IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 May 2025, accepted 14 July 2025, date of publication 22 July 2025, date of current version 5 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3591739

APPLIED RESEARCH

Data-Centric Fine-Tuning of Small Language Models for Automatic Extraction of Technical Requirements

LEOPOLD MÜLLER[1,2], NINA SCHWARZ[2,3], LARS BÖCKING[1,2], ANDREAS BERECZUK[4], HANNO STAGGE[4], WOLFGANG KRATSCH[2,3], AND NIKLAS KÜHL[1,2]

[1]Information Systems and Human-Centric Artificial Intelligence, University of Bayreuth, 95447 Bayreuth, Germany
[2]Fraunhofer FIT, 95444 Bayreuth, Germany
[3]FIM Research Center, Augsburg University of Applied Sciences, 86161 Augsburg, Germany
[4]TenneT TSO GmbH, 95448 Bayreuth, Germany

Corresponding author: Leopold Müller (leopold.johann.mueller@fit.fraunhofer.de)

ABSTRACT Training small language models for specific tasks often encounters a significant challenge: the limited availability of high-quality labeled data, which can restrict model performance. This constraint is especially critical in organizations handling sensitive data, where large language models cannot be readily deployed due to privacy concerns, high costs, and dependency on external providers. While existing research demonstrates the effectiveness of large language models in automating text-based tasks, there is a lack of methods tailored to fine-tuning small language models for secure, domain-specific applications with minimal labeled data. To address this gap, we introduce a data-centric fine-tuning method that systematically enhances training data through prompt engineering, making small language model fine-tuning feasible and effective in data-constrained environments. This study evaluates the proposed method on a real-world use case with an industry partner, focusing on the automatic extraction of technical requirements from full-text documents and using both quantitative metrics and qualitative expert assessments. Our findings reveal that the fine-tuned small language model achieves accuracy and consistency comparable to human service providers, while outperforming baseline models, including GPT-4-turbo, on key evaluation metrics. These results underscore the potential of data-centric fine-tuning for adapting small language models to high-stakes, privacy-sensitive tasks, offering a scalable alternative to LLMs enabling their applications to real-world use cases.

INDEX TERMS Data-centric AI, fine-tuning, large language models, requirement extraction.

I. INTRODUCTION

Automating tasks that involve processing and understanding large volumes of natural language text presents both opportunities and challenges for organizations across various industries [1]. Recent advances in large language models (LLMs) have substantially improved the automation of complex, text-based tasks [2]. These improvements suggest that LLMs could, in principle, streamline a wide range of specialized tasks, including automatically extracting technical requirements from unstructured text. Such tasks are critical in numerous domains—for instance, extracting project requirements [3], optimizing requirement management in software engineering [4], or analysis of legal documents [5]. While LLMs like GPT-4-turbo have demonstrated remarkable capabilities and generalizability, many organizations face constraints that prevent them from fully leveraging these models. Chief among these concerns are (a) strict data privacy and security regulations that forbid sharing sensitive information with external providers, (b) high operational costs that make long-term usage of proprietary LLM services prohibitive, and (c) the risk of vendor lock-in and limited customization options. Under these conditions, small language models (SLMs) offer a promising

The associate editor coordinating the review of this manuscript and approving it for publication was Dominik Strzalka.

alternative because they can be deployed fully on-premise or within private infrastructures, providing greater control and adherence to internal policies [6]. However, out-of-the-box SLMs often lack the domain-specific expertise needed to perform specialized tasks at a level comparable to LLMs. Fine-tuning an SLM with proprietary data can help bridge this gap, potentially achieving a performance that rivals larger models [7], [8], [9], [10], [11]. Yet, successfully fine-tuning SLMs hinges on the availability of high-quality, domain-relevant labeled data. In real-world scenarios, such labeled datasets are often scarce, incomplete, or noisy [12]. This scarcity raises important questions about how to construct and augment training data in a way that maximizes model performance [13]. Additionally, recent evidence suggests that training data quality is a fundamental factor influencing model errors, underscoring the need for careful dataset curation [14], [15]. To address these challenges, we adopt a data-centric AI (DCAI) perspective. Rather than focusing solely on model architecture or parameter tuning, data-centric approaches emphasize improving the quality, representativeness, and structure of the training data itself [16]. By systematically enhancing the training dataset—through methods like prompt engineering, data augmentation, and careful selection of source datasets—one can unlock substantial performance gains. This work presents a novel data-centric fine-tuning method that systematically creates and enhances training data to achieve high model performance for SLMs. This method is transferable to any supervised fine-tuning process aimed at optimizing a language model for a specified input-output task. Our work is developed and implemented in collaboration with TenneT, a large European electricity transmission system operator, using their task of automatically extracting and formulating technical requirements from unstructured text as a leading example. Our results show that fine-tuned SLMs can match or surpass the performance of external service providers and even GPT-4-turbo in terms of accuracy, consistency, and compliance with internal standards, all while maintaining on-premise data security and reducing operational costs.

Our contributions are threefold:

1) We introduce a data-centric fine-tuning method designed to systematically generate different training datasets for supervised SLM fine-tuning in data-scarce, specialized domains. This addresses problems organizations face when applying SLMs to their own use cases.
2) We empirically demonstrate that variations in the constructed training datasets lead to differences in model performance, underscoring the importance of experimenting with different dataset variants.
3) We validate our method using a real-world industry case, showing that a fine-tuned SLM can achieve performance levels on par with larger models and human experts, while ensuring data privacy and cost-effectiveness.

The remainder of this paper is organized as follows: Section II reviews the relevant literature on DCAI, related techniques, and requirement extraction. Section III introduces our data-centric fine-tuning method in detail. Section IV describes the industrial real-world use case of TenneT and outlines how we apply our method in this specific context. Section V presents the experimental findings, which we analyze and interpret in Section VI. Finally, Section VII summarizes our contributions and suggests avenues for future research.

II. RELATED WORK

In this section, we review relevant literature on LLM-based requirements engineering, data-centric fine-tuning, and prompt engineering, which form the foundational components of our proposed data-centric fine-tuning method.

LLMs have become a powerful tool for automating various text-based tasks across different industries [2]. In the context of requirements engineering, LLMs demonstrate potential in streamlining tasks such as elicitation, analysis, specification, and validation [17], [18]. However, the process of selecting and fine-tuning LLMs for specific requirements engineering tasks remains a complex challenge [13]. For instance, Gärtner and Göhlich [19] develop an LLM-based system for automatically detecting and classifying contradictions within requirements documents, while Lubos et al. [20] leverage LLMs to assess the quality of software requirements based on defined quality standards. Our work differs from these approaches as it focuses on the extraction of technical requirements from full-text documents. To the best of our knowledge, no existing research has addressed the direct transformation of full text into technical requirements using LLMs. While state-of-the-art SLMs like Meta's Llama-3 [6] are frequently updated, they are not yet as versatile or powerful as larger models like GPT [21]. However, with (supervised) fine-tuning, SLMs can be optimized for specific downstream tasks, often outperforming general-purpose LLMs on domain-specific applications [7], [8], [9], [10], [11].

Although there has been considerable progress in improving SLMs [6] and optimizing their fine-tuning processes [22], these advancements are not directly applicable to the considered real-world scenario. One of the primary challenges is the limited availability and quality of labeled training data, which inhibits the straightforward application of fine-tuning techniques. As Xu et al. [23] emphasizes, data plays a critical role in the fine-tuning of LLMs, and without sufficient or high-quality data, achieving optimal performance is difficult. To address this issue, we introduce a novel method to fine-tuning that is tailored for data-constrained environments. Our proposed method is grounded in the principles of DCAI [16], which shifts the focus from model optimization to enhancing the data itself. Following this approach by systematically improving the dataset, rather than modifying the model, the practical challenges of fine-tuning SLMs in real-world scenarios can be directly tackled. Specifically,

in order to address this need, we propose a data-centric fine-tuning method that fixes the model to a state-of-the-art SLM [6] and focuses on enhancing the data to achieve the desired performance. To the best of our knowledge, no extant work has proposed a method similar to ours [24]. However, there is extensive research on data augmentation techniques for natural language processing (NLP) in general [25], [26], [27], as well as work specifically related to LLMs [28], [29], [30]. These augmentation techniques are a central component of our method, as we systematically integrate various methods to enhance the training data for fine-tuning SLMs. Furthermore, our method aims to provide a systematic procedure for identifying the optimal set of augmentation techniques, recognizing that the effectiveness of these techniques can vary depending on the specific use case [31]. As Gan et al. [32] demonstrate, the quality and diversity of the data directly impact model performance, underscoring the importance of experimenting with different dataset variations. Our method is designed to incorporate this experimental aspect to achieve optimal performance.

Another crucial element of the proposed method is the concept of prompt engineering, which involves crafting optimal task descriptions to elicit the best possible responses from language models [33]. Prompt engineering has proven to be a valuable technique for leveraging the versatility of LLMs, such as GPT, by tailoring them for a wide range of tasks [34], [35], [36], [37], [38].

In our method, we utilize prompt engineering in connection with the versatile LLM GPT-4-turbo to enhance the fine-tuning process of SLMs, specifically by enabling the automatic generation of high-quality augmentations and labels. These tasks would otherwise require significant manual effort. Ongoing research continues to focus on optimizing prompt engineering methods, to which we refer here for further advancements [33].

