POTracker: Optimizing Large Language Models for Standard-Compliant Power Outage Report Generation

Anonymous ACL submission

Abstract

Recent large language models (LLMs) are good at general text generation, but it is still hard to use them for domain-specific data generation because the output must follow strict format and structure rules. Different from open-ended tasks like question answering or translation, domain-specific generation must be both correct in meaning and also follow existing guidelines and standards. In this work, we study the nationwide interoperability problem of utility power outage reports in the United States. In practice, outage reports need to be machine-readable (e.g., JSON or XML) and must strictly follow requirements from energy-sector regulatory bodies. To handle this, we propose POTracker, an optimized LLM for power outage report generation. We fine-tune Qwen2.5-7B-Instruct with our proposed objective. The key contribution is a new loss function, $POTracker_{Loss}$, that considers both textual similarity and structure (tag) similarity between the generated report and the ground-truth report. We evaluate POTracker on a dataset of 1,000 power outage reports and compare it with five well-known fine-tuning methods and one rule-based XML conversion method. Results show that POTracker performs better than other fine-tuning approaches, improving overall accuracy by up to 51% and reaching 86.47% structural accuracy for generated power outage reports. In addition, we conduct a human study to assess the quality of the ground-truth standard reports, where domain experts assign the generated labels an average score of **4.03** on a 0–5 scale.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across a variety of natural language processing (NLP) tasks, including summarization (Zhang et al., 2023), question answering (Kamalloo et al., 2023), and machine translation (Zhu et al., 2024). These models are increasingly being adopted in real-world applications due to their generalization abilities and fluent text generation. However, applying LLMs to domain-specific generation tasks remains a significant challenge (Ling et al., 2024). Unlike general-purpose applications, many domain-specific tasks require the output to be not only semantically accurate but also to strictly follow formal conventions defined by industry or institutional standards.

We identify two major challenges in applying Large Language Models (LLMs) to solve domain-specific problems. First, as the term suggests, these problems require the integration of knowledge from specialized resources at the industry or institutional level. This makes it difficult for general-purpose LLMs to provide high-quality answers, especially in terms of semantic accuracy. For instance, (Phan et al., 2024) shows that in the domain of environmental assessment, state-of-the-art models such as GPT-4 (OpenAI et al., 2024) and Claude (Anthropic, 2024) achieved low accuracy when answering open-ended questions sourced from domain experts. Second, the outputs users expect from LLMs are not limited to textual responses but often extend to other data formats, depending on specific requirements. For example, (Hollmann et al., 2023) optimized LLMs for feature generation in machine learning tasks. Other studies have focused on enhancing LLMs for machine learning interpretability (Bordt et al., 2024), code generation (Nascimento et al., 2024), and code performance optimization (Mahmud et al., 2025). Compared to general text generation tasks, these applications demand not only semantic accuracy but also syntactic validity. Consequently, a second challenge arises: the need for more stringent evaluation policies to assess output quality under domain-specific and format-specific constraints.

In this project, our domain for conducting a power outage report generation is the Outage Data Initiative Nationwide (ODIN) benchmark (Labo-

ratory and of Energy Office of Electricity, 2023), provided by the Oak Ridge National Laboratory (ORNL). Its primary goal is to establish a comprehensive digital reporting standard for power outage data, enabling utilities and stakeholders to exchange information seamlessly. This initiative aims to enhance restoration efforts, reliability, risk mitigation, and emergency response by providing standardized and real-time outage information. A central feature of ODIN is its delivery of real-time power outage data at the county level. This information is accessible via the Open Energy Data Portal, providing stakeholders with up-to-date insights into outages across various regions. By leveraging this data, emergency responders, utility providers, and other key entities can enhance their situational awareness and better coordinate their responses to power disruptions. Each data point provided in the ODIN dataset needs to follow standard of Common Information Model (CIM) IEC 61968-3 (International Electrotechnical Commission, 2021). Various defined templates of ODIN data points can be found in the ODIN documentation (Laboratory, 2023). Users face several challenges when providing input to the ODIN system. Often, the input is submitted in unstructured or free-form formats, including non-standard XML and JSON files, as well as free-text descriptions. This lack of standardization complicates data processing and integration.

