Automation of Network Configuration Generation using Large Language Models

Supratim Chakraborty
Software Engineer
Mobility Group
Cisco Systems Inc.
Bengaluru, India
supratch@cisco.com

Nithin Chitta
Principal Engineer
Mobility Group
Cisco Systems Inc.
Bengaluru, India
nithin.chitta@gmail.com

Rajesh Sundaresan
Professor, ECE
Associate Faculty, RBCCPS
Indian Institute of Science
Bengaluru, India
rajeshs@iisc.ac.in

***Abstract*—The life cycle for a service provider (SP) to launch a new service or tariff plan often requires months of planning and testing. The SP has traditionally used OSS (Operation support systems) and BSS (Business support systems) which are large, non-standard systems providing life cycle management related to launching new digital services (e.g. internet, voice, SMS, IoT) and tariff plans. These OSS/BSS systems, being multi-vendor and custom implementations, involve significant costs and a timeline of months to a year to roll out the service. This problem worsens with evolving technologies such as 5G. This paper addresses the specific need for faster provisioning of 5G network services and their corresponding tariff/billing plans, thus shortening the duration for launch. To provision the network, a combination of a deep neural network and a large language model (LLM) is proposed in this work to automate the generation of network configurations.**

***Index Terms*—Artificial Intelligence, Network Automation and Management, Large Language Models, Generative AI**

I. Introduction

The mobile packet core network is a complex network comprising multiple network functions that communicate with each other to provide end-to-end service. Once a network is commissioned by the network operator, default network settings like peer node IP addresses, build information, and various other system level configurations are provisioned which constitute the base-level configuration of the network. A typical digital service, for example, connection to the internet, requires setting up network service configurations and policy configurations like IP address pool, DNS server configurations, PLMN identifier, QoS, and others [1], are administered on top of the already complex base-level infrastructure and network function configurations (together called Network Services and Policies or NSPs), which can constitute 10,000 to 30,000 lines of code [2], spread over different network functions. This process is often manual, time-consuming, and error-prone. Our goal in this paper is to leverage large language models or LLMs to automate this process.

Once these complex network services and policy configurations are set up, the tariff plan configurations are applied. Tariff plans are specific offers to customers on new digital services and billing options, such as prepaid or postpaid internet 5G data plans or rates (for e.g. 100 GB/month), voice calls, roaming and other value-added services. The service provider first provisions for the tariff plans in the core network, then advertises the tariff plans in the market, allows customers to choose from among the tariff plans, and associates customer choices with the plans in the network.

The tariff plan and the NSPs collectively constitute the network configuration. Once configured for a customer, that customer can access the network and enjoy the entitled services. Note that service providers often offer multiple tariff plans, each requiring a possibly distinct configuration for provisioning.

The following are challenges faced by network operators:

- Intricate and extensive configurations, often referred to as "spaghetti configurations", which can encompass up to 30K lines of code, introduce significant uncertainty in outcomes. This uncertainty comes from the multiple interacting layers within the configuration, resulting in situations where outcomes, such as the customer's end-of-month billed amount, remains indeterminate.
- Assessing the suitability of layers of interacting configuration in meeting the quality of service needs during the launch of new tariff plans and digital services.
- Dynamic nature of tariff plans due to incorporation of new customer base, hosting of mega events, various emergencies or exigencies in the area of operation.
- High susceptibility to human error in manual configurations and outdated, lengthy manuals complicate understanding of inter-dependencies and optimisations.

These factors and challenges result in high total cost of ownership, significant manual effort, and extensive time taken to launch new digital services on the packet core network (5G Network) and their associated tariff plans. The main objective of this work is to build an automation engine using large language models to address these challenges, in particular,

- Build an AI engine that translates a tariff plan into network configuration.
- Include the AI engine in a pipeline that validates the generated network configuration minimises or eliminates the need for human intervention during network provisioning, and reduces the time to launch new tariff plans.

978-3-903176-66-9

II. Related Work

Extraction of information from tariff plans is similar to the workings of intent-based systems, a well-explored area in intent-based networking [3], [4]. Use of Natural Language Processing (NLP) techniques and LLMs to translate intents into tasks is becoming a popular choice in autonomous networking, such as a recent work [5] have demonstrated intent-to-policy translation, then mapping policies to an API with key-value pairs using LLMs. The proposed method in this work employed few-shot prompting with OpenAI APIs, evaluated on execution time and the number of translated policies. Another study by Ahlam et al., [6] showed the translation of high-level intents to specific network configurations using LLMs with few-shot prompting. However, these methods are prone to model hallucinations. In our work, we experimented with different approaches, including NLP and LLM fine-tuning [7], which has shown to be an effective remedy for hallucinations of LLM [8] compared to few-shot prompting. We evaluated our models against both valid and invalid inputs, proposing an input filtration technique to prevent possible misconfigurations in the network.

