets for Datasets” [7] offer systematic methods for describing qualities and constraints, promoting responsibility and confidence [17].

3.2. Data Processing Pipeline

Creating a comprehensive dataset comprising network engineering knowledge necessitates close attention to relevance, quality, and variety. As recent research has shown, creating suitable datasets requires a systematic data collection, curation, and assessment pipeline to guarantee that LLMs can generalize well and provide the appropriate domain-specific knowledge.

Data preparation involves the following steps: curation, chunking, vectorization & indexing, retrieval, and generation.

Data curation is critical for improving the dataset quality. Techniques like deduplication, noise reduction, and quality filtering are critical for maintaining consistency and focus. The LLaMA pipeline used deduplication and language identification to eliminate unnecessary or duplicated information and maintain dataset integrity [21]. Heuristic-based quality filters, such as sentence structure and the use of technical language based on the proper terminology, help to improve dataset relevance. This is especially important in network engineering, where correctness in vocabulary, technical clarity and precision are essential.

The diversity of datasets has a substantial influence on model robustness and fairness. Effective datasets include many network engineering subjects, including routing protocols, troubleshooting, and security setups. Given the field's worldwide scope, incorporating bilingual content is essential. Furthermore, gathering historical and contemporary data ensures that the dataset reflects changing trends in network technology [17].

Data chunking assumes breaking down and dividing the acquired documents into digestible, searchable chunks, also known as snippets, that are subsequently examined in more detail. The resulting chunks have a size of 1500 characters and an overlap of 200 characters. The text splitter uses a recursive algorithm to determine the best way of splitting the text in order to streamline the semantic flow between consecutive chunks. One approach that improves both chunking and vectorization entails the utilization of NLP capabilities, mainly through instruction-based prompts. These prompts direct the language model to produce task-specific embeddings customized for the desired application, such as information retrieval, semantic similarity, classification, or text assessment. This NLP-driven technique uses dynamic prompting to build embeddings that reflect the semantic subtleties pertinent to the job, unlike typical embedding methods that yield a singular, static vector for each input. This method enhances downstream performance by matching the created embeddings with particular use-case specifications.

Data vectorization and indexing assumes the conversion of data chunks into vector embeddings, as well as their storage and indexing into an appropriate vector database. Vector embeddings, as numerical representations of discrete or symbolic data points, prove indispensable for conducting similarity-based inquiries within artificial intelligence and machine learning frameworks. The indexing represents a fundamental transformation process, converting unstructured content into systematically organized, searchable excerpts. After a thorough analysis, these excerpts are encoded into a high-dimensional space and kept in a vector database. This configuration produces a repository that speeds up data association and retrieval. Also, Vector Databases are useful in addressing shortcomings, like hallucinations and outdated information. This makes them optimal for LLM integrations to provide more precise, current information in domain-specific contexts. [11]

Usually, the cosine similarity measure is used to compute the semantic similarity of two vectors stored in the vector database. This method allows us to compare symbolic concepts by computing the angle between their vector embeddings to estimate their semantic closeness. This measure quantifies the degree of alignment of the vectors, i.e., how well they point in the same direction. So, instead of just looking at the dot product, cosine similarity proposes a normalized quantity that brings a clearer idea of the degree of

similarity of two vectors, as described by Equation 1.

$$S_c(\mathbf{a}, \mathbf{b}) = \cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{1}$$

The research done in [19] shows a method to improve the efficacy of instruction-based fine-tuning. Possible applications are information retrieval, semantic similarity, classification, or text evaluation. By training on 330 diverse tasks with human-curated instructions, the method adapts its representations on the fly without needing additional fine-tuning for each new task. Extensive evaluations on 70 datasets demonstrate an average improvement of 3.4% over previous techniques, highlighting the effectiveness of instruction-based fine-tuning in producing versatile and robust text embeddings [19].

Data retrieval is responsible for finding and extracting relevant information from a vector database in accordance with a specified topic. During our dataset production process utilizing RAG (Retrieval-Augmented production), established topics are employed to query the vector database, extracting the most pertinent text segments to provide precise and contextually informed response generation.

It uses query processing algorithms that analyze and translate user input into relevant embeddings, leveraging the structured data obtained in the vectorization and indexing step.

The retrieval process searches and retrieves the most relevant document chunks by comparing these queries, a high-dimensional vector that semantically encapsulates the chosen topic.

The query embeddings are subsequently compared to pre-indexed document embeddings retained in the vector database. The indexing process preceding retrieval entails vectorizing and structuring the document corpus to facilitate rapid and precise similarity searches. The query embeddings are then compared to the pre-indexed document embeddings stored in the vector database. The indexing procedure, before retrieval, involves vectorizing and organizing the document corpus to enable swift and accurate similarity searches.

Data generation compiles and tunes the recovered document segments to be more coherent and informative (see the example from Figure 2).

The costly generative method is based on prompting techniques utilized on the recovered document segments. These prompts direct the language model to synthesize information and to produce logical, contextually aware queries and responses. This phase uses the model's ability to convert input data into significant outputs by blending many data points into a cohesive and useful outcome. The resultant question-answer pairs are subsequently consolidated into a newly generated dataset.

The workflow of the generating process is outlined by Algorithm 1. This algorithm uses an initial vectorized document corpus K and a mix of retrieval and generative techniques. It returns a large and comprehensive dataset D that complies with the predetermined specifications of the topic T, the document corpus K, the number of documents N that are retrieved, and the generative steps from 1 to S.

The algorithm starts with initializing the generator G using the appropriate generative model M and the specific topic T. These are essential components of data synthesis and sourcing.