To overcome this challenge, we address the following important task in the context of power outage reporting: Power Outage Report Generation. This task involves transforming loosely structured or noisy XML documents, collected from users, into canonical representations as defined by ORNL documentation. In this work, we consider the well-known Close model, GPT-5.2 (OpenAI, 2025), for generating labeled power outage reports. We propose $POTracker_{metric}$ score for meassuring the quality of the generated report given a synthetic report as label by GPT-5.2 with respoect to both context and tag similarities. By experiment, we show whether or not our proposed $POTracker_{Loss}$ can significantly improve Qwen-2.5-7b-Instruct in power outage reports' generation against multiple baselines, including a rule-based approach and other fine-tuning optimization approaches. We propose following contribution. Supplemental material is provided at here[1].

1. **Metric.** We design $POTracker_{Metric}$, a formula for measuring the similarity between generated and predicted power outage reports with the consideration of both textual and structural correctness.

2. **Custom-Loss Fine Tuning**. We propose $POTracker_{Loss}$, a specific loss function with the integration to the traditional cross entropy loss function to allow LLMs better generating power outage report

3. **Evaluation**. We implemented multiple strategies for power outage report generation as baselines for comparison with POTracker in power outage report generation. We show that POTracker can outperform baselines in terms of accuracy and performance.

Next sections are provided as following. The Motivation Example and Background sections provide examples and important concepts/ design selection of our work. The next section, Approach, introduces our proposed fine-tuning with our designed custom loss function, along with alternative fine-tuning strategies we use as the baselines. Experiment introduces our evaluation's configuration and results. Final sections are Related Work, Discussion and Conclusion.

2 Motivation Example

An example of the power outage generation process is shown in Figure 1. In this scenario, the input—provided by data collectors—contains a list of electric outage reports by region. For each region, various types of information may be included, such as the number of affected customers, the outage timeline, and a status message. It is important to note that outage reports may differ in the properties they contain. For instance, some reports include a `relatedEventId` field that links to other related outage events. The output, formatted according to ODIN templates, adheres to a more restrictive and unified schema for all electric outage reports. To transform nonstandard inputs into this standardized format, domain experts are often required to manually map properties and values to the target template specified in the ODIN documentation (Laboratory, 2023).

3 Background

In this section, we describe key concepts used throughout the paper.

[1] https://tinyurl.com/3tat5sy5

Input as non-standard Power Outage Report


```
{
  "electricOutages": [
    {
      "id": "250011343",
      "customerCount": "1",
      "x": "2592812.726003",
      "y": "567010.25121",
      "reportedDateTime": "2025-03-07T13:04:36.000+00:00",
      "estimatedRestoreTime": "2025-03-07T16:30:00.000+00:00",
      "estimatedRepairTime": "2025-03-07T16:30:00.000+00:00",
      "ertRangeWidth": 0,
      "relatedEventId": null,
      "ertScenario": 2,
      "statusMessage": "We are aware of an outage in your area affecting 1 customer...",
      "outageMapMessage": "We are aware of an outage in your area affecting 1 customer...",
      "ertProvided": true
    },
    ...
  }
```