III. Proposed System Design for Tariff Plan Layer

Tariff plan variability: A tariff plan outlines the services offered by a mobile network operator, including data allowances, voice call minutes, SMS bundles, and pricing structures. As part of this study, various tariff plans were studied in 27+ customers, including across geographies. The study found a high variability in these plans for the following reasons: a) *Market demands*: Providers offer various plans for different segments (consumer, enterprise, and IoT), leading to variability. b) *Audience segmentation*: Targeted plans like Gold (premium services) and Silver (limited services) create variation. Student plans focus on data costs, while enterprise plans emphasise voice calls. c) *Regulatory variations*: Geographic regulations affect plans. Some applications are blocked in places like the Middle East and parts of India. US providers have app-specific QoS rules. d) *Technological innovation*: New technologies (e.g., 5G) and features (e.g., unlimited data) change tariffs. e) *Marketing strategies*: Providers differentiate with unique marketing. "Unlimited data" can vary greatly, e.g., 50 GB before throttling or 50 Mbps capped speed.

Need for standardised API: This high variability of tariff plans necessitates a standardised API to capture key features of a tariff plan. As part of this work, the following five keywords (entities) are identified, *'dnn', "data allowance", "avg speed data allowance", "data speed video stream" and "FUP redirection url"* to quantify key features in a tariff plan. Keywords improve the accuracy of data extraction from tariff plans by providing a deeper understanding of the plan's intent beyond the literal wording. They facilitate automation within the API by offering a standardised way to search for and interpret plan details. This combination of precision and automation leads to a more efficient and reliable process to translate diverse tariff plans into consistent network configurations.

API structure: We arrived at a suitable configuration API structure, described in Listing 1, closely following the standard REST-based API architecture defined by the TMForum [9], [10]. The primary purpose of the configuration API is to consume the extracted information from the tariff plan as a payload and to connect it to the API-based network provisioner (network orchestrator), as shown in Fig. 1. Subsequently, the network orchestrator maps the API into sections of configurations and provisions these configurations across the network functions. Fig. 1 shows blocks sequenced together as a pipeline. The sequence in its entirety is executed by an AI Processing Engine or simply AI-Engine, followed by a Network Orchestration Engine. AI Engine executes the following blocks -

i) *BLOCK-1*: Is the INPUT block. Input being a textual tariff plan. ii) *BLOCK-2*: Is the AI engine. AI engine takes BLOCK-1 as input, processes and extracts keywords from the same. The following information is captured from the tariff plan under the five keywords, namely 'dnn', "data allowance", "avg speed data allowance", "data speed video stream" and "FUP redirection url". iii) *BLOCK-3*: Is the OUTPUT block which is output by the AI engine. The output consists of key-value pairs that have been extracted from BLOCK-1 (INPUT). iv) *BLOCK-4*: The OUTPUT consisting of key-value pairs is mapped to parameters in a predefined standard API template, as shown in Listing 1. BLOCK-4 is where the API in the form of a REST-based-API is filled with values and handed over to the Network Orchestration Engine by a post-processing step in the AI Engine. v) *Network Orchestration Engine*: AI Engine invokes the REST-based API (populated with values which had been prepared in BLOCK-4) on the Network Orchestration

```
{
"bplan(string:The name of the configuration API
↪  template)": [
{
"name": "<string:Telecom Plan Name>",
"data": {
"name": "<string:Operator Name>",
"dnn": "<string:Data Network Name>",
"allowance": "<integer:Data Allowance>",
"allowance_unit": "<string:Allowance Unit (e.g.,
↪  GB)>",
"average_speed_limit_within_allowance":
↪  "<interger:Average Speed Limit Within
↪  Allowance>",
"average_speed_limit_unit_within_allowance":
↪  "<string:Average Speed Limit Unit (e.g.,
↪  MBps)>",
"final_unit_action": "<string:Final Unit Action
↪  (e.g., redirect)>",
"final_unit_action_url": "<string:Final Unit
↪  Action URL>",
"video_streaming": {
"name": "<string:Video Streaming Service Name>",
"speed": "<integer:Video Streaming Speed>",
"speed_unit": "<string:Video Streaming Speed
↪  Unit (e.g., MBps)>"}}}]}
```

Listing 1: Standardised Configuration API template. All keys can be identified as key features of a tariff plan.

Engine. Network Orchestration Engine takes the REST-based API and its parameter values to determine from its internal database the following: a) NF (Network functions) to apply the configuration to and b) standard template APIs to invoke on each network function. The network orchestration engine then invokes various network function APIs on the respective NFs. Hence, the REST-based API results in orchestration of API for Policy Control Function, API for Session Management Function, API for Charging Function, and others, with values mapped from the REST API (that was filled by AI Engine). In sections III-A to III-B we have explained in detail each functional block of the proposed system.