Then, the algorithm systematically searches the relevant resources in the vector database K that comply with the topic T. This search is guided by a scoring system that evaluates

Algorithm 1 Retrieval and generation

Require: Topic T
Corpus of vectorized documents K
Generative model M
Number of documents to retrieve N
Number of generating steps S
Ensure: Generated dataset D
Initialize generator $G(M, T)$
$D \leftarrow \emptyset$ {Set of generated data}
$DocScores \leftarrow \emptyset$
for each $doc \in K$ matching T with $score$ **do**
  $DocScores \leftarrow \{(doc, score)\} \cup DocScores$
end for
for $s = 1, S$ **do**
  $TopDocs \leftarrow$ top N documents from $DocScores$
  $DatasetPrompt \leftarrow concatenate(T, TopDocs)$
  $GeneratedData \leftarrow G.generate(DatasetPrompt)$
  $D \leftarrow D \cup GeneratedData$
end for
return D

how well each retrieved document matches the topic T. The documents with the highest relevance are prioritized by selecting the top N that are the most relevant.

Afterward, the selected topic-specific documents are used to create a single, meaningful dataset partition that is relevant to topic T. In addition to synthesizing the chosen documents into a dataset, the generative process enhances the dataset by comprehending context and integrating additional content derived from the same source knowledge data.

The generative process depends on a prompt engineering method that employs topic-conditioned prompts. Each question is formulated by combining the chosen subject T with the top-N most pertinent retrieved document parts. This method directs the language model to produce Q&A pairs that are specific, contextually relevant, and consistent with the specified topic. This strategy, albeit lacking extensive parametric control, successfully confines the generated material to the targeted topic by anchoring the prompt in highly relevant textual evidence.

The assessment of the resultant dataset—specifically its congruence with topic T, the uniformity of replies, and the variety of included viewpoints—is elaborated upon in Section 4, since it constitutes an integral component of the experimental validation of our technique.

The network engineering topics were chosen for their importance in industry certification curricula (e.g., JNCA, JNCP, CCNA, CCNP), training materials, and technical documentation, guaranteeing comprehensive coverage of fundamental and advanced networking ideas. This selection method guaranteed that the dataset would thoroughly encompass both basic and advanced networking topics relevant to real-world professional scenarios. The end result dataset covers a diverse array of topics, including but not limited to Networking Fundamentals, OSI Model, TCP/IP Model, Routing and Switching Principles, Static versus Dynamic Routing, Routing Protocols (e.g., RIP, OSPF, EIGRP, BGP,

IS-IS), MPLS, VRF, PBR, Route Summarization, Load Balancing, Failover Mechanisms, Anycast and Multicast Routing, IPv6 Routing, Router Configuration, among others.

3.3. Q&A Dataset Generation with Multiple LLMs

In our experiments, we produced generated datasets for questions and answers with the following models: Qwen 2.5, Phi-4, LLaMA 3.3, LLaMA 3.1, Mixtral, and DeepSeek-R1, each including 1280 entries, with minor discrepancies in LLaMA 3.3 (1277 entries) and LLaMA 3.1 (1273 entries).

Each open-source LLM model produced a dataset that has 130 distinct topics,

guaranteeing comprehensive coverage of networking and security principles within a balanced distribution. Although the subject distribution is predominantly consistent, LLaMA 3.3 has minor discrepancies, with specific topics occurring 9 times, whilst LLaMA 3.1 features an anomaly where IP Addressing appears just 3 times. Notwithstanding these slight discrepancies, the datasets collectively offer a systematic and thorough depiction of essential networking and security concepts.

In scenarios where reliable data sources are not widely available, the ideas of variety and specificity in datasets are essential for improving the resilience and flexibility of models. For models to be able to adjust to new and unexplored data efficiently, dataset variety is a crucial component. Diverse datasets enable models to work across various activities and domains by lowering biases and enhancing flexibility. This diversity results from using multiple models to generate our data.

Lightly supervised data augmentation is one of the most effective strategies for promoting dataset variety. Exposing models to various instances, these datasets help models establish more robust feature representations and enhance their generalization capabilities.

Another way to ensure diversity is the utilization of multi-source datasets. Models can detect and take advantage of cross-domain correlations when different data types—like textual and structured data—or languages are integrated. Understanding subtle linguistic and structural signals is crucial for success in complicated tasks like named item identification and connection extraction, where these correlations are particularly helpful.

Conversely, dataset specificity offers a clear strategic advantage by allowing models to handle specialized jobs precisely. For instance, we could generate datasets for specialized applications using domain-specific and expert knowledge. This has proven useful in specialized settings as these datasets enable models to perform better than more generic solutions [6].

Additionally, specific datasets encourage innovation by addressing gaps usually missed by ordinary datasets. Private or otherwise difficult-to-acquire data may result in significant performance improvements when publicly accessible datasets are insufficient.

The synergy between specificity and variety is most evident in contexts with limited resources. The lack of labeled data in these situations calls for creative methods of dataset generation. Researchers may achieve a crucial balance between task-specific specialization and broad generalization by combining a variety of data augmentation approaches with domain-specific unique datasets. In addition to being adaptable to various applications, this balance guarantees that models perform very well in the specific settings for which they are created [6].