Output as Standard Power Outage Report in CIMIEC 61968-3 standard


```
<?xml version="1.0" encoding="UTF-8"?>
<po:PubOutages xmlns:po="http://iec.ch/TC57/2014/PubOutages#">
  <po:Outage>
    <po:mRID>250011343</po:mRID>
    <po:communityDescriptor>37920</po:communityDescriptor>
    <po:customerCount>1</po:customerCount>
    <po:cause>Pending Investigation</po:cause>
    <po:causeKind>unplanned</po:causeKind>
    <po:outageKind>outageReported</po:outageKind>
    <po:reportedStartTime>2025-03-07T13:04:36.000+00:00</po:reportedStartTime>
    <po:statusKind>assigned</po:statusKind>
    <po:actualPeriod>
      <po:start>2025-03-07T13:04:36.000+00:00</po:start>
    </po:actualPeriod>
    <po:EstimatedRestorationTime>
      <po:ert>2025-03-07T16:30:00.000+00:00</po:ert>
    </po:EstimatedRestorationTime>
    <po:Incident>
      <po:Location>
        <po:geoInfoReference>37920</po:geoInfoReference>
        <po:zoneKind>zip</po:zoneKind>
      </po:Location>
    </po:Incident>
    <po:Names>
      <po:name>UTIL002</po:name>
      <po:nameType>UtilityID</po:nameType>
      <po:nameTypeAuthority>U.S. Energy Information Administration</po:nameTypeAuthority>
    </po:Names>
  </po:Outage>
</po:PubOutages>
```


Figure 1: Examples of inputs and output of POTracker Standardization and Generation .

Power Outage Tracker. In the context of this paper, the Power Outage Tracker refers to an application provided by ORNL through the ODIN system. Given inputs such as outage reports submitted by individuals across the United States, the application generates a standardized outage report following the CIM-IEC-61968-3 specification (International Electrotechnical Commission, 2021). This tool allows developers to submit various forms of input, including incomplete XML or JSON reports, as well as natural language descriptions. The application is integrated into the chatbot interface of the ODIN live map, accessible at this location[2].

Selected Dataset. We use the dataset of **1000** entities of power outage reports in USA collected from March, 2024. These entities are provided in a non-standard format. We split **800** entities for training and **100** for validation during fine-tuning. The other 100 entities was used for testing the performance of models.

Selected LLMs. For standard reports as labeled data, we use GPT 5.2 (OpenAI, 2025) for generating a standard power outage report following the CIM-IEC-61968-3 standard. We build our pipeline as a power outage report prediction application based on the fine-tuning process of the well-known and compact LLM Qwen-2.5-7b-Instruct[3].

4 Approach

In this section, we describe the modules provided for the POTracker application, shown in Figure 2.

4.1 Data Preprocessing

In this module, the input data is preprocessed to extract meaningful information related to power outages. ODIN's data collection process is designed to ingest outage reports from a wide range of sources, including utility companies, emergency response agencies, and even the general public. These inputs reflect real-time or near-real-time conditions across different regions in the United States. Because contributors operate at different technical levels, the incoming data varies widely in format—ranging from well-structured XML and JSON files to unstructured natural language descriptions submitted via forms or chat interfaces. Utility companies often send data directly from their Outage Management Systems (OMS) or Geographic Information Systems (GIS), while emergency responders may relay information manually or through partner platforms. Public contributions may include citizen reports submitted through local portals or integrated communication channels. ODIN accommodates this heterogeneity using a flexible interface that supports both automated API submissions and manual uploads. Each report, regardless of format, is parsed and preprocessed to extract key metadata such as location identifiers, outage start time, number of affected customers, and restoration estimates. The data is then queued for validation and transformed into a standardized schema. We refer to this level of raw input data as **nonstandard (power outage) reports**.

4.2 Data Label Generation

One significant challenge is that, while there are many datasets containing users' power outage reports, there is no readily available dataset of expected outputs in the CIM-IEC-61968-3 standard format as defined by the ODIN documentation (International Electrotechnical Commission, 2021). Although these standardized datasets—referred to as *standard reports*—are critical for guiding LLM optimization, generating them manually is resource-intensive due to the dynamic nature of outage reports and the sheer volume of user-submitted