A. Extraction of Keywords

1) BLOCK-2: In Fig. 1 this block expands the AI engine to show how the input tariff plan text in BLOCK-1 is processed, vectorised and the relevant data elements are extracted. BLOCK-2A of Fig. 1 outlines the steps that take place to convert the tariff plan text to individual tokens after performing due text preprocessing such as removal of unnecessary special characters, the uniform lower case text conversion and data labelling (this is done specially in NER for entity linking). Tokenised output from BLOCK-2A with input tariff plan in BLOCK-1 is shown on the right of figure. After tokenisation, the tariff plan is ready to be passed through the vector embedding space in BLOCK-2B, where each of the tokens is converted into individual vectors. Vectorised output from BLOCK-2B is shown below the illustrated output of BLOCK-2A. Now, the vectorised tariff plan is passed through BLOCK-2C, which is the fine-tuned (trained LLM) or NER model layer and responsible for finding the keywords or entities in the tariff plan in the form of a dictionary (key-value pairs). Output from BLOCK-2C is shown in BLOCK-3.

Training method and performance: We have experimented the solution across three different approaches namely, a) NER trained model, b) Fine-Tuned GPT-3.5-Turbo-16K model and, c) Fine-Tuned Llama2-7B model. BLOCK-2C consists of trained models which are used to extract the desired keywords from the tariff plan. We used the spaCy[1] pipeline for the training of the NER model, GPT-3.5 fine-tuning APIs for GPT-3.5-Turbo-16K fine-tuning, and a base Llama2-7B model for fine-tuning Llama2.

a) Dataset: Given the sparse nature of tariff plan data, we observed 50 tariff plans from over 27 service providers and generated 1,000 samples for training. High-quality and diverse datasets are crucial for training NER models and fine-tuning LLMs, but acquiring such datasets in the network management domain is challenging due to the need for domain expertise, data collection, annotation, and preprocessing. To address this, LLMs like GPT-4 can generate synthetic datasets using prompt engineering and few-shot learning techniques tailored to the required scenarios. Before incorporating these synthetic tariff plans into the model for training and avoid possible data degradation, it is crucial to evaluate their quality. Using a scoring method, such as the one proposed in [11], the quality of the generated plans can be evaluated. Based on this score, conclusions can be made to either regenerate the data further post necessary changes or proceed with training the model using the generated samples.

b) Model Training: For Named Entity Recognition (NER) in tariff plans, we referred Entity Linking (EL) [12] which has proven to be an effective way to extract information from unstructured data. Each sample is labelled with five different aforementioned categories or keywords, along with their respective values and positions in the plan, forming the dataset. During training, the entire tariff plan is tokenised, the model then predicts probabilities for each token in the five categories. This distribution of the predicted probabilities is compared with the distribution of true labels (ground truth) for each category or keyword and the maximum predicted probability is used for this comparison. The difference in distribution between predicted probabilities and true labels is calculated as the loss, which is then used to update and improve the model performance over time. This process ensures that the model learns to recognise and categorise keywords accurately in tariff plans. Similarly, to fine-tune the generative models (GPT-3.5 and Llama2), we used instruction led [13] supervised fine-tuning on the training data. During training, each tariff plan is tokenised, vectorised, and input into the model along with the instruction prompt. The model then generates an output based on the prompt and the tariff plan. This output is compared to the ground truth, and cross-entropy loss is calculated for each token's position in the predicted output versus the ground truth. This loss is used to fine-tune the generative models. For training in the NER model with 3 epochs, the training loss was close to 6%. However, for GPT-3.5 and Llama2, the training loss was found to be close to 1% and 4% respectively (with the instruction prompting). Further details on model parameters are provided in Table III.

B. Configuration Generation and Validation

Once the entities are extracted as indicated by BLOCK-3 in Fig. 1, the predefined configuration API template in Listing 1 is filled by a post-processing step integrated into the AI Engine. The output configuration API from the input tariff plan in BLOCK-1 is shown in BLOCK-4. The network orchestrator consumes this API and performs the necessary sanity checks through various test cases to validate the configuration API. Once the sanity checks pass, the network orchestrator configures the network elements and sends feedback to the user, as outlined in the bottom left of Fig. 1.