Improvements in model generalization and performance largely depend on the twin concepts of dataset variety and uniqueness. We may significantly improve the models' effectiveness and versatility by combining various linguistic and contextual components while creating task-specific datasets. In addition to addressing the difficulties of low-resource situations, these strategies help to further improve the data-driven approaches in NLP and other fields. High-quality datasets constitute the basis for practical model training, particularly in network engineering. Precise and flawless data enables models to provide valid setups and adjust to real-world requirements. Unreliable or overly generic information in a dataset might result in erroneous outcomes and constrain the model's capacity to perform specialized functions, such as routing and security.

Quality also encompasses the relevance of the data to the domain. A dataset comprised of irrelevant information will not facilitate a model's comprehension of network engineering intricacies. A model trained on software development data or unrelated tasks will have difficulty configuring firewalls or optimizing traffic flow. Datasets concentrating on critical domains—such as routing protocols, VLAN configurations, or compliance mandates—enhance the model's comprehension of network engineering.

As networks evolve and expand, high-quality datasets guarantee that models can adapt to emerging protocols, devices, and security threats. Training the model on substandard data may result in network configurations that are ineffective or susceptible to vulnerabilities. Upholding rigorous data standards not only mitigates these issues but also provides the model with the necessary adaptability in a dynamic field.

Coverage shows the extent and variety of the dataset, guaranteeing that the model encounters numerous types of devices and configurations. A comprehensive dataset in network engineering encompasses instances of network components, including switches, routers, and ACLs, as well as diverse protocols such as BGP, OSPF, and VLAN settings. This variability enables the model to manage diverse devices, traffic patterns, and operational requirements more efficiently.

To ensure complete coverage, the dataset must encompass many network situations, including fault tolerance designs, traffic load management tactics, and intrusion detection systems. Incorporating these real-world examples enhances the model's preparedness to address the varied difficulties in network management. This degree of coverage ultimately results in more precise, dependable, and versatile outputs that can accommodate intricate network environments.

Although the number is significant, it must not compromise quality or coverage. Extensive datasets are genuinely valuable only when they offer significant and varied instances that accurately represent the realities of network environments. Practically, possessing diverse data encompassing various device types, distinct protocols, and an array of configurations—from basic installations to intricate architectures—is significantly more advantageous than depending solely on volume. For example, incorporating thousands of nearly identical router configurations may increase the dataset's size. Yet, it minimally enhances the model's proficiency in handling specialized tasks such as managing complex routing topologies, implementing security protocols, or diagnosing rare failure conditions.

An effectively designed dataset harmonizes depth and breadth, encompassing routine procedures and exceptional cases requiring specialist expertise. This diversity enhances the model's comprehension of the network domain while mitigating the risk of overfitting to a limited range of configurations. When the data matches with real-world tasks, it fa-

cilitates more precise forecasts and suggestions, guaranteeing that essential network scenarios—such as high-traffic load management, fault tolerance, or intrusion detection—are not neglected. Consequently, the model acquires the necessary flexibility and reliability to function efficiently in the field instead of becoming ensnared by recurring instances that fail to test its capabilities.

3.4. Cleaning and Deduplication

Controlling dataset overlap and redundancy is a crucial part of large-scale data management. Data quality, storage efficiency, and analytical performance can all be severely harmed by duplicate entries and overlapping data. Deduplication algorithms have been created to overcome these obstacles and keep datasets optimal for contemporary uses.

Cutting down on duplication has wider ramifications than being a technical need. Redundant data makes data retrieval more difficult, raises expenses, and wastes storage. Analytical models and decision-making systems operate more quickly and accurately when dataset redundancy is reduced [6]. Furthermore, removing duplicate entries improves data integrity by avoiding inconsistencies resulting from keeping different copies of the same information. This is particularly important in fields where data reliability is crucial, such as healthcare and finance [6].

Controlling the semantic overlap of the dataset corpus becomes even more critical in low-resource contexts, where every data point has a significant influence on model training. The impact of duplicate or overlapping entries increases when data is scarce. According to [6], methods like weak supervision and targeted data augmentation highlight how crucial well-selected datasets are to maximize their usefulness. In addition to these techniques, deduplication is essential for ensuring models generalize well to new data.

Table 1. Cleaning Metrics

Model	**Before**	**After**	**Decrease (%)**	**Retained (%)**
Qwen 2.5	1280	1040	18.8	81.3
Phi-4	1280	596	53.4	46.6
LLaMA 3.3	1277	910	28.7	71.3
LLaMA 3.1	1273	496	61.0	39.0
Mixtral 8x7B	1280	274	78.6	21.4
Deepseek-R1	1280	55	95.7	4.3
Total	**7670**	**3371**	**56.0**	**44.0**

To refine LLMs and help network engineers, Wuang et al. propose multiple directions in which this improvement could be made: Network Design, Network Diagnosis, Network Configuration, and Network Security [8]. They also underline the issues for data security and privacy when using an LLM that is not open-source. By fine-tuning these models with

network-specific data, they can perform more relevant and secure operations, potentially transforming how network tasks are managed and executed.

An alternative approach that was researched involves converting the natural language high-level rules and requirements into low-level network Application Programming Interfaces (API) [23]. Thanks to this translation, network engineers can apply complex network settings more easily, eliminating the need to manually understand and apply complicated regulations. This technique translates verbose natural language instructions into exact network commands and settings using LLMs. This automates converting network needs into executable configurations, simplifying the process and reducing mistakes.

In the process of using LLMs to help network engineers configure or understand the network devices configuration, the main issue is validating the output [15] and the stated objective is to maximize the use of automated verifiers and minimize the use of human ones.

In a context where the need to fine-tune LLMs to more specific fields is growing, we aim to create a specific dataset that can help us improve LLMs for use in network engineering applications. This research aims to generate an extensive overview of the state of networking domains, utilizing various freely available sources.