III. DATA-CENTRIC FINE-TUNING METHOD

In this section, we present our method for data-centric fine-tuning of SLMs for text-to-text generation in a single input-output manner with highly limited data available. Our method involves several key components, including defining the downstream task, generating and augmenting datasets, training the models, and evaluating its performance. The following subsections provide a comprehensive overview of each step of our method.

LLMs have gained popularity due to their ability to perform tasks such as question answering, text rephrasing, and summarization. However, industrial applications often require models to follow predefined structures and consistently produce outputs aligned with specific organizational standards. For instance, a model may need to extract or summarize technical requirements from a given text in a systematic manner. To address this need, we propose a method for fine-tuning text-to-text generation in a single input-output manner. Instead of using a typical chat format, we send only one request to the model and treat its generation as the output. Thereby, one data point is given by the input text, which comprises a prompt and a relevant context. Generally, various tasks can be learned using different prompts. However, we simplify the explanation here by focusing on a single task. So, for each data point x_i, where $i = 1, \ldots, n$ represents the index of the data point and $n \in \mathbb{N}$ describes the number of total data points in the underlying dataset, the prompt q remains constant, while only the context c_i varies from instance to instance (c.f. Figure 1).

Given the input texts, our goal is to fine-tune a SLM, θ_{small}, to generate the required output $y_i := \theta_{\text{small}}(x_i)$ corresponding to this input. In order to illustrate the concept, we consider a prompt q containing an instruction tailored to the specific downstream task. For a given data point x_i, the complete input is formed by appending the context c_i to the prompt q. The output y_i is defined as the label. For the supervised fine-tuning of the SLM θ_{small}, we need a set of data points to update the model's weights during the training process. This requires existing input-output pairs called labeled data so that the predicted output $\hat{y}_i := \theta(x_i)$ can be compared to the ground-truth label y_i for loss calculation. Labeled data is often limited in real-world scenarios and must be reserved for testing and validation. In the following section, we propose the use of prompt engineering to generate additional data points for fine-tuning the model.

A. DATA LABELING

One of the main challenges in fine-tuning language models for downstream tasks in industry is the limited availability of labeled data [12]. The ultimate goal is to have an SLM solve a specific downstream task. In order to fine-tune an SLM on a downstream task we need such labeled data, showcasing the expected response y_i of a model to a given request x_i.

To address this problem, we propose leveraging powerful LLMs, such as GPT-4-turbo, denoted as θ_{large}, which possess highly versatile capabilities in combination with prompt engineering. We assume that θ_{large} is capable of generating predictions of sufficient quality to serve as labels for a given data point x_i. Let T denote the set of possible tasks, including the labeling task L and a set of different augmentation tasks A: $T := \{L \cup A\}$. For each task $t \in T$ exists a set of different prompts $q_j(t)$, where $j \in \mathbb{N}$. In the special case of the labeling task $L \subset T$, the core idea is to systematically find a prompt $q^*(L) \in \{q_j(L) \mid j \in \mathbb{N}\}$, so that θ_{large} can achieve the desired output quality (c.f. Figure 2). For detailed guidelines on prompt engineering, we refer to Liu et al. [33].

The optimized labeling prompt $q^*(L)$ enables us to generate the missing labels for each data point. Once we have generated the exptected responses y_i we can use them to fine-tune a SLM to solve a specific downstream task. In subsequent sections, we will not only describe how to create labels for various data points, but we will also use this technique to generate different augmented versions of data points by consistently applying the same procedure: utilizing

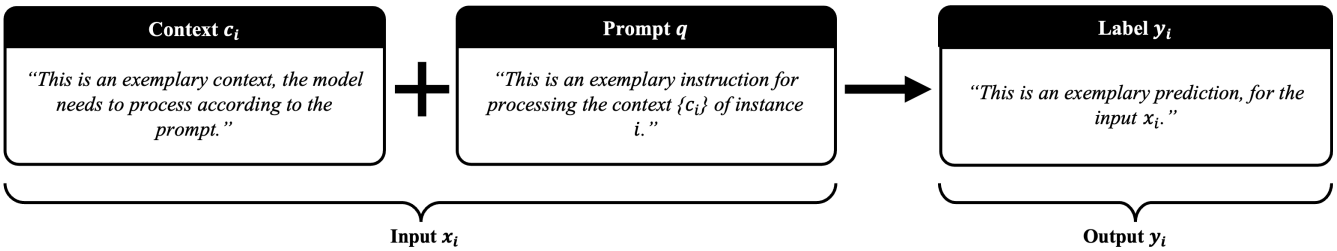


FIGURE 1. Overview of how the input and target output are defined.

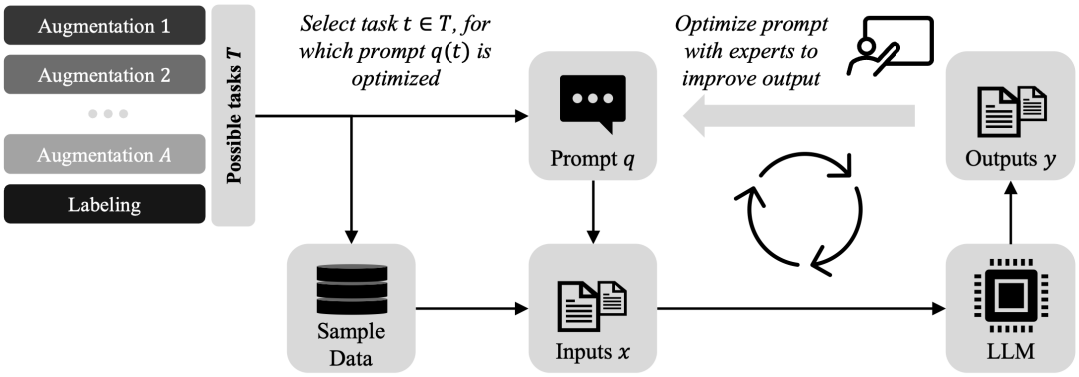


FIGURE 2. Overview of how prompt engineering can be used to solve different tasks $t \in T$. The task can be the creation of labels, but also the creation of different augmentations.

prompt engineering to apply an LLM θ_{large} with the optimized prompt $q^*(A)$ for the set of augmentation tasks $A \subset T$.

B. DATASET GENERATION

Given the downstream task we aim to fine-tune the SLM on, we need to determine the appropriate dataset for training. As this is unknown initially, we suggest following a DCAI strategy. In this section, we propose a method for systematically creating and enhancing training, validation, and test datasets in case a limited amount of labeled data is given (c.f. Figure 3).

1) DATASET POOLS

The starting point for building the training, validation, and test datasets is the ground truth data D_{truth}, which is already labeled and represents the target quality we aim to achieve with our SLM θ_{small}. However, these instances are costly and time-intensive, making them a scarce resource in most scenarios. This dataset is the most valuable resource in the fine-tuning process. Given their rarity, balancing the trade-off between allocating these labeled instances across the training, validation, and test datasets becomes critical. This allocation depends on the availability of ground truth data. The main objective is to balance the number of data points used for training to boost model performance and those needed for a meaningful evaluation. In our case, we assume we have at least a sufficient number of data points for the validation and test datasets, with portions $p_{\text{val}}, p_{\text{test}}, p_{\text{train}} \in (0, 1)$ and $p_{\text{val}} + p_{\text{test}} + p_{\text{train}} = 1$ of the ground truth data. If we have less than sufficient data, we suggest selecting representative unlabeled data and using prompt engineering, as presented in Section III-A, to create labels for each data point. With these portions, we are able to create three fixed datasets out of the ground truth data, namely, $X_{\text{val}}, X_{\text{test}}, X_{\text{train}} \subset D_{\text{truth}}$. The X_{val} and X_{test} datasets are not modified in the following process and are only used for validation and testing. The subset X_{train} forms one of the so called data pools, which is used to generate the final training datasets. The whole set of data pools $D := \{D_1, \ldots, D_S \mid S \in \mathbb{N}\}$ can be extended with any additional data representing the task for which the SLM is to be fine-tuned. For example, different public datasets can be labeled using the strategy described in Section III-A. Following our case, the train dataset X_{train} corresponds to D_s for one $s \in \{1, \ldots, S\}$. As a result, we obtain a set of data pools, a fixed validation dataset, and a fixed test dataset.

It is important to highlight that more data pools can lead to more extensive training, as we will explore different combinations of these to find the best training dataset. However, more data pools provide more options to build training datasets. As we will demonstrate later, our method helps to identify which of the possible data pools should be used for the final training of the SLM.