Configuration Validation: The following sanity checks need to be performed before provisioning the configuration in a live production environment and they involve: a) Checking the configuration by a human operator before proceeding further with network provisioning. b) Identifying the functional and system level automation test suites that test the corresponding services or features being enabled by the API on the packet core and executing the test cases thus validating the success or failure of the process. The mandatory criteria for passing being 100% of smoke test cases, and high priority cases. The sanity

[1]https://spacy.io/

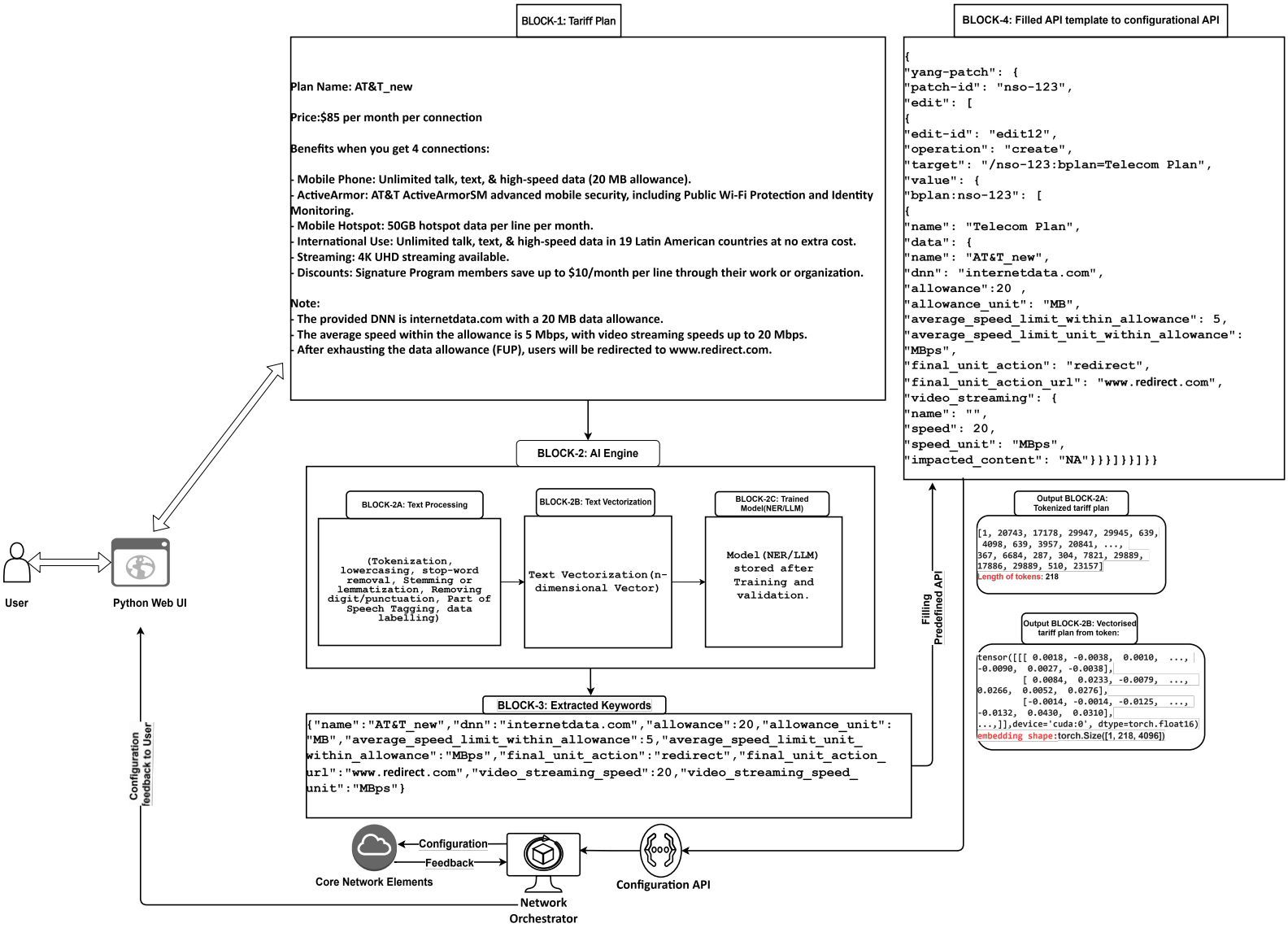

Fig. 1: Proposed system design to convert a tariff plan to desired configuration. BLOCK-1 to BLOCK-4 shows how a tariff plan is converted into configuration API. The output at different stages is also shown in the figure.

testing is performed on a replica staging environment and upon success the configurations are pushed by the Network Orchestration Engine into the production environment.

IV. Implementation and Results

As discussed in Section III-A1, three different types of models were implemented and evaluated via experiments. The models extracted information from the tariff plan using the NER technique [14] and using two fine-tuned generative models [15], namely, GPT-3.5 [16] and Llama2 [17]. The evaluation and training of the models are performed on Google Collab labs with Nvidia T4 GPUs. The GPT-3.5 fine-tuned model processed the tariff plan faster than the Llama2-7B model because GPT-3.5 was hosted on OpenAI's server and evaluated via API calls, while Llama2-7B required loading on the Collab GPU for fine-tuning.