[2] https://odin.ornl.gov/outagemap/index.html

[3] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

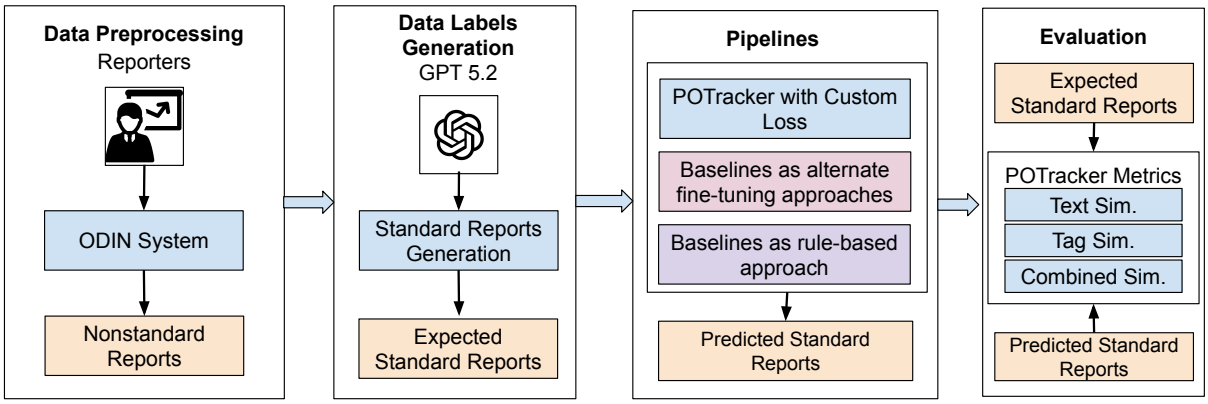

Figure 2: Architecture Overview of POTracker.

data each year. To address this, we employ closed LLMs guided by tailored prompts to generate large-scale standard reports from nonstandard inputs. Closed LLMs, which require access fees and API tokens, are known for their high-quality generative capabilities. In our project, We prompt GPT-5.2 to convert nonstandard reports into standard reports based on the requirements outlined in the ODIN documentation. The resulting outputs serve as candidate labels for further validation. We provide the prompt of data label generation in the replication package of this project.

4.3 Evaluation.

The target and predicted objects for our problem don't only represent textual content but also structured content. We focus on the problem of comparing standard XMLs. To the best of our knowledge, there is no prior work provided a module for evaluating the quality of XML files generated by LLMs, since LLM-based XML generation systems are mainly at idealizing phase (Zah, 2024). We design $POTracker_{Metric}$, a metric for calculating the similarities between XML files as predicted and expected results. Overall, the $POTracker_{Metric}$ function is designed as a combination formula of the following aspects.

Text Similarity. The first key aspect of XML comparison is *text similarity*, which evaluates how closely the textual values of corresponding XML nodes match between two documents. This is especially useful when validating whether the transformed or generated document preserves the intended semantics from the original. Text similarity is critical in high-stakes domains like outage reporting, where changes to specific fields—such as customer counts or timestamps—could lead to misinformation or operational errors. Accurately capturing these content differences ensures that outputs generated by models or pipelines maintain their factual integrity across revisions. We measure the Text Similarity score between two given reports using cosine similarity (Manning et al., 2008).

Tag Structure Similarity. Another fundamental dimension is *tag structure similarity*, which measures how well the XML structure aligns between two documents. This includes the hierarchical arrangement of elements, the presence of required attributes, and the ordering of nodes. This is especially relevant for schema validation, where even small structural inconsistencies—such as missing wrapper elements or reordered tags—can cause downstream parsing or validation to fail. Tag structure comparison is therefore important for ensuring interoperability between systems that rely on standards such as CIM-IEC-61968-3. To quantify tag similarity between standard-format power outage reports, we use the sequence-matching approach implemented in Python's `difflib.SequenceMatcher` library, which identifies the longest contiguous matching blocks between two sequences and returns a normalized similarity score in the range $[0, 1]$ (Python Software Foundation).