2) DATASET EXTENSIONS

Given the set of data pools, the next step is to apply different augmentations to each data point in the different

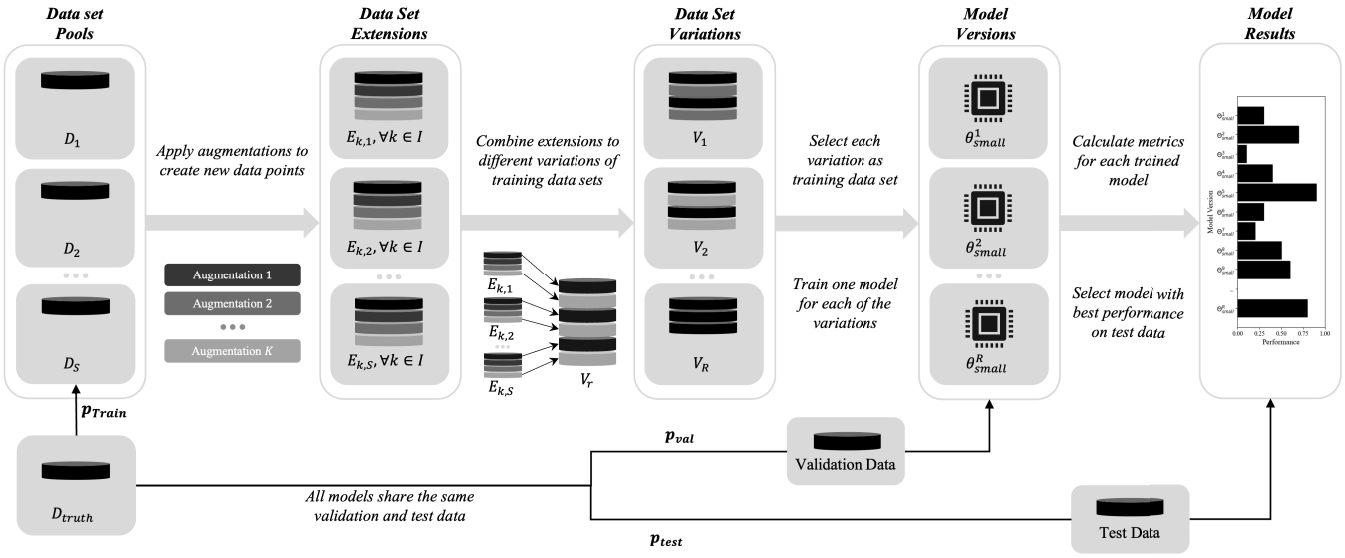


FIGURE 3. Overview of the dataset generation process, illustrating the creation of training, validation, and test datasets from various data pools and augmentations.

data pools. The idea is to create more data points and more variety in the dataset. Similar to the selection of different data pools, we suggest employing multiple augmentation techniques such as backtranslation or rewriting. As with the datapools, a higher number of augmentation techniques increases the training effort because more combinations of these can be explored. The augmentations are not applied to the validation and test datasets, which remain fixed to ensure a realistic evaluation. By A we define the set of augmentations, where each augmentation is represented by a function $a_k, k = 1, \ldots, K$, where $K \in \mathbb{N}$ describes the number of used augmentation methods. More precisely, for each $k \in I = \{1, \ldots, K\}$, the function $a_k : D \rightarrow Y$ represents the optimal prompt regarding the specific augmentation method and Y denotes a set of output data generated by applying the augmentation method a_k to the data points within a data pool in D. So, for each data point $x^s_{i_s} \in D_s$, where s corresponds to the related data pool and i_s is the index of the data points within the data pool D_s, the augmentation method a_k produces an augmented version $\hat{x}^s_{i_s} := a_k(x^s_{i_s})$. As with label creation, careful prompt engineering is essential to ensure that the augmented versions achieve the desired quality. Depending on the type of augmentation, we suggest generating new labels using the large LLM θ_{large} with the task-specific prompt $q^*(L)$, optimized through prompt engineering, for the augmented data points $\hat{x}^s_{i_s}$. By applying each augmentation a_k to each data pool D_s one or multiple times, we obtain a data extension $E_{k,s} := a_k(D_s)$ for each data pool. Each extension consists of the original data points of the data pool D_s and all augmented versions of its instances. It is crucial to view these extensions as sets, as in the next section, we will systematically combine different subsets of each extension to create dataset variations.

3) DATASET VARIATIONS

The next step in our method involves the creation of a set V of dataset variations $V := \{V_1, \ldots, V_R \mid R \in \mathbb{N}\}$, which consist of various combinations of subsets from the dataset extensions $E_{k,s}$. These variations are essential for systematically exploring the impact of different data pools and augmentation techniques on the performance of the SLM θ_{small}. To manage complexity and avoid excessive variations, we suggest restricting the possible combinations. For instance, combinations without the data pool D_{truth} could be prioritized less, as these are probably the most valuable data points. This ensures that the number of dataset variations R, remains feasible while still providing valuable insights into the most effective data augmentation and pooling strategies.

C. MODEL TRAINING

From the previous steps, we have a set of dataset variations V building R different variations out of the extensions that serve as individual training datasets, along with a fixed validation dataset X_{val} and a fixed test dataset X_{test}. Given these resources, we propose a standard fine-tuning process for the SLMs. We recommend using a recent state-of-the-art SLM and keeping it fixed, in accordance with the DCAI paradigm, as our objective is to improve performance by systematically creating and enhancing the training data [16]. For each dataset variation $V_r \in V, r \in \{1, \ldots, R\}$, an SLM θ^r_{small} is trained. All variation-specific SLMs are validated on the same fixed validation dataset X_{val} during the training process. This consistent validation approach allows us to apply early stopping to balance under- and overfitting and ensures that the validation loss across different models is comparable. By following this process, we aim to systematically determine which dataset variation V_r* leads to the best performance for the SLM θ_{small} on the given task.

The final evaluation of the best-performing model will be conducted on the fixed test dataset to ensure an unbiased assessment of its generalization capabilities.

D. EVALUATION FRAMEWORK

After all models are trained, we receive a set of R different SLMs $\Theta := \{\theta^1_{\text{small}}, \theta^2_{\text{small}}, \ldots, \theta^R_{\text{small}}\}$. The differences only appear from the dataset variation on which the model was trained. Now, for each model, we generate predictions on the fixed test dataset. Given the predictions of each model, we perform a quantitative and a qualitative evaluation.

1) QUALITATIVE EVALUATION

For the qualitative evaluation, domain experts are required to evaluate the predictions of the models. These insights ensure that the properties that can not be covered by the defined metrics in the quantitative evaluation are reviewed. If there are fundamental errors in the prediction of the models, this can lead to a restart of the dataset generation process Section III-B. The limitations should be addressed in the prompt engineering process depicted in Figure 2.

2) QUANTITATIVE EVALUATION

The loss of the validation data can give a first indication of the performance of the different models. However, the evaluation on the validation data can lead to a biased performance evaluation. In addition, the loss does not necessarily consider how well the trained SLM can perform on the downstream tasks. To measure the quality of an output, several characteristics such as length, structure, or wording should be analyzed. On benchmark datasets such asto Wang [39], several metrics are defined that try to measure how well the model performs the task. A set of different metrics should be used to measure the performance of each trained model [40]. These can be existing metrics as BLEU [41], but for novel downstream tasks, we highly recommend defining new metrics that suit the different properties of the underlying task. This can require collaboration with domain experts. The metrics are then combined and used to select the best dataset variation V^*_r, which leads to the best-performing model θ^*_r. This is achieved by consistently comparing the performance of different models on the same test dataset, ensuring a reliable evaluation.

IV. EXPERIMENTAL SETUP

To evaluate our method, we apply it to a real-world use case from the industry. In the following sections, we describe the use case, define the components we use to solve the task with our method and present the results we obtain.

A. DOWNSTREAM TASK

TenneT, a large European electricity transmission system operator with over 8000 employees, relies on extensive technical specification documents to define the requirements for constructing, operating, and maintaining high-voltage power systems. These documents contain not only technical requirements but also contextual details, best practices, and supplementary advice. To integrate these requirements into a structured database—where additional information can be linked as metadata—it is necessary to extract, refine, and standardize them. This process involves distinguishing requirements from non-requirement text, converting them into stand-alone statements, and aligning them with established guidelines, such as those provided by the INCOSE Guide for Writing Requirements [42]. Accurately identifying and refining requirements is complex and requires deep domain knowledge and engineering expertise. Historically, this task has been supported by external service providers, who propose initial requirement drafts for review by TenneT's internal domain experts. However, relying on external providers presents several challenges, including potential gaps in domain expertise, variability in output quality, and prolonged turnaround times. These factors motivate the exploration of automated approaches to improve efficiency, reduce costs, and enhance consistency. The downstream task of automatically extracting and reformulating technical requirements is well-suited to our data-centric fine-tuning method for several reasons. First, the sensitive nature of TenneT's data necessitates on-premises processing, making large, proprietary LLMs less feasible due to privacy and data ownership concerns. SLMs can be fine-tuned within secure environments, ensuring compliance with internal policies and regulatory frameworks. Second, the scarcity and complexity of domain-specific training data underscore the importance of a data-centric strategy that systematically refines and augments the available dataset to achieve high model performance. While this use case focuses on high-voltage power systems, the approach is not limited to this specific domain. Any industry or organization facing analogous challenges in an input-output manner—such as extracting structured requirements from large volumes of technical, legal, or regulatory text—can apply the same method. This versatility is due to the fact that the methodology is designed to be adaptable, regardless of the nature of the downstream tasks or the specific types of data involved.

Figure 4 provides a synthetic example illustrating the input-output format for this downstream task. The output is a standardized dictionary of requirements, ready for direct integration into a database and subsequent retrieval, analysis, and maintenance.

B. DATA LABELING

The ground truth data available from TenneT is limited and sufficient only to cover the validation and test datasets, with a minimal amount available for training. To address this, we employ prompt engineering to generate additional training data points by labeling unlabeled data points (see Section IV-C1). For this purpose, we use GPT-4-turbo as the LLM θ_{large}. We conduct the prompt engineering process in collaboration with industry experts to ensure that the generated data meets the required quality standards. We, in part, incorporate guidelines [42] for writing requirements in a comprehensive manner to enable a standardized format.