Input Filtration: Erroneous or garbage inputs can generate misconfigurations of network elements and lead to malfunctioning and security vulnerabilities. For example, in the case of tariff plan illustrated in BLOCK-1 in Fig. 1, if the phrase "video_streaming_speed" is replaced with the meaningless input "APN name", the model extracts "video_streaming_speed" from the tariff plan as shown in the following section of invalid tariff plan.

Section of invalid tariff plan

```
Input: The average speed within the allowance is 5 Mbps, with
APN name up to 20 Mbps.
Output: {"video_streaming_speed": 20, "unit": "Mbps"}
```

We therefore need an input filter for all types of models (NER and LLMs) which will validate the input tariff plan as meaningful or otherwise.

TABLE I: Left-hand column of confusion matrices are for the system without the input filter for the tariff plan. The right-hand column matrices are with the input filter. Row (a) is for NER, (b) is for GPT-3.5-Turbo-Fine-tuned Model, and (c) is for Llama2-7B-Fine-tuned Model. The F1 scores are 1 for all situations except (b) without input filter for which it is 0.962.

		Predicted (0)	Predicted (1)
(a) Truth	(0)	25	0
	(1)	0	25
Truth	(0)	25	0
	(1)	0	25
(b) Truth	(0)	23	2
	(1)	0	25
Truth	(0)	25	0
	(1)	0	25
(c) Truth	(0)	25	0
	(1)	0	25
Truth	(0)	25	0
	(1)	0	25

TABLE II: Left-hand column of confusion matrices are for system without the input filter for PLMN. Right-hand column matrices are with the input filter. Row (a) is NER, (b) is GPT-3.5-Turbo-Fine-tuned Model, and (c) is Llama2-7B-Turbo-Fine-tuned Model. The F1 scores for left and right matrices are (a) 0.51 and 0.74, (b) 0.79 and 0.95, and (c) 0.69 and 0.95.

		Predicted (0)	Predicted (1)
(a) Truth	(0)	90	110
	(1)	95	105
Truth	(0)	200	0
	(1)	83	117
(b) Truth	(0)	115	85
	(1)	13	187
Truth	(0)	196	4
	(1)	15	185
(c) Truth	(0)	65	135
	(1)	22	178
Truth	(0)	198	2
	(1)	18	182

Key Criteria for Input Filter: i) Fine-tuned models are already trained to extract entities from a given tariff plan. Now, further adding complexity to classify a valid and invalid tariff plan will add to the cost of training, as training LLMs is a costly affair in terms of both money and time. Thus, there was a requirement to have a separate text filter before the fine-tuned model to filter out unwarranted inputs. ii) The input filter should be able to differentiate clearly between a valid and invalid tariff plan. Thus, and input filter is essentially a model for text classification. iii) We tested several methods for the input filter, such as shallow neural networks, naive Bayes classifier, and deep neural networks. Only the deep neural network confidently distinguished between valid and invalid tariff plans, yielding satisfactory results. The input filter is added before the model and when the output of the input filter is positive, only then the tariff plan is passed into the model for further processing. In our implementation, the input filter is a deep neural network with four layers: 1 embedding layer, 1 global average pooling layer, followed by two dense layers. The total number of trained parameters for this model is 783,233. A total of 5000 samples were used for training the input filter model with an equal number of valid and invalid tariff plans.

A. Validation of Implemented Models

To validate the NER model with and without input filters, we created a dataset of 50 tariff plans, evenly split between valid and invalid inputs. Each sample was augmented with keywords and their corresponding values. For most invalid samples, keyword values are set to 0, while a few have values from invalid tariff plans. The model is expected to correctly identify the nature of the plans considering the entire context: prediction 0 for invalid tariff plans and predict 1 for valid tariff plans. If the model predicts 1 (valid plan) when the ground truth is 0 (invalid plan), it has made an error. If it predicts 0 (invalid plan) when the ground truth is 1 (valid plan), it fails to extract at least one or more values correctly. Correct predictions are when the model predicts 0 (invalid plan) for ground truth 0 (invalid plan) and 1 (valid plan) for ground truth 1 (valid plan), showing that it can correctly not only identify invalid plans but can also extract values accurately from valid ones. Table I-Row(a) presents the performance of Named Entity Recognition (NER) model on 50 test tariff plans, comparing the results with and without input filtration. All invalid tariff plans are identified correctly and all valid tariff plans are processed correctly, both with and without the input filter. In the second implementation for fine-tuninig of GPT-3.5 model, GPT-3.5 *FineTuningJob* API [20] was used for model fine-tuning with 3 epochs (keeping in mind the cost associated with training). Once more, as delineated in Table I-Row(b), the integration of a deep neural network layer for the filtration of input text has mitigated the potential risk of misconfiguration of network elements from incorrect input. Llama2 fine-tuning was achieved on top of Llama2-7B base model with supervised fine-tuning [21], [22].