The $POTracker_{Metic}$ function is calculated as the combination of Content Similarity and Tag Structure Similarity by following:

$$POTracker_Metric = \alpha \cdot \text{Text_Similarity} + (1 - \alpha) \cdot \text{Tag_Similarity} \quad (1)$$

In formula 1, $\alpha \in [0, 1]$ is a weighting factor that reflects the importance of structural integrity

Algorithm 1 POTracker Custom Loss Function

Require: logits Z, labels Y (prompt masked with -100), similarity metric $\mathrm{Sim}(\cdot,\cdot)$, weight λ

Ensure: $\mathcal{L}_{\text{total}}$

1: Compute token-level cross-entropy on response tokens: $\mathcal{L}_{\text{CE}} \leftarrow \mathrm{CE}(Z, Y)$
2: Greedily decode a predicted response $\hat{y}$ from Z and decode reference response y from Y
3: Compute sequence-level quality score $s \leftarrow \mathrm{Sim}(y, \hat{y})$ (text + XML/tag structure)
4: Convert to penalty $p \leftarrow 1 - s$
5: Return total loss: $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{CE}} + \lambda\, p$

relative to content accuracy. By default, we set the $\alpha = 0.5$.

4.4 Pipelines

In the era of large language models (LLMs), a key challenge is adapting a general-purpose model to new knowledge and domain-specific requirements. A common solution is fine-tuning, where the model is trained on paired examples—each sample includes a prompt and an expected response in the required format—so it learns to follow task instructions and generate compliant outputs (Wei et al., 2022). In practice, many modern pipelines use parameter-efficient fine-tuning (PEFT), which keeps the base model fixed and learns lightweight task-specific modules (e.g., adapters or low-rank updates). This approach produces small adapter weights that can be attached to the original LLM to steer domain-specific generation (Houlsby et al., 2019; Hu et al., 2021). In our work, in addition to our proposed pipelines, we evaluate two baseline branches. First, we implement several established fine-tuning variants and use their learned adapters to assess the quality of power report generation. Second, we implement a simpler rule-based conversion method for power report generation and compare its performance against learning-based approaches. We show baselines' algorithms in the Appendix.

4.4.1 POTracker with Custom Loss Function.

We design our proposed custom loss function in Algorithm 1. Given the fact that cross-entropy loss is frequently used in many fine-tuning pipelines, we train the model with standard token-level cross-entropy so it can learns to predict the correct XML tokens. To enhance the ability of fine-tuning in generating good power outage reports in terms of XML format we also add a sequence-level quality signal that measures whether the entire generated XML looks correct (Line 5). For each mini-batch, we form a greedy predicted XML (from parameter logits Z, decode it and the gold XML into text, and compute a single similarity score that reflects both surface text match and XML tag/structure consistency. This score is then converted into a penalty and added to the cross-entropy objective with a weighting factor λ. In our setting, we set this weight factor as $\lambda = 0.5$.

4.4.2 Baselines as Alternate Fine-tuning approaches.

Token Weight Cross Entropy (TW-CE). The idea of TW-CE appeared original in dense object detection (Lin et al., 2018). In this work, they provide a weight for each category to measure the cross-entropy function. The intuition of this weight assignment is that some categories have higher chance to be selected compared to other category in object detection. We borrow this idea to implement the baseline with a token weight applied to the original cross-entropy.

We implement the cross-entropy loss by reweighting individual target tokens according to their structural importance in the power outage report. The first step of TW-CE created a fixed vocabulary-level weight vector by decoding each token and assigning higher weights to tokens indicative of XML structure (e.g., tag markers such as < and >), while leaving ordinary content tokens at uniform weight. During training, it computes the match between tags with higher score than the match between normal tokens. This preserves the efficiency and stability of CE while prioritizing learning on XML-critical tokens.