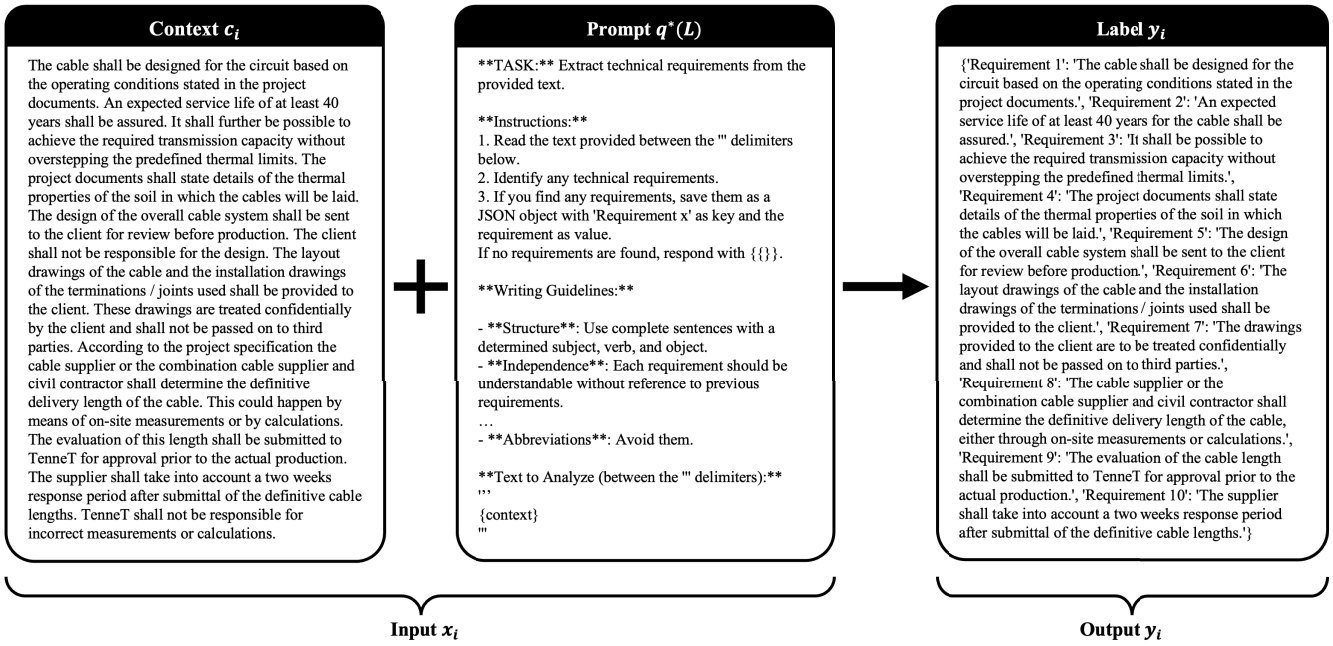


FIGURE 4. Synthetic example for input and output for the TenneT use case.

The final prompt $q^*(L)$, used for the data labeling task $L \subset T$, is shown in Figure 4.

C. DATASET GENERATION

In the following, we describe the inner part of the data-centric fine-tuning method to systematically create and enhance the training data.

1) DATASET POOLS

The available ground truth data from TenneT includes a limited amount of processed full texts. These were manually revised by an external service provider. To ensure meaningful validation and evaluation, we use only a small portion of this data for training to assess its impact on model performance. Specifically, we set $p_{\text{train}} = 0.2$, $p_{\text{val}} = 0.16$, and $p_{\text{test}} = 0.64$. The portion used for training represents the first of three data pools. The second data pool comprises a set of unlabeled full texts from various documents provided by TenneT. For the third data pool, we utilize external publicly available data. Since there is no dataset specifically available for requirements of technical systems such as powerlines, we use the PURE dataset [43], which describes requirements for software and websites, as a related domain. Given that the second and third data pools lack labels, we first split the full texts into separate chunks. We then create labels for each chunk to form input-output pairs for the datasets. These labels are generated using GPT-4-turbo with our engineered prompt from Section IV-B in collaboration with domain experts.

2) DATASET EXTENSIONS

Given the data pools, we define three different augmentations to increase the number of data points for each pool. Thus, for each original data point, there are four augmented instances. After generating the augmented instances, new labels are created for all augmented instances using GPT-4-turbo and the labeling prompt from Section IV-B. As a result, each extension $E_{k,s}$ contains four times the number of data points as the original data pool D_s. This is because, for each original data point, three augmented versions are added (see Table 1). We validate the quality of the augmentations with domain experts and include their expertise during the prompt engineering phase.

TABLE 1. Description of augmentations applied to each data pool.

$a_k(x_i)$	Description
$a_1(x_i)$	For each input context c_i of x_i, GPT-4-turbo is used to create a rewritten version without altering the content itself.
$a_2(x_i)$	For each input context c_i of x_i, GPT-4-turbo is used to translate the text into another language and back to English to generate an augmented version of the text.
$a_3(x_i)$	For each input context c_i of x_i, the numbers in the text are randomly replaced by other numbers to prevent the model from learning specific numbers from one instance and instead focus on accurately extracting numbers from the given context.

3) DATASET VARIATIONS

Next, we systematically combine the different dataset extensions to create a diverse set of dataset variations. The goal of creating these variations is to thoroughly examine how different combinations of data pools and augmentation techniques influence model performance. To maintain manageability and prevent combinatorial explosion, we limit the combinations. Specifically, if an augmentation $a_k(x_i)$ is selected for inclusion in a dataset variation V_r, we apply it uniformly across all data pools D_s included in that specific variation. For example, if the augmentation

"back-translation" is chosen for a particular dataset variation, it is applied to all data pools within that variation.

Table 2 explicitly lists the combinations of the three data pools (D_1, D_2, and D_3) with the three augmentation techniques (a_1, a_2, and a_3) across dataset variations V_{01} to V_{48}. For instance, variation V_{03} combines all three data pools (D_1, D_2, and D_3) with the augmentation method "back-translation" (a_2) applied uniformly across them. Dataset variation V_{33} includes only data pool D_2 without augmentations. Variation V_{13} combines data pools D_1 and D_2 along with their augmented versions generated through the "rewriting" augmentation method (a_1). Data pool D_3 and its augmented versions are excluded from variation V_{13}.

We note that some data points may not contain extractable requirements for several reasons: they may comprise only contextual or descriptive information, provide informal recommendations rather than formal requirements, or result from incomplete text segments due to splitting. Additionally, since labels for some data points are automatically generated, occasional mislabeling might occur. To address this issue, we introduce a rebalancing step after combining all data points within a dataset variation: if the proportion of non-requirement instances exceeds one percent of the total dataset, we randomly remove excess non-requirement data points until this threshold is met. This step ensures a balanced and task-relevant dataset, making the resulting dataset variations more directly comparable.

D. MODEL TRAINING

For the SLM, we use the state-of-the-art model "unsloth/llama-3-8b-Instruct-bnb-4bit" from Hugging Face [6]. We choose the instruction-tuned version as it already understands the input-output principle, having been trained in a chat format. Additionally, the quantized version facilitates faster training, which is advantageous for our data-centric method as we train and compare many different models. To further enhance the fine-tuning process, we integrate LoRA [44]. Training and evaluation are conducted sequentially using a single Nvidia A100 GPU with 40 GB of memory. Completing all 48 dataset variations requires approximately 35.2 hours in total runtime. For the creation of labels and augmentations, we use approximately 16 million tokens. While our method can be applied to any model, the choice of model may vary depending on the specific tasks. For multiple complex tasks, larger models might be preferable. However, as the performance of these models continues to improve, even smaller models could also be effective candidates.

E. PERFORMANCE EVALUATION

In the following, we describe the evaluation metrics used for the industry use case and how the qualitative evaluation is conducted.

1) QUALITATIVE EVALUATION

For the qualitative evaluation, we engage domain experts from the industry partner TenneT to assess whether the quality of the extracted requirements meets their standards. This qualitative assessment aims to verify if the models are generally capable of solving the task effectively and to cross-check the quantitative results. In weekly workshops, four TenneT domain experts analyzed model predictions on a test dataset of size 185 and compared them with the requirements extracted by the external service provider. They reviewed model outputs against requirements produced by an external service provider, evaluating both contextual accuracy and adherence to syntactic guidelines (based on the INCOSE Guide for Writing Requirements [42]).

2) QUANTITATIVE EVALUATION

As there is no specific metric to evaluate the accuracy of requirement extraction, we utilize several well-known metrics to gain insights into the model performances. We apply both statistical and LLM-based scores to the predictions on the test dataset, calculating the average for all test instances. The metrics used include Absolute Error, LangChain Accuracy Score, LangChain Criteria Score, Rouge L, Rouge 1, Rouge 2, SacreBLEU, Google BLEU, Meteor, Bert Precision, Bert Recall, and Bert F1 [41], [45], [46], [47], [48]. The absolute error measures the difference between the number of requirements extracted and the number of requirements extracted by the external service provider. The LangChain Accuracy Score uses GPT-4-turbo to evaluate how closely the model output aligns with the reference answers based on a score between 1 (The answer is completely unrelated to the reference) and 10 (The answer is completely accurate and aligns perfectly with the reference). The LangChain Criteria Score uses GPT-4-turbo to evaluate the output based on multiple custom criteria (completeness of the extraction, accuracy of the content, sentence structure and dictionary format). If all criteria are met, the score assigned to this prediction is one. If one or more criteria are not fulfilled, the score is zero. Additionally, to assess the mean performance and consistency of the models, we include the standard deviation (std) of the scores achieved on the test dataset. For the Criteria Score, we do not report the std due to the binary characteristic.