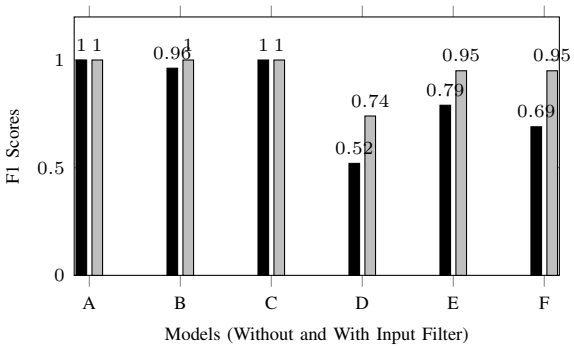

Fig. 2: F1 score comparison across models. The black columns represent F1 scores without the input filter, while the gray columns indicate scores with the input filter. Models from left to right: dataset: Tariff Plan- A) NER, B) GPT-3.5 Turbo (Fine-tuned), C) Llama2-7B (Fine-tuned); dataset: PLMN identifier- D) NER, E) GPT-3.5 Turbo (Fine-tuned), F) Llama2-7B (Fine-tuned).

Compute Challenges and Mitigation: While fine-tuning

TABLE III: Comparative F1 scores and model parameters for various approaches.

Model (F1 score)	**Model Parameters**
a) Tariff plan: GPT-3.5 fine-tuned (0.9615)	Fine-tuned with GPT-3.5-Turbo API. Learning rate multiplier=2, epoch=3, batch size=2, 3.68M tokens trained.
b) Tariff plan: GPT-3.5 fine-tuned with Filter (1)	Fine-tuning parameters are the same as Tariff plan: GPT-3.5 fine-tuned. *Input Filter:* Activation for all but the last dense layer = RELU, activation for the final layer = softmax, epochs = 145, trained parameters = 0.75M.
c) Tariff plan: Llama2-7B fine-tuned with/without Filter (1)	Fine-tuned on Llama2-7B model. LoRA attention dimension=64, α for LoRA scaling=16, Dropout probability for LoRA layers, Batch size per GPU for training=4, Batch size per GPU for evaluation=4, Optimizer=Adam, Tokenizer=Auto, QLoRA quantisation type=4-bit, Learning rate scheduler=constant. Trained parameters: 6.73B, epoch=1. *Input Filter parameters* same as Tariff plan: GPT-3.5 fine-tuned with Input Filter.
d) Tariff plan: Spacy based NER With/without Filter (1)	Learning rate=$1 \times \exp^{-5}$, β_1=0.9, β_2=0.999, optimizer=Adam [18], Embedding for tokenisation= MultiHashedEmbedding [19], batch size=200. *Input Filter parameters* same as Tariff plan: GPT-3.5 fine-tuned with Input Filter.
e) MCC-MNC: GPT-3.5 fine-tuned (0.7924)	Model parameters same as Tariff plan: GPT-3.5 fine-tuned. 1.21291M tokens trained.
f) MCC-MNC: GPT-3.5 fine-tuned with Filter (0.9512)	Model parameters same as Tariff plan: GPT-3.5 fine-tuned (Filter). 1.21291M tokens trained for GPT-3.5 fine-tuned model and 0.26M parameters trained for input filter.
g) MCC-MNC: Llama2-7B fine-tuned (0.694)	Model parameters same as Tariff plan: Llama2-7B fine-tuned with/without Filter.
h) MCC-MNC: Llama2-7B fine-tuned with Input Filter (0.948)	Model parameters same as MCC-MNC: Llama2-7B fine-tuned. *Input Filter parameters* same as Tariff plan: GPT-3.5 fine-tuned with Input Filter.
i) MCC-MNC: NER Model (0.506)	Model parameters same as Tariff plan: Spacy based NER With/Without Filter.
j) MCC-MNC: NER Model with Input Filter (0.738)	Model parameters same as MCC-MNC: NER Model. *Input Filter parameters* same as Tariff plan: GPT-3.5 fine-tuned with Input Filter.