The selection of books that together reflect the state-of-the-art in network engineering is the first step in the process. Another of our objectives is to create those datasets and minimize the need for RLHF. The Q&A dataset created from this corpus can be used as LLM fine-tuning material.

These expansions not only promise to enhance the current system's capabilities but also align with the ongoing advancements in AI and machine learning technologies, aiming to create more sophisticated, accurate, and user-friendly tools for data processing and analysis.

3.5. Fine-Tuning Approach

We began the training process by running three initial epochs for both LLMs to establish a baseline for their performance. To assess if this training period resulted in quantifiable gains in the model's performance, we raised the number of training epochs to 10. With this modification, we evaluated how more training iterations affected the models' capacity for efficient learning and generalization.

LoRA (Low-Rank Adaptation) is a method used to effectively fine-tune extensive machine learning models by modifying only a limited number of parameters, hence considerably diminishing computational resources and storage needs. This method is very helpful when dealing with resource-intensive models since it allows for effective task adaptability without requiring a lot of GPU memory or processing power. To maintain the scalability and affordability of the fine-tuning process, we sought to compromise model flexibility and computing efficiency by using LoRA . However, empirical studies have shown that in standard low-rank settings, LoRA often underperforms full fine-tuning regarding sample efficiency and target domain performance, particularly in challenging domains such as programming and mathematics. For instance, when it comes to instruction finetuning, full finetuning might get better results on tests like HumanEval and GSM8K, but LoRA keeps the base model's abilities better on tasks that aren't in the target domain. Moreover, a detailed singular value decomposition analysis reveals that full fine-tuning tends to learn perturbations with ranks 10 to 100 times higher than those typically used in LoRA

configurations. This suggests that the whole fine-tuning process captures a richer set of adaptations. Despite its limitations in achieving peak domain-specific performance, the reduced forgetting and enhanced efficiency offered by LoRA make it an attractive alternative, particularly in scenarios where GPU memory is a limiting factor. These insights provide a promising direction for future research to balance the trade-off between adaptation and generalization in LLMs [3].

4. Experiments and Results

4.1. Experimental setup

We evaluated our methodology with a dataset of 3,097 question-answer pairs of network engineering subjects, divided into a 9:1 training-to-test ratio for each topic. This guarantees diversity in both the training and testing datasets.

The experiments were conducted in three phases:

Phase 1: Embedding Generation from Source documents The first step involved processing and converting textual material from our selected sources into embeddings. These embeddings were stored in collections in a Chroma[4] database. The text embeddings were generated using LangChain's GPT4AllEmbeddings Python package [5]. With an average of roughly 292.5 embeddings per source, the resulting collection has 23,400 embeddings with 384 dimensions. This organized representation makes effective semantic search and retrieval possible in later stages.

Phase 2: Q&A Pair Generation The Q&A pairs were generated using DeepSeek-R1 671B, LLaMA 3.1 70B, Qwen 2.5 72B, and Mixtral 8x7B. Usually, running these model sizes requires dedicated, expensive infrastructure, so we rely on cloud-based LLM providers, such as OpenRouter [6], which provides a dedicated API to access the above-mentioned models.

Phase 3: Fine-tuning Fine-tuning was performed on LLaMA 3.2-1B, LLaMA 3.2-3B and Qwen 2.5-1.5B using LoRA, and we evaluated both training and computational performance key metrics summarized in Table 2. We run the fine-tuning process on cloud-based infrastructure with an RTX 6000 Ada GPU card with 48 GB VRAM, 64GB RAM and 16vCPU AMD EPYC 9354.

We conducted experiments comparing training for 3 versus 10 epochs to evaluate the impact of extended training, assessing the predictive and generative capabilities of the model using the metrics summarized in Table 3.

4.2. Results

In our experiments, we have used the following models for dataset generation: Mixtral, Llama, Phi-4, Qwen2.5, and DeepSeek-R1. After applying cleaning and deduplication steps, we combined the resulting data from all the LLMs and used it for fine-tuning.

[4] https://www.trychroma.com/
[5] https://python.langchain.com/
[6] https://openrouter.ai/

Table 2. Summary of Training and Computational Performance Metrics

Category	Metric	Description
Training Metrics	Gradient Norm	Measures the norm of gradients to monitor training stability.
	Loss	Represents the overall training loss function value.
	Training Loss	Captures the evolution of loss throughout the training process.
Computational Performance Metrics	Total FLOPs	Estimates the number of floating-point operations during training.
	Train Samples per Second	Indicates training throughput based on processed samples.
	Train Steps per Second	Measures the frequency of completed training steps.

Table 3. Summary of model's predictive and generative Evaluation Metrics

Metric	Description
Perplexity	Measures how well a model predicts the next word in a sequence based on predicting context. Lower perplexity indicates better language model performance.
BLEU	Calculates the similarity between generated and reference text based on overlapping n-grams, commonly used in machine translation evaluation.
METEOR	Considers synonyms, stemming, and word order while comparing generated text with reference text, improving upon BLEU.
ROUGE-L	Measures the longest common subsequence between generated and reference text, often used in summarization evaluation.
BERTScore F1	Evaluates semantic similarity using contextual embeddings from BERT, capturing meaning rather than exact word matches.
Average Inference Time	Records the average time taken to generate a response, which is crucial for evaluating model efficiency.
Sentence Similarity	Measures how semantically close the generated text is to the reference text using vector-based similarity methods.