V. RESULTS

In this section, we present the results of our method applied to the industry use case of TenneT.

A. QUALITATIVE RESULTS

The results indicate that the performance of all models is on par with that of the external service provider. The quality of the model outputs appears to be sufficient for practical application. Unlike the manual extraction performed by the external service provider, which is susceptible to human errors, the automated pipeline of the SLM minimizes such risks and allows for systematic enhancements to ensure greater accuracy and reliability. Additionally, the quality of the SLM outputs seems to be more consistent, as the external service provider's quality can vary depending on

TABLE 2. Overview of the different dataset variations V_r. Note that if an augmentation is included, it is included for all data pools D_s used in the dataset variation V_r. A filled circle (●) indicates the presence of a specific augmentation or data pool in the corresponding dataset variation.

	Dataset Variations V_{01} to V_{16}															
	V_{01}	V_{02}	V_{03}	V_{04}	V_{05}	V_{06}	V_{07}	V_{08}	V_{09}	V_{10}	V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	V_{16}
D_1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
D_2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
D_3	●	●	●	●	●	●	●	●								
$a_1(.)$					●	●	●	●					●	●	●	●
$a_2(.)$			●	●			●	●			●	●			●	●
$a_3(.)$		●		●		●		●		●		●		●		●
	Dataset Variations V_{17} to V_{32}															
	V_{17}	V_{18}	V_{19}	V_{20}	V_{21}	V_{22}	V_{23}	V_{24}	V_{25}	V_{26}	V_{27}	V_{28}	V_{29}	V_{30}	V_{31}	V_{32}
D_1	●	●	●	●	●	●	●	●								
D_2									●	●	●	●	●	●	●	●
D_3	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
$a_1(.)$					●	●	●	●					●	●	●	●
$a_2(.)$			●	●			●	●			●	●			●	●
$a_3(.)$		●		●		●		●		●		●		●		●
	Dataset Variations V_{33} to V_{48}															
	V_{33}	V_{34}	V_{35}	V_{36}	V_{37}	V_{38}	V_{39}	V_{40}	V_{41}	V_{42}	V_{43}	V_{44}	V_{45}	V_{46}	V_{47}	V_{48}
D_1									●	●	●	●				
D_2	●	●	●	●	●	●	●	●								
D_3													●	●	●	●
$a_1(.)$					●	●	●	●	●					●		
$a_2(.)$			●	●			●	●		●					●	
$a_3(.)$		●		●		●		●			●					●

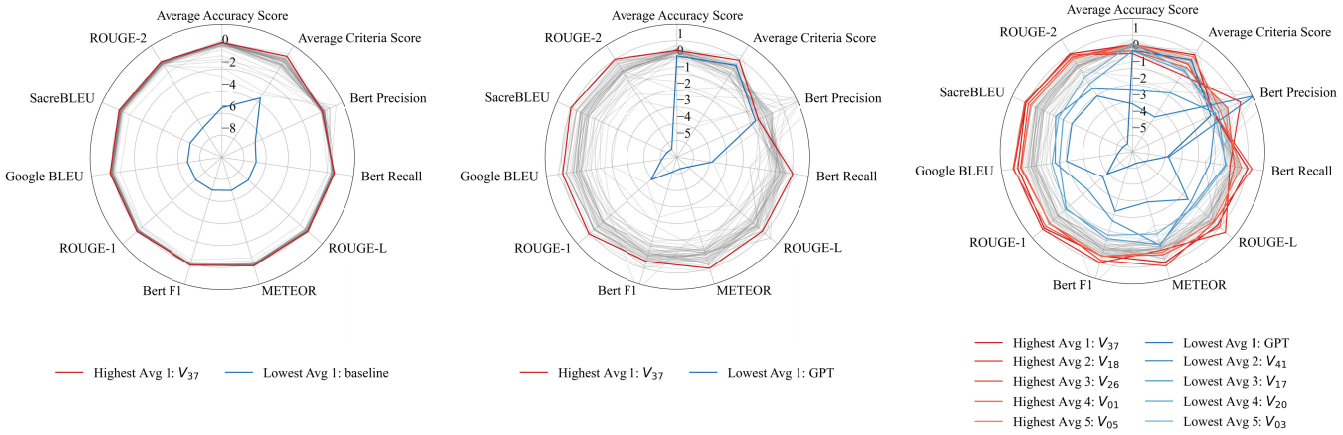
the individual performing the extraction. However, in this particular use case, the differences between the models are too small to be discerned through qualitative analysis alone. Errors that occur are typically due to missing context within given requirements or text phrases that are formulated as requirements but require external knowledge to correctly classify as additional information rather than actual requirements. Nevertheless, the inaccuracies and performance differences among the models are too minor for a meaningful qualitative assessment, underscoring the importance of quantitative evaluation.

B. QUANTITATIVE RESULTS

First, we evaluate the different models derived from various dataset variations by examining their mean performances across the metrics presented in Section IV-E2. The complete results are presented in Table 3. The selection of metrics is non-trivial, and we aim to objectively report the observed effects of our data-centric method on each of them.

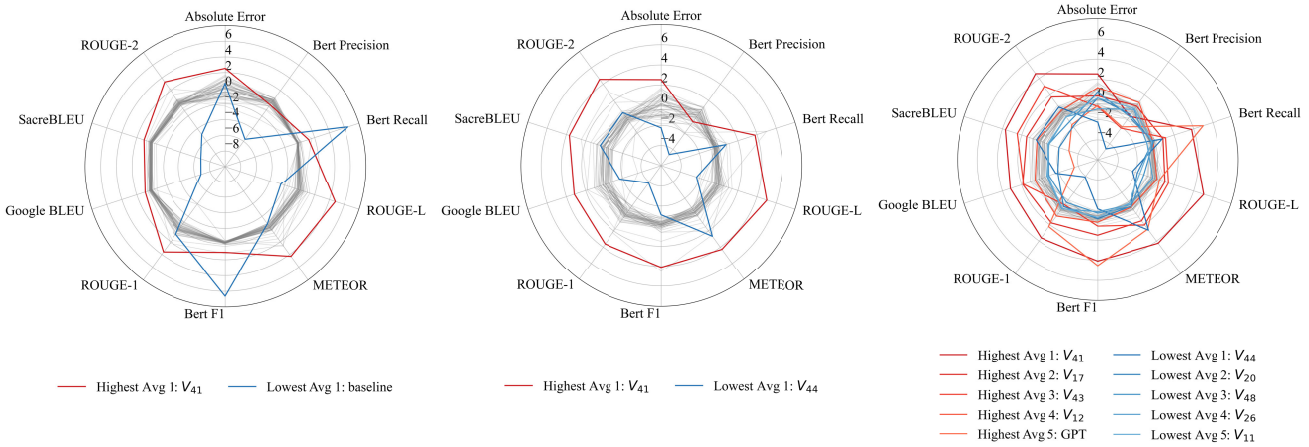
Figure 5(a) illustrates the improvements of each model compared to the baseline SLM, which is the out-of-the-box pretrained model without any fine-tuning. Our central conclusion of this analysis is that even the most elementary fine-tuning, conducted using our data-centric method, outperforms the baseline to a considerable extent. Figure 5(b) provides a direct comparison between the dataset variations and GPT-4-turbo. It is noteworthy that all dataset variations consistently outperform GPT-4-turbo across the majority of metrics, although GPT-4-turbo remains competitive in specific metrics, such as Bert Precision and Average Criterion Score. Figure 5(c), provides a more detailed exploration of the range of performances achieved by the different datasets. The best-performing variations consistently yield high performances, while even the least effective dataset variations show competitive results in certain metrics. In addition to mean performance, the consistency of the models is also crucial. Figure 6 displays the std for each model across all metrics.

It is important to note that a consistent model does not necessarily imply high quality. For instance, while the baseline model appears to be the most consistent (except for the accuracy score using LangChain), it performs worse across all metrics compared to the fine-tuned models as shown in Figure 6(a). Figure 6(c) illustrates that the most effective dataset variations exhibit minimal variation across metrics, indicating consistent and reliable performance. Conversely, the least effective variations demonstrate a greater degree of inconsistency, with greater variation observed in the std in metrics across multiple runs. For a more detailed analysis of the individual impact of each individual augmentation variant, we provide an ablation study in Section VII.



(a) Baseline (b) Performance range (c) Performance consistency

FIGURE 5. Detailed analysis of the performances achieved by the different dataset variations with respect to the discussed metrics. Each metric has been standardized by its mean and standard deviation to facilitate better comparison. (a): Comparing all dataset variations vs. the baseline (not fine-tuned model). (b): Highlighting the range of performances acchived by the best and worst dataset variation (excluding the baseline). (c): Showing the consistency of top 5 and worst 5 dataset variations.



(a) Baseline (b) Std range (c) Std consistency

FIGURE 6. Analysis of the performance variation (std) achieved over multiple runs by the different dataset variations with respect to the discussed metrics. (a): Comparing all dataset variations vs. the baseline (not fine-tuned model) with respect to the std of each metric across multiple runs. (b): Highlighting the different levels of robustness of performances, highlighting the highest and lowest std of dataset variation. (c): Highlighting the std of top 5 and worst 5 dataset variation.