the Llama2-7B model, we faced GPU constraints using the standard Google Colab setup with Nvidia T4 GPUs and 20 GB RAM, where the base model alone required 12 GB, leaving insufficient memory for training and evaluation. To mitigate these challenges, we limited training to a single epoch and employed Low-Rank Adaptation (LoRA) and Quantised LoRA (QLoRA) [23], [24] to optimise GPU utilization by reducing the number of trainable parameters. LoRA uses low-rank matrices for efficient weight updates, while QLoRA applies 4-bit quantisation to further decrease memory usage without compromising performance. Specific hyperparameters with respect to Llama2-7B fine-tuning included a LoRA attention dimension of 64, a scaling α of 16, a dropout of 0.1, and a batch size of 4 per GPU for both training and evaluation, with the Adam optimizer and a constant learning rate scheduler. Table I-Row(c) shows the evaluation results of a fine-tuned Llama2 model on the tariff plan where all invalid tariff plans are identified correctly and all valid tariff plans are processed correctly, both with and without the input filter.

V. Next Layer: MCC and MNC

The next layer of configuration that sits below the tariff plan layer contains parameters such as the public land mobile identifier (PLMN ID) that identifies the entire mobile network and contains MCC (Mobile Country Code) and MNC (Mobile Network Code). These parameters are not specified in tariff plans (example being the real life tariff plan depicted in BLOCK-1 of Fig. 1). This parameter appears in the network services and policies layer, and is captured via a questionnaire from the network operator. It was observed that similar to situation of tariff plans, even with context-led fine-tuning, the fine-tuned LLM models exhibit erroneous processing in the extraction of MCC and MNC from the input. Phrases such as 'ipv6-prefix' is erroneously considered as PLMN ID, leading to extraction of MCC and MNC from the input prompt. The following example illustrates the issue.

Invalid input for MCC and MNC

```
<s>[INST] <<SYS>> You are an AI model tasked with the job of extracting MCC
and MNC in the given format: [[MCC],[MNC]] from the given prompt. If no con-
-text of MCC or MNC is found, give response as "not a valid input". <</SYS>>
ipv6-prefix 272925 [/INST]
Output: Assistant: [[272, 925]]
```

To validate the models with and without input filters, we created a dataset of 400 MCC and MNC samples, evenly split between valid and invalid inputs. As discussed in Section IV-A, the model is expected to correctly identify invalid inputs (0s) and accurately extract MCC and MNC values from valid samples (1s), with the stringent rule that the prediction is deemed incorrect even if one extracted value is incorrect. Table II shows that the use of an input filter significantly improves the models' abilities to identify invalid inputs. Furthermore, as shown in Table II-Row(b), the GPT-3.5-Turbo fine-tuned model with an input filter has proven to be the best-performing model.

Table III compares the performance and configuration details of various models along with their F1 scores, a harmonic mean of precision and recall [25]. Models with input filters consistently achieved higher F1 scores, demonstrating improved accuracy in distinguishing valid and invalid tariff plans, as illustrated in Fig. 2. For instance, the fine-tuned GPT-3.5 model with an input filter attained a perfect F1 score of 1 (Table III-Row(b)), compared to 0.962 without it (Table III-Row(a)). The Llama2-7B model also showed significant improvements with input filter (Fig. 2-Column(F)). The table outlines key parameters such as learning rate, epochs, batch size, and configurations like LoRA scaling for Llama2-7B. Fig. 2 further emphasises the enhanced robustness of the pipeline against invalid inputs, highlighting the critical role of input filtration in optimising automated network configuration, with fine-tuned GPT-3.5 with input filter emerging as most effective across all approaches.

VI. Conclusion and Future Work

In this work, we proposed an LLM-based pipeline that converts input tariff plans and other additional parameters like PLMN identifier to related configuration and thus provision complex networks, offering an advanced alternative to traditional non-AI OSS/BSS systems. The novelty of the method compared to contemporary intent-based approaches is, that it addresses robustness i.e. errors, hallucinations with the help of an input filter (a deep neural network), and supervised fine-tuning of large language models. The presented solution tackles the complexities of mobile network configuration in consumer and enterprise networks. The generation of accurate configuration is essential, as misconfigurations could disrupt critical services like voice and emergency communications. Our method efficiently derives relevant keywords and values, eliminates erroneous inputs, and maps them to well-defined APIs, enabling a deterministic, reliable approach to network configuration, making it viable for real-world deployment. For future improvements, we aim to enhance model performance using reinforcement learning through human feedback (RLHF) [26], [27], which updates model weights by rewarding accurate responses and penalizing errors. Additionally, we plan to employ Retrieval-Augmented Generation (RAG) to improve LLM outputs by incorporating configuration guides as supplementary information. Upgrading base models to Llama-3 and GPT-4 will further improve fine-tuning results. By studying usage patterns, the AI engine could also tailor personalised tariff plans to maximise customer satisfaction and benefit service providers. To reduce human intervention, we propose several strategies: (a) Automation: Leverage functional and system automation testing to validate configurations in a staging environment before applying them in production. To further optimise this, use of feature interaction map can help identify test suites related to specific network nodes (new features or configurations) and their interactions. After identifying this smaller test suite, it can be tested in a staging environment to confirm success before applying the configuration to the live network. (b) Continuous Learning: Automating the feedback loop from validated configurations will enhance the RLHF process. We believe that as a result of the study in this paper, a significant challenge faced by network operators during the roll-out of tariff plans can be resolved, enabling a transition to a higher level of abstraction in network provisioning.