Tag Masking Cross Entropy (TM-CE). The idea of Tag Masking for cross-entropy in fine-tuning models has come from the BERT-based model (Devlin et al., 2019a; Joshi et al., 2020). In the BERT model, given input as a sequence of tokens, a set of tokens was replaced by a masking token. During the training process, the masking tokens will be predicted, thus improving the quality of models for text generation. SpanBERT (Joshi et al., 2020) also relied on masking technique, but this work attempted to prioritize masking spans of text instead of masking discontinuous tokens. Usually, the masked tokens/ spans were considered as important content inside the input for training models.

For power outage report generation, we implement a module that injects XML-structure awareness into standard causal language modeling by requiring the model to identify which response tokens belong to XML tags (i.e., substrings matching <. . . > within the response region) and upweighting their contribution to the cross-entropy in fine-tuning. Our implementation of tag-masking CE concatenates the prompt and response into a single sequence, locates the response span using a delimiter, detects all tag character spans via a regular expression, and uses a fast tokenizer's offset mapping to project these character spans onto token indices to construct a binary `tag_mask`. The training labels follow the usual instruction-tuning convention where prompt tokens are ignored and only response tokens contribute to the loss. During optimization, the trainer computes per-token cross-entropy under the standard causal shift and applies a per-token weight of w_{tag} to tokens with `tag_mask`=1 and w_{text} otherwise; an optional normalization rescales the weights so that the overall loss magnitude remains comparable to vanilla CE, avoiding unintended learning-rate effects. This is closely related in spirit to objectives that concentrate supervision on selected tokens or spans while keeping a CE-style optimization pipeline (Devlin et al., 2019a; Joshi et al., 2020).

Example Weight Cross Entropy with PO Custom Loss (EW-CE-PO). This pipeline was implemented similarly to the Token Weight Cross Entropy, but the weight was calculated per sample instead of per token for cross-entropy calculation.This implementation was inspired from the success of reweighting examples for deep learning applications (Ren et al., 2019; Shu et al., 2019). We assign the weight as POTracker combined metric between labels and predicted results for reweighting example for cross entropy calculation.

Direct Preference Optimization (DPO). Preference optimization (Fürnkranz and Hüllermeier, 2010) is a technique aimed at aligning language models with human preferences by training them on datasets where outputs are ranked based on desirability.

In our implementation, there is one challenge that is different from other fine-tuning approaches: Direct Preference Optimization requires the training and validation set to have examples of bad responses besides good responses. From power outage reports in nonstandard and standard formats, we construct responses as follows. Given each gold XML response, the code generates a negative example (`bad_response`) by attempting to parse the XML and applying one random corruption strategy: swap text between two leaf nodes; replace a leaf text with an implausible value; remove a random element, or (iv) perturb numeric leaf values. If parsing fails or the corrupted output is too similar/too short,a fallback routine truncates the XML and injects broken/incorrect tags.

4.4.3 Baseslines as Rule-based Approaches.

Besides LLM-based approaches, an ad-hoc but simpler approach for XML generation can be using a rule-based mechanism for power outage report generation. The main idea of this approach is provided as follows. First, we collect a set of rules for converting non-standard user input report tags to standard output tags. Next, we implement a program for traversing the input code, and heuristically convert each input tag to a standard output tag. The program can leave as it visits a tag that hasn't appeared in the vocabulary of rules we defined. In fact, this rule-based approach is inspired from research works on XML conversion in other languages or domains (Kudrass and Krumbein, 2003; Berdaguer et al., 2007).

One challenge of conducting rules for power outage tracker generation is that they need to be collected from experts. To achieve that set of rules, we hire two data scientists who have worked with energy sector datasets for more than 2 years. They manually look at 100 samples of non-standard and standard outage reports and write rules. Next, they discussed, among other about, the validity of each rule. The rules agreed upon by both were included in the implementation. The subset of rules is shown in the Appendix.

5 Experiment

In our experiments, we evaluate power outage report generation across multiple fine-tuning strategies and a rule-based conversion baseline. Next, we describe the evaluation setup.