C. DECISION RULE

As the qualitative results shows that all models are principally meeting the required quality standard, we decide using the quantitative analysis. We rank the models regarding mean performance and consistency and select the model with the best-weighted rank over both categories. The decision rule highly depends on the use case and should be elaborated on by domain experts. For example, it should be discussed which metrics should be considered and how to prioritize performance and consistency. In our use case, dataset variation V_{05} leads to the most favorable model. It consists of all three data pools and the augmentations of these using $a_1(.)$.

VI. DISCUSSION

Our empirical results demonstrate that a data-centric approach to fine-tuning SLMs can yield performance levels that are both quantitatively and practically comparable to human-generated requirement extractions. Moreover, several fine-tuned models perform competitively against large-scale LLMs like GPT-4-turbo. By systematically curating data pools and applying targeted augmentation techniques, we effectively mitigate the initial scarcity of labeled data and improved both the mean performance and consistency of the model across various evaluation metrics. From a methodological standpoint, this study highlights the importance of data quality, diversity, and representation for specialized tasks. Rather than relying on large general-purpose models, we show that refining a smaller model using a carefully selected and augmented dataset of domain-relevant texts can meet the specific requirements of a real-world industrial application. Notably, several dataset variations outperform GPT-4-turbo according to the majority of our metrics, including LLM-based scoring measures, showing

that fine-tuned SLMs can reach or exceed the performance of their larger counterparts on domain-specific downstream tasks. Qualitative assessments by domain experts further confirm that the SLM's outputs were sufficiently accurate and consistent for practical use, reducing the need for costly external service providers. This expert validation, combined with quantitative metrics, indicates that SLMs refined through a data-centric approach can reliably achieve production-level quality. In fact, the tuned SLM produced more consistent and reliable results than human service providers in this particular case.

A. PRACTICAL & THEORETICAL IMPLICATIONS

These findings carry direct implications for organizations integrating language models into sensitive, domain-specific workflows. Rather than adopting large proprietary LLMs—which may raise concerns related to data privacy, high operational costs, or vendor lock-in-organizations can fine-tune SLMs on-premises. This method reduces data exposure risks, improves cost efficiency, and maintains performance aligned with organizational standards. The generalization of the method to other domains, therefore, offers transformative potential, especially for applications that are affected by one of the three concerns mentioned.

From a theoretical perspective, our work supports the data-centric AI principle, showing that performance in specialized domains depends more on data suitability and preparation than on model size alone. By focusing on prompt refinement, dataset augmentation, and domain-specific labeling methods, substantial gains in predictive accuracy and consistency can be achieved without resorting to resource-intensive models. In addition, our data-centric fine-tuning method can serve as a template for developing more advanced fine-tuning pipelines, providing a starting point for future refinement and expansion.

B. LIMITATIONS & FUTURE WORK

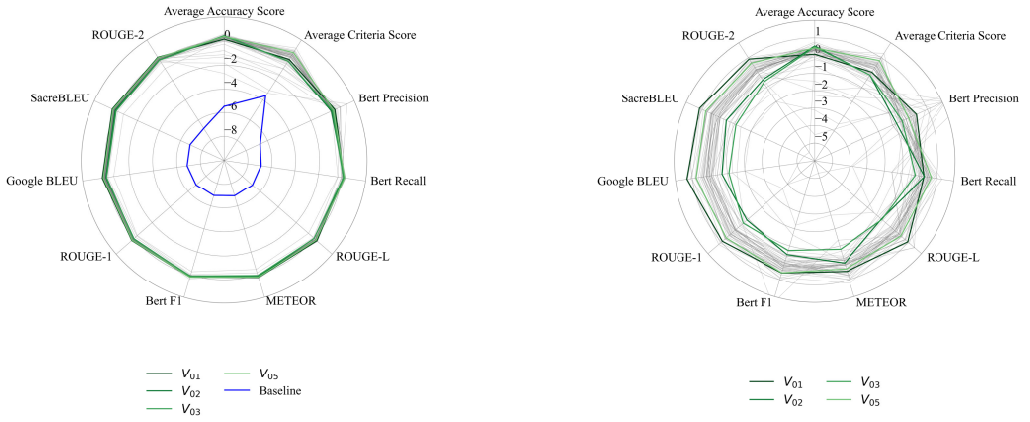
A key assumption of our method is the availability of an LLM capable of accurately generating labels via prompt engineering. This implies that the quality of these labels could set the upper bound for the quality of the fine-tuned SLM. However, our results show that a fine-tuned SLM can, in some cases, exceed the performance of the LLM used to generate its training labels. The prompts for labeling play a crucial role as they significantly impact the quality of the generated labels. Currently, prompt engineering is performed manually with the help of experts, which is a time-intensive process. Additionally, there are no clear guidelines to determine when a prompt is sufficiently optimized. Future work could focus on developing and including an automated prompt engineering method that iteratively improves the prompt to achieve the desired quality on sample instances [36]. This could reduce the time required from experts and automate the entire method. In addition, more customized evaluation metrics could improve the evaluation of different models trained on different training datasets. The two LLM-based scoring methods (LangChain Accuracy and LangChain Criteria Score) offer scalable and automated evaluation, but they inherently rely on the underlying language model and may introduce systematic biases. Future work could investigate the impact of the language model used and the design of these scores. Currently, our method is evaluated on a single downstream task and is not meant to be used in zero-shot or cross-task scenarios. This limitation reinforces the importance of clear task definition and targeted data preparation in domain-specific applications. Extending our method to automatically generate training datasets optimized for different tasks, including other domains (e.g., medical or legal requirement extraction) or multi-task scenarios (e.g., simultaneous requirement and risk extraction), could broaden the applicability and efficiency of data-centric fine-tuning pipelines. For instance, in healthcare, domain-specific LLMs have already shown promising results in data-scarce and data-sensitive environments, such as medical documentation [11]. Similarly, in the development of robust IoT security, integrating generative AI and LLMs holds great potential for tasks like cyber threat detection or penetration testing, although it requires careful consideration of data privacy [49]. Furthermore, the required computational resources depend heavily on the chosen model and the amount of available data. Future work could conduct multiple benchmarking analyses based on different models and dataset sizes. These analyses would provide more comprehensive and precise recommendations for the final training dataset, as well as reveal the impact of these variables.

VII. CONCLUSION

This paper presents a data-centric fine-tuning method for SLMs tailored to the automated extraction of technical requirements, addressing challenges associated with limited labeled data and the need for secure, on-premise processing. Applied to a real-world industry use case, our method systematically enhances training data quality through prompt engineering and augmentation techniques, yielding models that outperform a baseline SLM, rival GPT-4-turbo, and achieve quality on par with external service providers. The method demonstrates how fine-tuned SLMs can maintain performance consistency, reduce costs, and ensure data privacy. While our prompt engineering strategy proved effective, further automation may streamline this process, reducing the need for extensive expert input. Future work could extend the applicability of this method to other domains, tasks, or multi-task settings, thus optimizing model fine-tuning in data-constrained environments.

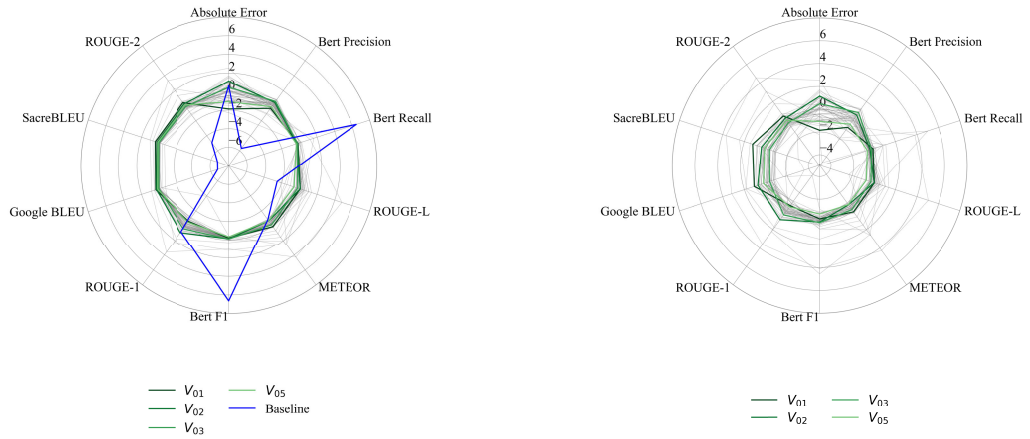
APPENDIX A
ABLATION STUDY

To isolate the impact of individual augmentation strategies, we present an ablation study that isolates the impact of the basic dataset variations used throughout our experi-



(a) Baseline (b) Performance range

FIGURE 7. Detailed analysis of the performances achieved by the different dataset variations with respect to the discussed metrics. Each metric has been standardized by its mean and standard deviation to facilitate better comparison. (a): Comparing all dataset variations vs. the baseline (not fine-tuned model) with focus on dataset variation V_{01} (no augmentation), V_{02} ("random-number replacement", a_3), V_{03} ("back-translation", a_2), and V_{05} ("rewriting", a_1). (b): Highlighting the range of performances (excluding the baseline).



(a) Baseline (b) Std range

FIGURE 8. Analysis of the performance variation (std) achieved over multiple runs by the different dataset variations with respect to the discussed metrics. (a): Comparing all dataset variations vs. the baseline (not fine-tuned model) with respect to the std of each metric on the test dataset with focus on dataset variation V_{01} (no augmentation), V_{02} (random-number replacement, a_3), V_{03} (back-translation, a_2), and V_{05} (rewriting, a_1). (b): Highlighting the different levels of robustness of performances (excluding the baseline).

ments. We highlight four key dataset variations: V_{01} (no augmentation), V_{02} (random-number replacement, a_3), V_{03} (back-translation, a_2), and V_{05} (rewriting, a_1).