We deployed a data-cleaning workflow to achieve accuracy and relevance. This involved clustering-based filtering, eliminating empty or excessively similar items, and deduplication, all designed to prevent recurrent queries from biasing the training process. Furthermore, we employed a configuration extractor that automatically recognized and validated the structural coherence of configurations incorporated into answers. This stage was crucial for preserving consistency and accuracy across datasets, which increased their dependability for jobs that came next.

With the preparation of the datasets, we see a reduction in size for each dataset, as depicted by Table 1: Qwen25 decreased by 18.8%, Phi-4 by 53.4%, LLaMA3.3 by 28.7%, LLaMA3.1 by 61%, Mixtral by 78.6%, and DeepSeek-R1 by 95.7%. It is noted that DeepSeek-R1 has the largest volume of useless data. Utilizing all the datasets produced by the models and following the cleaning process yielded 3,371 entries. The dataset comprises a total of 3,352,361 tokens (questions tokens: 361,865 and answers tokens: 2,990,496).

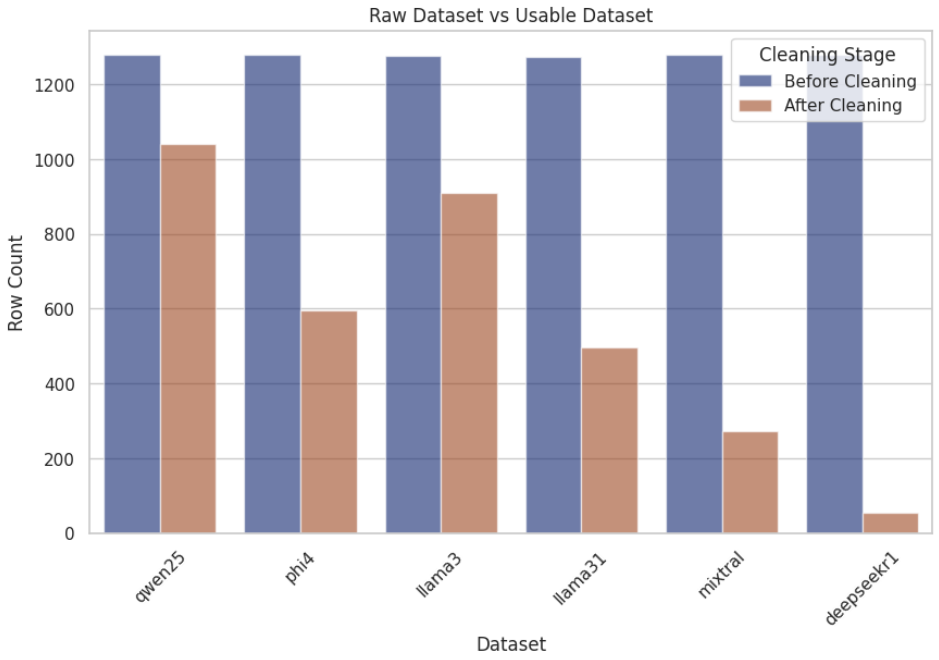

Fig. 2. Generated dataset vs usable dataset

Fine-tuning an LLM using LoRA has proven a more efficient way to adapt LLMs to new tasks and domains while significantly reducing the memory and computational overhead associated with full model fine-tuning. LoRA leverages low-rank adaptation by training only low-rank perturbations to selected weight matrices, thereby saving memory by freezing most of the pre-trained model's parameters [3].

This method not only cuts down on the number of trainable parameters but also keeps the model from going too far from its original state, which helps prevent catastrophic forgetting.

We used the resulting dataset to fine-tune models and improve the output for network-related tasks. We ran into a number of difficulties, especially while the model was being evaluated. The handling of the context window was one of the primary problems since there were times when the input sequence length was longer than the model's maximum permitted limit. We addressed this by implementing truncation, which made sure that any input that was longer than the maximum sequence length established by the model was trimmed appropriately.

Table 4. Performance Metrics for LLaMA 3.2 1B Before and After Fine-Tuning (3 Epochs)

	Test		Train	
Metric	**Before**	**After**	**Before**	**After**
Average Inference Time (s)	7.8891	10.1844	20.7003	11.0384
BERTScore F1	0.9389	0.9389	0.9398	0.9398
BLEU	0.6905	0.6438	0.5474	0.6448
METEOR	0.7687	0.8307	0.7549	0.7847
Perplexity	4.8191	4.1848	1.1811	3.3652
ROUGE-L	0.8124	0.8057	0.7931	0.7889
Sentence Similarity	0.9085	0.9085	0.9187	0.9187

Each model performance metric has been essentially impacted by fine-tuning, as shown in Table 4, with noteworthy variations in each model's quality and efficiency. For example, the LLaMA 3.2 1B model optimized for three epochs demonstrates a 13% decrease in perplexity (from 4.8191 to 4.1848) and an 8% rise in METEOR on the test dataset (from 0.7687 to 0.8307), suggesting enhanced fluency and confidence in language creation. Although the text quality increases in some ways, the production speed is hampered, as seen by the minor 7% decrease in BLEU and the 29% increase in inference time (from 7.8891 s to 10.1844 s) as specified in Table 5. Even while the perplexity rises—possibly as a result of a change in distribution brought on by regularization effects during fine-tuning—BLEU and METEOR improve by around 18% and 4%, respectively, on the training dataset for the same model.