References

[1] A. R. Choudhary, "Policy-based network management," *Bell Labs Technical Journal*, vol. 9, no. 1, pp. 19–29, 2004.

[2] Cisco, "Ultra cloud core 5g session management function, release 2020.02 - configuration and administration guide - sample smf configuration [cisco ultra cloud core - session management function]," 2020. [Online]. Available: https://tinyurl.com/mry2f3xr

[3] A. Leivadeas and M. Falkner, "A survey on intent-based networking," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 625–655, 2023.

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

[5] K. Dzeparoska, J. Lin, A. Tizghadam, and A. Leon-Garcia, "Llm-based policy generation for intent-based management of applications," in *2023 19th International Conference on Network and Service Management (CNSM)*, 2023, pp. 1–7.

[6] A. Fuad, A. H. Ahmed, M. A. Riegler, and T. Čičić, "An intent-based networks framework based on large language models," in *2024 IEEE 10th International Conference on Network Softwarization (NetSoft)*, 2024, pp. 7–12.

[7] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu, and G. Wang, "Instruction tuning for large language models: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2308.10792

[8] L. Wang, J. He, S. Li, N. Liu, and E.-P. Lim, "Mitigating fine-grained hallucination by fine-tuning large vision-language models with caption rewrites," 2023. [Online]. Available: https://arxiv.org/abs/2312.01701

[9] TMForum, *Frameworx 19.0 Product Conformance Certification Report Tecnotree DOCS -Digital Online Charging System v5.0.2*. TMForum, 2020. [Online]. Available: https://tinyurl.com/3bktx4be

[10] I. Ahmad, E. Suwarni, R. I. Borman, Asmawati, F. Rossi, and Y. Jusman, "Implementation of restful api web services architecture in takeaway application development," p. 132–137, Oct 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9649679/

[11] B. Peng, C. Li, P. He, M. Galley, and J. Gao, "Instruction tuning with gpt-4," 2023. [Online]. Available: https://arxiv.org/abs/2304.03277

[12] P. H. Martins, Z. Marinho, and A. F. T. Martins, "Joint learning of named entity recognition and entity linking," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, F. Alva-Manchego, E. Choi, and D. Khashabi, Eds. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 190–196. [Online]. Available: https://aclanthology.org/P19-2026

[13] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," 2023. [Online]. Available: https://arxiv.org/abs/2211.01910

[14] A. Roy, "Recent trends in named entity recognition (ner)," Jan 2021. [Online]. Available: https://arxiv.org/abs/2101.11420

[15] R. Gozalo-Brizuela and E. C. Garrido-Merchán, "A survey of generative ai applications," Jun 2023. [Online]. Available: https://arxiv.org/abs/2306.02781

[16] OpenAI, "Openai api," 2022. [Online]. Available: https://platform.openai.com/docs/models

[17] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, and B. Fuller, "Llama 2: Open foundation and fine-tuned chat models," Jul 2023. [Online]. Available: https://arxiv.org/abs/2307.09288

[18] R. M. Schmidt, F. Schneider, and P. Hennig, "Descending through a crowded valley - benchmarking deep learning optimizers," *CoRR*, vol. abs/2007.01547, 2020. [Online]. Available: https://arxiv.org/abs/2007.01547

[19] L. J. Miranda, Ákos Kádár, A. Boyd, S. V. Landeghem, A. Søgaard, and M. Honnibal, "Multi hash embeddings in spacy," 2022. [Online]. Available: https://arxiv.org/abs/2212.09255

[20] "Preparing your dataset with gpt during fine-tuning," 2022. [Online]. Available: https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset

[21] "A practical introduction to llm fine-tuning," 2023. [Online]. Available: https://tinyurl.com/mvayjcna

[22] "Llama 2: Open foundation and fine-tuned chat models," 2023.

[23] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *CoRR*, vol. abs/2106.09685, 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[24] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023.

[25] D. M. W. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2020. [Online]. Available: https://arxiv.org/abs/2010.16061

[26] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," 2022.

[27] N. Lambert, "Illustrating reinforcement learning from human feedback (rlhf)," Dec 2022. [Online]. Available: https://huggingface.co/blog/rlhf