Figure 7 and Figure 8 compare these variants against each other, the baseline (pre-trained but not fine-tuned model), and the overall best- and worst-performing variations. While each basic augmentation improves some metrics, none dominates across the entire metric suite. The same holds for standard-deviation plots, which reflect consistency. No single augmentation universally reduces variability, and several other (non-basic) dataset variations outperform the highlighted ones in both mean performance and robustness. Taken together, these findings underscore the need for systematic experimentation with multiple dataset variations rather than relying on any single augmentation strategy.

APPENDIX B
EXTENSIVE RESULTS

Table 3 shows the quantitative performance metrics corresponding to Figure 5.

TABLE 3. Detailed analysis of the performances achieved by the different dataset variations with respect to the discussed metrics.

	Average Accuracy Score	Average Criteria Score	Absolute Error	Bert Precision	Bert Recall	ROUGE-L	METEOR	Bert F1	ROUGE-1	Google BLEU	SacreBLEU	ROUGE-2
baseline	0.70	0.29	5.30	0.83	0.73	0.24	0.21	0.77	0.30	0.10	0.06	0.17
GPT	0.95	0.52	2.31	0.90	0.90	0.50	0.60	0.90	0.62	0.39	0.36	0.44
V_{01}	0.95	0.49	2.14	0.90	0.92	0.54	0.64	0.91	0.65	0.43	0.41	0.48
V_{02}	0.97	0.48	2.57	0.90	0.92	0.52	0.64	0.91	0.64	0.42	0.40	0.47
V_{03}	0.96	0.48	2.48	0.90	0.91	0.53	0.63	0.91	0.64	0.41	0.39	0.47
V_{04}	0.96	0.49	2.60	0.90	0.92	0.53	0.64	0.91	0.65	0.43	0.41	0.48
V_{05}	0.96	0.53	2.29	0.90	0.92	0.53	0.64	0.91	0.65	0.43	0.41	0.48
V_{06}	0.95	0.51	2.29	0.90	0.92	0.53	0.64	0.91	0.65	0.42	0.40	0.48
V_{07}	0.96	0.52	2.25	0.90	0.92	0.53	0.64	0.91	0.65	0.43	0.40	0.48
V_{08}	0.96	0.52	2.34	0.90	0.92	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{09}	0.96	0.46	2.30	0.90	0.91	0.53	0.64	0.91	0.65	0.42	0.40	0.48
V_{10}	0.96	0.52	2.47	0.90	0.92	0.53	0.65	0.91	0.64	0.42	0.40	0.48
V_{11}	0.96	0.54	2.40	0.90	0.92	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{12}	0.96	0.52	2.46	0.90	0.92	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{13}	0.96	0.52	2.33	0.90	0.92	0.53	0.64	0.91	0.65	0.43	0.41	0.48
V_{14}	0.96	0.49	2.25	0.90	0.92	0.54	0.64	0.91	0.65	0.43	0.41	0.48
V_{15}	0.96	0.50	2.28	0.90	0.91	0.53	0.64	0.91	0.65	0.43	0.41	0.48
V_{16}	0.96	0.51	2.27	0.90	0.91	0.53	0.64	0.91	0.65	0.42	0.40	0.48
V_{17}	0.89	0.41	2.67	0.90	0.91	0.52	0.64	0.90	0.63	0.41	0.40	0.47
V_{18}	0.95	0.47	2.44	0.91	0.91	0.54	0.64	0.91	0.65	0.43	0.41	0.48
V_{19}	0.96	0.51	2.52	0.90	0.91	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{20}	0.95	0.48	2.55	0.90	0.91	0.52	0.64	0.91	0.64	0.41	0.39	0.47
V_{21}	0.93	0.44	2.55	0.90	0.91	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{22}	0.94	0.49	2.42	0.90	0.91	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{23}	0.95	0.47	2.48	0.90	0.91	0.53	0.64	0.91	0.65	0.43	0.41	0.48
V_{24}	0.96	0.51	2.68	0.90	0.92	0.53	0.65	0.91	0.64	0.42	0.40	0.48
V_{25}	0.96	0.54	2.52	0.90	0.92	0.53	0.65	0.91	0.64	0.42	0.40	0.48
V_{26}	0.96	0.50	2.31	0.90	0.92	0.53	0.65	0.91	0.65	0.43	0.41	0.48
V_{27}	0.96	0.48	2.30	0.90	0.92	0.53	0.64	0.91	0.65	0.42	0.40	0.48
V_{28}	0.96	0.48	2.27	0.90	0.92	0.54	0.64	0.91	0.65	0.43	0.41	0.48
V_{29}	0.96	0.53	2.59	0.90	0.92	0.53	0.64	0.91	0.65	0.42	0.40	0.48
V_{30}	0.96	0.50	2.32	0.90	0.92	0.53	0.65	0.91	0.65	0.42	0.41	0.48
V_{31}	0.96	0.51	2.21	0.90	0.92	0.54	0.64	0.91	0.65	0.43	0.41	0.48
V_{32}	0.95	0.51	2.37	0.90	0.92	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{33}	0.97	0.51	2.44	0.90	0.92	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{34}	0.96	0.51	2.34	0.90	0.92	0.53	0.64	0.91	0.65	0.42	0.41	0.48
V_{35}	0.97	0.50	2.17	0.90	0.92	0.54	0.64	0.91	0.65	0.43	0.41	0.48
V_{36}	0.97	0.51	2.39	0.90	0.92	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{37}	0.96	0.54	2.27	0.90	0.92	0.54	0.65	0.91	0.65	0.43	0.41	0.49
V_{38}	0.96	0.55	2.42	0.90	0.92	0.53	0.65	0.91	0.65	0.42	0.41	0.48
V_{39}	0.97	0.54	2.56	0.90	0.91	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{40}	0.96	0.52	2.36	0.90	0.92	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{41}	0.86	0.33	2.99	0.91	0.90	0.52	0.62	0.90	0.62	0.41	0.39	0.46
V_{42}	0.91	0.41	2.42	0.91	0.91	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{43}	0.89	0.44	2.18	0.91	0.91	0.54	0.64	0.91	0.65	0.43	0.41	0.48
V_{44}	0.82	0.31	2.08	0.91	0.91	0.53	0.63	0.91	0.64	0.42	0.40	0.48
V_{45}	0.96	0.49	2.48	0.90	0.91	0.52	0.64	0.91	0.64	0.42	0.40	0.47
V_{46}	0.96	0.50	2.38	0.90	0.92	0.53	0.64	0.91	0.64	0.42	0.40	0.48
V_{47}	0.96	0.48	2.58	0.90	0.92	0.53	0.65	0.91	0.64	0.42	0.40	0.47
V_{48}	0.95	0.52	2.45	0.90	0.92	0.53	0.64	0.91	0.64	0.42	0.40	0.47

A. DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work, the authors used ChatGPT and DeepL Write in order to assist with language refinement and clarity. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

REFERENCES

[1] Z. Sun, "A short survey of viewing large language models in legal aspect," 2023, *arXiv:2303.09136*.

[2] W. X. Zhao et al., "A survey of large language models," 2023, *arXiv:2303.18223*.

[3] S. Wein and P. Briggs, "A fully automated approach to requirement extraction from design documents," in *Proc. IEEE Aerosp. Conf. (0)*, Mar. 2021, pp. 1–7.