The test results significantly improve after fine-tuning to 10 epochs for the LLaMA 3.2 1B model showing an overall improvement in fluency and resemblance to human reference texts as ROUGE-L shows a 3% boost, perplexity decreases by almost 15% (from 4.8191 to 4.0984), and METEOR rises by about 10% (from 0.7687 to 0.8465) despite a minor 6% decrease in BLEU and a 37% increase in inference time (from 8.0152 s to

Table 5. Performance Metrics for LLaMA 3.2 1B Before and After Fine-Tuning (10 Epochs)

	Test		Train	
Metric	**Before**	**After**	**Before**	**After**
Average Inference Time (s)	8.0152	10.9568	20.9625	6.7583
BERTScore F1	0.9389	0.9390	0.9398	0.9398
BLEU	0.6905	0.6492	0.5474	0.7186
METEOR	0.7687	0.8465	0.7549	0.8449
Perplexity	4.8191	4.0984	1.1811	1.8397
ROUGE-L	0.8124	0.8374	0.7931	0.8400
Sentence Similarity	0.9085	0.9085	0.9187	0.9187

Table 6. Performance Metrics for LLaMA 3.2 3B Before and After Fine-Tuning (3 Epochs)

	Test		Train	
Metric	**Before**	**After**	**Before**	**After**
Average Inference Time (s)	373.4122	23.2606	15.1097	35.6101
BERTScore F1	0.9389	0.9389	0.9398	0.9398
BLEU	0.2984	0.5834	0.7468	0.5064
METEOR	0.8008	0.8398	0.7998	0.8066
Perplexity	4.1884	2.0123	3.6710	2.3354
ROUGE-L	0.8539	0.7860	0.8529	0.7560
Sentence Similarity	0.9085	0.9085	0.9187	0.9187

10.9568 s). More time spent fine-tuning the model makes it better at picking up on small linguistic patterns, even if it means that the training distribution is not as closely matched to the test distribution. On the training side, BLEU improves by 31% and METEOR by 12%. Our findings are presented in Table 6. The LLaMA 3.2 3B models show even more significant gains. The test dataset demonstrates a nearly 94% decrease in inference time (from 373.41 s to 23.26 s), a nearly 96% increase in BLEU (from 0.2984 to 0.5834), and a 52% decrease in perplexity (from 4.1884 to 2.0123) for the 3-epoch version. These improvements show that fine-tuning significantly boosts efficiency while improving the output text's quality. Even though the training dataset shows a 32% decrease in BLEU, the model's higher test performance suggests that it improves after fine-tuning. The LLaMA

Table 7. Performance Metrics for LLaMA 3.2 3B Before and After Fine-Tuning (10 Epochs)

	Test		Train	
Metric	**Before**	**After**	**Before**	**After**
Time (s)	373.4629	15.0883	15.5681	14.1322
BERTScore F1	0.9389	0.9389	0.9398	0.9398
BLEU	0.2984	0.6986	0.7468	0.6919
METEOR	0.8008	0.8667	0.7998	0.8622
Perplexity	4.1884	2.6968	3.6710	1.6675
ROUGE-L	0.8539	0.8303	0.8529	0.8156
Sentence Similarity	0.9085	0.9085	0.9187	0.9187

Table 8. Performance Metrics for Qwen 2.5 1.5B Before and After Fine-Tuning (3 Epochs)

	Test		Train	
Metric	**Before**	**After**	**Before**	**After**
Average Inference Time (s)	0.9512	1.4414	0.8115	0.6249
BERTScore F1	0.9664	0.9664	0.9675	0.9675
BLEU	0.9502	0.9504	0.9468	0.9466
METEOR	0.9640	0.9614	0.9651	0.9668
Perplexity	4.8994	3.5097	3.5957	2.7674
ROUGE-L	0.9850	0.9816	0.9848	0.9859
Sentence Similarity	0.9469	0.9469	0.9500	0.9500

3.2 3B model achieves even more significant gains with 10 epochs: perplexity decreases by 36% (from 4.1884 to 2.6968), BLEU rises by about 135% (from 0.2984 to 0.6986), METEOR improves by 8%, and inference time decreases by almost 96% (from 373.46 s to 15.09 s). Significant improvements are also shown in the training dataset, including a 55% increase in perplexity. These findings clearly show that thorough fine-tuning of the 3B model results in a much more effective model that generates text that is more similar to high-quality human references.

On the other hand, the Qwen 2.5 1.5B models already have scores that are close to ideal. The test dataset's perplexity decreased by 28% (from 4.8994 to 3.5097) in the 3-epoch version, although BLEU, BERTScore, METEOR, ROUGE-L, and Sentence Sim-

Table 9. Performance Metrics for Qwen 2.5 1.5B Before and After Fine-Tuning (10 Epochs)

	Test		Train	
Metric	**Before**	**After**	**Before**	**After**
Average Inference Time (s)	0.9569	1.1790	0.8272	1.1962
BERTScore F1	0.9664	0.9664	0.9675	0.9675
BLEU	0.9502	0.9504	0.9468	0.9470
METEOR	0.9640	0.9620	0.9651	0.9589
Perplexity	4.8994	3.5218	3.5957	2.2646
ROUGE-L	0.9850	0.9835	0.9848	0.9816
Sentence Similarity	0.9469	0.9469	0.9500	0.9500

ilarity remained high, ranging from 0.95 to 0.97. Nevertheless, the inference time rises from 0.9512 s to 1.4414 s, a 52% increase. The values for BLEU, BERTScore, METEOR, and ROUGE-L stay constant, but inference time even drops by 23% on the training set as shown in Table 8. The 10-epoch version of the Qwen model 9 shows a slight rise in inference time (about 23%) and a decrease in perplexity on the test dataset of around 28%, but other quality metrics essentially stay the same. This illustrates that fine-tuning improves the Qwen model's performance, mainly by reducing perplexity, which implies a smoother, more coherent generation.