[4] Z. Shakeri Hossein Abad, V. Gervasi, D. Zowghi, and K. Barker, "ELICA: An automated tool for dynamic extraction of requirements relevant information," in *Proc. 5th Int. Workshop Artif. Intell. Requirements Eng. (AIRE)*, Aug. 2018, pp. 8–14.
[5] S. Abualhaija, M. Ceci, N. Sannier, D. Bianculli, L. C. Briand, D. Zetzsche, and M. Bodellini, "AI-enabled regulatory change analysis of legal requirements," in *Proc. IEEE 32nd Int. Requirements Eng. Conf. (RE)*, Jun. 2024, pp. 5–17.
[6] A. Grattafiori et al., "The llama 3 herd of models," 2024, *arXiv:2407.21783*.
[7] C. Jeong, "Fine-tuning and utilization methods of domain-specific LLMs," 2024, *arXiv:2401.02981*.
[8] Z. Chu, H. Guo, X. Zhou, Y. Wang, F. Yu, H. Chen, W. Xu, X. Lu, Q. Cui, L. Li, J. Zhou, and S. Li, "Data-centric financial large language models," 2023, *arXiv:2310.17784*.
[9] Y. Li, Z. Li, K. Zhang, R. Dan, S. Jiang, and Y. Zhang, "ChatDoctor: A medical chat model fine-tuned on a large language model meta-AI (LLaMA) using medical domain knowledge," *Cureus*, vol. 15, no. 6, 2023, Art. no. e40895.
[10] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, "BloombergGPT: A large language model for finance," 2023, *arXiv:2303.17564*.
[11] S. Goyal, E. Rastogi, S. P. Rajagopal, D. Yuan, F. Zhao, J. Chintagunta, G. Naik, and J. Ward, "HealAI: A healthcare LLM for effective medical documentation," in *Proc. 17th ACM Int. Conf. Web Search Data Mining*, Mar. 2024, pp. 1167–1168.
[12] S. Saxena, S. Prasad, M. I, A. Shankar, V. V, S. Gopalakrishnan, and V. Vaddina, "Automated tailoring of large language models for industry-specific downstream tasks," in *Proc. 17th ACM Int. Conf. Web Search Data Mining*, Mar. 2024, pp. 1184–1185.
[13] A. Vogelsang and J. Fischbach, "Using large language models for natural language processing tasks in requirements engineering: A systematic guideline," 2024, *arXiv:2402.13823*.
[14] Z. Allen-Zhu. (Jul. 2024). *ICML 2024 Tutorial: Physics of Language Models*. [Online]. Available: https://physics.allen-zhu.com/
[15] Z. Allen-Zhu and Y. Li, "Physics of language models: Part 3.1, knowledge storage and extraction," 2023, *arXiv:2309.14316*.
[16] J. Jakubik, M. Vössing, N. Kühl, J. Walk, and G. Satzger, "Data-centric artificial intelligence," *Bus. & Inf. Syst. Eng.*, vol. 66, no. 4, pp. 507–515, 2024.
[17] C. Arora, J. Grundy, and M. Abdelrazek, "Advancing requirements engineering through generative AI: Assessing the role of LLMs," in *Generative AI for Effective Software Development*. Cham, Switzerland: Springer, 2024, pp. 129–148.
[18] N. Marques, R. R. Silva, and J. Bernardino, "Using ChatGPT in software requirements engineering: A comprehensive review," *Future Internet*, vol. 16, no. 6, p. 180, May 2024.
[19] A. E. Gärtner and D. Göhlich, "Automated requirement contradiction detection through formal logic and LLMs," *Automated Softw. Eng.*, vol. 31, no. 2, p. 49, Nov. 2024.
[20] S. Lubos, A. Felfernig, T. N. T. Tran, D. Garber, M. El Mansi, S. P. Erdeniz, and V.-M. Le, "Leveraging LLMs for the quality assurance of software requirements," in *Proc. IEEE 32nd Int. Requirements Eng. Conf. (RE)*, Jun. 2024, pp. 389–397.
[21] T. Schick and H. Schütze, "It's not just size that matters: Small language models are also few-shot learners," 2020, *arXiv:2009.07118*.
[22] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. Ka-Wei Lee, "LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models," 2023, *arXiv:2304.01933*.
[23] X. Xu, Z. Wu, R. Qiao, A. Verma, Y. Shu, J. Wang, X. Niu, Z. He, J. Chen, Z. Zhou, G. K. R. Lau, H. Dao, L. Agussurja, R. Hwee L. Sim, X. Lin, W. Hu, Z. Dai, P. W. Koh, and B. K. H. Low, "Data-centric AI in the age of large language models," 2024, *arXiv:2406.14473*.
[24] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu, "Data-centric artificial intelligence: A survey," 2023, *arXiv:2303.10158*.
[25] L. F. A. O. Pellicer, T. M. Ferreira, and A. H. R. Costa, "Data augmentation techniques in natural language processing," *Appl. Soft Comput.*, vol. 132, Jan. 2022, Art. no. 109803.
[26] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, "A survey of data augmentation approaches for NLP," 2021, *arXiv:2105.03075*.
[27] B. Li, Y. Hou, and W. Che, "Data augmentation approaches in natural language processing: A survey," *AI Open*, vol. 3, pp. 71–90, Jan. 2022.
[28] B. Ding, C. Qin, R. Zhao, T. Luo, X. Li, G. Chen, W. Xia, J. Hu, A. Tuan Luu, and S. Joty, "Data augmentation using large language models: Data perspectives, learning paradigms and challenges," 2024, *arXiv:2403.02990*.
[29] Y. Li, K. Ding, J. Wang, and K. Lee, "Empowering large language models for textual data augmentation," 2024, *arXiv:2404.17642*.
[30] Z. Li, L. Si, C. Guo, Y. Yang, and Q. Cao, "Data augmentation for text-based person retrieval using large language models," 2024, *arXiv:2405.11971*.
[31] J. Cegin, J. Simko, and P. Brusilovsky, "LLMs vs established text augmentation techniques for classification: When do the benefits outweight the costs?" 2024, *arXiv:2408.16502*.
[32] R. Gan, Z. Wu, R. Sun, J. Lu, X. Wu, D. Zhang, K. Pan, J. He, Y. Tian, P. Yang, Q. Yang, H. Wang, J. Zhang, and Y. Song, "Ziya2: Data-centric learning is all LLMs need," 2023, *arXiv:2311.03301*.
[33] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–35, Sep. 2023.
[34] K. Ronanki, B. Cabrero-Daniel, J. Horkoff, and C. Berger, "Requirements engineering using generative AI: Prompts and prompting patterns," in *Generative AI for Effective Software Development*. Cham, Switzerland: Springer, 2024, pp. 109–127.
[35] T. Heston and C. Khun, "Prompt engineering in medical education," *Int. Med. Educ.*, vol. 2, no. 3, pp. 198–205, Aug. 2023.
[36] B. Meskó, "Prompt engineering as an important emerging skill for medical professionals: Tutorial," *J. Med. Internet Res.*, vol. 25, Oct. 2023, Art. no. e50638.
[37] L. Giray, "Prompt engineering with ChatGPT: A guide for academic writers," *Ann. Biomed. Eng.*, vol. 51, no. 12, pp. 2629–2633, Dec. 2023.
[38] H. Dang, L. Mecke, F. Lehmann, S. Goller, and D. Buschek, "How to prompt? Opportunities and challenges of zero- and few-shot learning for human-AI interaction in creative applications of generative models," 2022, *arXiv:2209.01390*.
[39] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," 2018, *arXiv:1804.07461*.
[40] M. Gao, X. Hu, J. Ruan, X. Pu, and X. Wan, "LLM-based NLG evaluation: Current status and challenges," 2024, *arXiv:2402.01383*.
[41] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 311–318.
[42] INCOSE, *INCOSE Systems Engineering Handbook*. Hoboken, NJ, USA: Wiley, 2023.
[43] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "PURE: A dataset of public requirements documents," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 502–505.
[44] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," 2021, *arXiv:2106.09685*.
[45] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. Text summarization branches out*, 2004, pp. 74–81.
[46] M. Post, "A call for clarity in reporting BLEU scores," 2018, *arXiv:1804.08771*.
[47] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proc. ACL Workshop Intrinsic Extrinsic Eval. Measures Mach. Transl. Summarization*, 2005, pp. 65–72.
[48] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," 2019, *arXiv:1904.09675*.
[49] F. Alwahedi, A. Aldhaheri, M. A. Ferrag, A. Battah, and N. Tihanyi, "Machine learning techniques for IoT security: Current research and future vision with generative AI and large language models," *Internet Things Cyber-Phys. Syst.*, vol. 4, pp. 167–185, Jan. 2024.

LEOPOLD MÜLLER is currently pursuing the Ph.D. degree with the Chair of Information Systems and Human-Centric AI, University of Bayreuth. He is a Research Assistant with the Branch of Business and Information Systems Engineering, Fraunhofer FIT. In his research, he focuses on developing LLM solutions for real-world applications.

NINA SCHWARZ is currently pursuing the Ph.D. degree with the Chair of Information Systems and Human-Centric AI, University of Bayreuth. She is a Research Assistant with the Branch of Business and Information Systems Engineering, Fraunhofer FIT. In her research, she focuses on big data analytics, machine learning, and the application of artificial intelligence.

LARS BÖCKING is currently pursuing the Ph.D. degree with the Chair of Information Systems and Human-Centric AI, University of Bayreuth. He is a Research Assistant with the Branch of Business and Information Systems Engineering, Fraunhofer FIT. In his research, he focuses on data-centric AI, uncertainty quantification, algorithm selection, and time series analysis.

ANDREAS BERECZUK is a Sub-Project Leader in requirements engineering with TenneT TSO GmbH. His focus is on the improvement and development of technical requirements and the digitalization of requirements management. His professional career started with the University of Regensburg, as a Ph.D. Student in theoretical physics on the topic of correlations in mesoscopic quantum systems.

HANNO STAGGE received the Ph.D. degree in power systems from TU Clausthal. He is the Program Manager of systems engineering with TenneT TSO GmbH. He is working on requirements engineering (RE) and requirements management (RM) and the linkage of RE/RM to digitalization topics. Previously, he was a Chief Engineer with the Institute for Power Electronics and Electrical Drives, RWTH Aachen University. He was active in more than 30 publications in various formats. He has been a member of different working groups in organizations, such as CIGRE and VDE Germany.

WOLFGANG KRATSCH received the Ph.D. degree from the University of Bayreuth, in 2020. He is a Research Professor for applied AI with the Technische Hochschule Augsburg and the Director with the FIM Research Institute, with roles at Fraunhofer FIT and the Center for Process Intelligence. His research focuses on data-driven process management, AI applications in process optimization, and data quality in event logs. He co-founded the AI startup credium covering geospatial data analytics. He has published extensively, collaborates with several industry partners, and is still active in the startup scene.

NIKLAS KÜHL received the Ph.D. degree in information systems and the Habilitation degree in applied computer science from Karlsruhe Institute of Technology. He is a Full Professor of information systems and human-centric AI with the University of Bayreuth, with roles at Fraunhofer FIT, FIM Research Institute, and IBM. Previously, he led AI projects with IBM. He has published over 130 peer-reviewed articles and collaborates with global institutions, such as CMU, UT Austin, and MIT-IBM Watson AI Laboratory. His research spans machine learning, human-AI collaboration, fairness, and appropriate reliance.