Out of all the assessed models, the LLaMA 3.2 3B model that has been adjusted for 10 epochs performs the best overall. Its test dataset shows a remarkable 135% gain in BLEU, 36% decrease in perplexity, and 96% reduction in inference time. These enhancements show that the model is more effective in practice, which is crucial for real-world applications. In conclusion, even if each model may be improved, the LLaMA 3.2 3B 10-epoch model offers the most appealing alternative for deployment when speed and text quality are crucial due to its significant efficiency improvements.

5. Conclusions

In this paper, we developed and evaluated a novel approach for automatically generating domain-specific datasets to fine-tune open-source Large Language Models and we described our approach in the context of network engineering applications.

We introduced an effective pipeline for dataset generation that leverages existing technical documentation and literature in the network engineering field that contributes to building knowledge databases to create high-quality question-answer pairs spanning diverse network engineering topics. The automated generation process produced initial datasets from multiple LLMs (LLaMA, Qwen, Mixtral, Phi-4, and DeepSeek), which were refined through rigorous cleaning and deduplication. This process resulted in a final

dataset of 3,371 entries containing over 3.3 million tokens, covering networking concepts like routing protocols, security configurations, and network services.

We conducted a comparative analysis of data quality across different source models that revealed significant variations in the usability of generated content. We indicate that models like Qwen 2.5 maintained high retention rates (81.3%) after cleaning. In contrast, others, such as DeepSeek-R1, showed extremely low retention (4.3%), underscoring the importance of model selection in the dataset generation phase.

Our experiments with fine-tuning smaller LLMs using LoRA showed significant performance improvements. The LLaMA 3.2 3B model fine-tuned for 10 epochs showed the most dramatic improvements. This increased the BLEU score by 135%, decreased perplexity by 36%, and reduced the inference time by 96%. Based on these observations, we can say that the results confirm that even relatively small models can achieve significant domain-specific performance gains through targeted fine-tuning with high-quality datasets.

The methodology depicted in this paper offers a practical approach for adapting open-source LLMs to the network engineering domain without requiring extensive computational resources or human-annotated datasets. This approach is particularly valuable where domain expertise is critical, but labeled training data may be insufficient.

Future research directions include extending the proposed methodology to other domains, such as cloud-based systems, to develop a more complex validation procedure, including human expert evaluation, to guarantee the correctness of the generated content and explore data augmentation techniques for improving model performances.

References

1. Abdin, M., Aneja, J., Behl, H., Bubeck, S., Eldan, R., Gunasekar, S., Harrison, M., Hewett, R.J., Javaheripi, M., Kauffmann, P., Lee, J.R., Lee, Y.T., Li, Y., Liu, W., Mendes, C.C.T., Nguyen, A., Price, E., de Rosa, G., Saarikivi, O., Salim, A., Shah, S., Wang, X., Ward, R., Wu, Y., Yu, D., Zhang, C., Zhang, Y.: Phi-4 technical report (2024), https://arxiv.org/abs/2412.08905
2. Ben Houidi, Z., Rossi, D.: Neural language models for network configuration: Opportunities and reality check. Computer Communications 193, 118–125 (Sep 2022), http://dx.doi.org/10.1016/j.comcom.2022.06.035
3. Biderman, D., Portes, J., Ortiz, J.J.G., Paul, M., Greengard, P., Jennings, C., King, D., Havens, S., Chiley, V., Frankle, J., Blakeney, C., Cunningham, J.P.: Lora learns less and forgets less (2024), https://arxiv.org/abs/2405.09673
4. Christiano, P.F., Leike, J., Brown, T.B., Martic, M., Legg, S., Amodei, D.: Deep reinforcement learning from human preferences. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 4302–4310. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
5. DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z.F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J.L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R.J., Jin, R.L., Chen,

R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S.S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W.L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X.Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y.K., Wang, Y.Q., Wei, Y.X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y.X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z.Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., Zhang, Z.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning (2025), https://arxiv.org/abs/2501.12948
6. Deng, S., Ma, Y., Zhang, N., Cao, Y., Hooi, B.: Information extraction in low-resource scenarios: Survey and perspective (2024), https://arxiv.org/abs/2202.08063
7. Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J.W., Wallach, H., III, H.D., Crawford, K.: Datasheets for datasets (2021), https://arxiv.org/abs/1803.09010
8. Huang, Y., Du, H., Zhang, X., Niyato, D., Kang, J., Xiong, Z., Wang, S., Huang, T.: Large language models for networking: Applications, enabling techniques, and challenges (2023), https://arxiv.org/abs/2311.17474
9. Jiang, A.Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D.S., de las Casas, D., Hanna, E.B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L.R., Saulnier, L., Lachaux, M.A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T.L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., Sayed, W.E.: Mixtral of experts (2024), https://arxiv.org/abs/2401.04088
10. Jiang, X., Gember-Jacobson, A., Feamster, N.: Caip: Detecting router misconfigurations with context-aware iterative prompting of llms (2024), https://arxiv.org/abs/2411.14283
11. Jing, Z., Su, Y., Han, Y.: When large language models meet vector databases: A survey (2024), https://arxiv.org/abs/2402.01763
12. Juniper: Day one books (2023), https://www.juniper.net/documentation/jnbooks/us/en/day-one-books, accessed on 20 november 2023
13. Maatouk, A., Ampudia, K.C., Ying, R., Tassiulas, L.: Tele-llms: A series of specialized large language models for telecommunications (2024), https://arxiv.org/abs/2409.05314
14. Mani, S.K., Zhou, Y., Hsieh, K., Segarra, S., Chandra, R., Kandula, S.: Enhancing network management using code generated by large language models (2023), https://arxiv.org/abs/2308.06261
15. Mondal, R., Tang, A., Beckett, R., Millstein, T., Varghese, G.: What do llms need to synthesize correct router configurations? (2023), https://arxiv.org/abs/2307.04945
16. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R.: Training language models to follow instructions with human feedback (2022), https://arxiv.org/abs/2203.02155
17. Parmar, J., Prabhumoye, S., Jennings, J., Liu, B., Jhunjhunwala, A., Wang, Z., Patwary, M., Shoeybi, M., Catanzaro, B.: Data, data everywhere: A guide for pretraining dataset construction (2024), https://arxiv.org/abs/2407.06380
18. Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., Qiu, Z.: Qwen2.5 technical report (2025), https://arxiv.org/abs/2412.15115
19. Su, H., Shi, W., Kasai, J., Wang, Y., Hu, Y., Ostendorf, M., tau Yih, W., Smith, N.A., Zettlemoyer, L., Yu, T.: One embedder, any task: Instruction-finetuned text embeddings (2023), https://arxiv.org/abs/2212.09741

20. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models. CoRR abs/2302.13971 (2023), https://doi.org/10.48550/arXiv.2302.13971
21. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models (2023), https://arxiv.org/abs/2302.13971
22. Trăistaru, C., Pop, F., Bădică, C., Ciochiu, D., Bădoi, M., Nedianu, G.C.: Leveraging open source large language models to generate datasets from existing field-specific texts (2024)
23. Wang, C., Scazzariello, M., Farshin, A., Kostic, D., Chiesa, M.: Making network configuration human friendly (2023), https://arxiv.org/abs/2309.06342
24. Wu, D., Wang, X., Qiao, Y., Wang, Z., Jiang, J., Cui, S., Wang, F.: Netllm: Adapting large language models for networking. In: Proceedings of the ACM SIGCOMM 2024 Conference. p. 661–678. ACM SIGCOMM '24, ACM (Aug 2024), http://dx.doi.org/10.1145/3651890.3672268
25. Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X.V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P.S., Sridhar, A., Wang, T., Zettlemoyer, L.: Opt: Open pre-trained transformer language models (2022), https://arxiv.org/abs/2205.01068

Claudiu Trăistaru is an assistant lecturer at the Department of Computer and Information Technology, within the Faculty of Automation, Computers and Electronics at the University of Craiova, Romania. His research interests include cloud computing technologies, computer networks, cybersecurity, and applications of artificial intelligence. He has been working as an IT professional for more than 20 years and has extensive experience in systems engineering, DevOps, and infrastructure management.

Florin Pop is full Professor at the Computer Science and Engineering Department of National University of Science and Technology POLITEHNICA Bucharest, Romania. His main research interests are in large-scale distributed systems, adaptive and autonomous methods, multi-criteria optimization methods, Cloud middle-ware tools and applications development, prediction methods, self-organizing systems, applications for Big Data and IoT systems, Smart Cities, Climate Neutral Governance. Florin Pop is an editor of Future Generation Computer Systems. He is senior researcher at National Institute for Research Development in Informatics - ICI Bucharest and member of Academy of Romanian Scientists. He is IEEE Senior Member.

Costin Bădică b. February 19, 1966, in Craiova, Dolj. He is a Professor Doctor Engineer at the University of Craiova, Faculty of Automation, Computers, and Electronics. Since 2012, he has been the Director of the "Constantin Belea" Doctoral School within the Faculty of Automation, Computers, and Electronics at the University of Craiova. As of 2024, he is an Associate Member of the Academy of Technical Sciences of Romania (ASTR), Section: Information and Communications Technology, Computers, and Telecommunications. He serves as Co-Chair of the Technical Committee on Computational Collective Intelligence for the IEEE SMC Society. This committee received the "Most Active SMC Technical Committee Award" in Cybernetics from the IEEE SMC Society in both 2021 and 2024. He is a member of the editorial boards for the following journals: Computer

Science and Information Systems (ComSIS), ISSN 1820-0214; System Theory, Control and Computing Journal, ISSN: 2810–4099; Cybernetics and Information Technologies (CIT), The Journal of the Institute of Information and Communication Technologies of the Bulgarian Academy of Sciences, eISSN 1314-4081; Simulation Modelling Practice and Theory, ISSN 1569-190X (WoS); Vietnam Journal of Computer Science, ISSN (print): 2196-8888 — ISSN (online): 2196-8896. He is a founding member of the international conference IDC (Intelligent Distributed Systems) – 2007. His areas of interest include: artificial intelligence, multi-agent systems, distributed systems, and software engineering.

Catalina-Felicia Mancas is a lecturer and a researcher of the Computer and IT department at University of Craiova since 2010. With a solid teaching expertise in Computer Networks and a strong research interest in Network Engineering and Quantum Communication, Catalina combines academic expertise with practical knowledge to support innovation in computing and communication technologies.

Ionuț Murarețu is a lecturer in the Department of Computer and Information Technology, within the Faculty of Automation, Computers and Electronics at the University of Craiova, Romania. His research spans several areas of computer science and engineering, with a focus on artificial intelligence, software engineering, embedded systems, the Internet of Things (IoT), multi-agent systems, and computational optimization. He is actively involved in teaching, research, and collaborative projects at the intersection of AI and embedded technologies.

Received: April 16, 2025; Accepted: July 25, 2025.