5.1 Datasets Preparation

We collect the power outage reports from Janurary, 2024 to March, 2025. We select 1000 data points for standard report generation using GPT-5.2. In this set, we randomly select 100 data points for evaluation. We manually verified this evaluation set to see if it has meaningful report information. For the remaining 900 data points we use them as

the training data for the fine-tuning algorithms. In total, we use 900 data points for LLM Optimization techniques and use 100 data points for evaluation. We create a compact prompt with the integration of an example of standard power outage reports as context to help GPT-5.2 generate results with high quality. We perform multiple queries using the OpenAI key and the OpenRouter interface to achieve this dataset.

5.2 Settings of Open LLM

Qwen-2.5-7B-Instruct provides a strong accuracy–efficiency trade-off for domain assistants that must follow instructions and emit schema-constrained outputs. Despite its compact size, the model is explicitly optimized for instruction following and structured generation—capabilities that transfer well to XML-style report synthesis (Qwen Team, 2024b; Yang et al., 2025). It also supports long-context inference (up to 128K tokens) and long-form generation (up to 8K tokens), which is beneficial when prompts include extensive context, guidelines, or multi-record outage narratives (Qwen Team, 2024b). In addition, Qwen2.5 models (including the 7B variant) are released as open weights under a permissive license (Apache 2.0 for all but specific excluded sizes), enabling reproducible research and practical deployment in real applications (Qwen Team, 2024a).

Given Qwen-2.5-7B-Instruct as our model for implementing strategies of fine-tuning approaches, we use popular hyperparameter settings for this model. It uses 28 Transformer layers with grouped-query attention (28 query heads and 4 key/value heads), Rotary Positional Embedding (RoPE) positional encoding, and Root Mean Square Layer Normalization (RMSNorm), and is released with bfloat16 weights. For text generation, we use the released default sampling setup enables stochastic decoding (`do_sample=true`) with moderate temperature and nucleus/top-k filtering, alongside a mild repetition penalty. Important setting of Qwen-2.5-7B-Instruct can be shown in the Appendix.

Prompt for answer generation. For all configurations except label generation, we do not include in-context examples of standard power outage reports. This design isolates the effect of fine-tuning by encouraging the model to learn the report format and domain conventions from the training and validation data, rather than relying on few-shot prompting or contextual imitation during inference.

Original Baseline. In the experiment, we add an extra configuration that generates a power outage report with the original selected LLM. For prompt preparation with this setting, we provide a hint for the report generation process that the output must follow the CIM-IEC-61968-3 format.

Table 1: Performance comparison across training objectives and baselines.

Configuration	Overall	Text Acc/	Tag Acc.
Qwen2.5-7B-Instruct	16.20%	28.85%	3.56%
TW-CE	15.93%	28.39%	3.48%
TM-CE	16.13%	28.76%	3.50%
EW-CE-PO	15.87%	28.25%	3.49%
DPO	16.16%	28.89%	3.43%
Rule-based Conversion	61.46%	43.91%	79.00%
POTracker Loss	69.14%	51.81%	86.47%

5.3 Hardware Setting

We ran all training and evaluation on a dedicated single-node server equipped with `1×NVIDIA A100` GPU (`40`,GB VRAM). The host system uses an `Intel Core i9` CPU with `16` physical cores and `100`,GB RAM, backed by a `1`,TB high-throughput NVMe SSD. Experiments were executed on `Ubuntu 24.04` with CUDA `12.0`.

5.4 Evaluation Metrics

We evaluate the quality of power report generation by the combined accuracy, text accuracy, and tag accuracy measured by $POTracker_{Metric}$. Noted, the text accuracy score was calculated when the weight of $POTracker_{Metric}$ was 1, and the tag accuracy score was calculated when the weight as 0 (see Formula 1).

5.5 Results

Original LLM limitations. Table 1 indicates that the baseline *Qwen2.5-7B-Instruct* performs poorly for outage-report XML generation (Overall 16.20%, Tag Accuracy 3.56%). This result is consistent with the intuition that the target outputs are highly *domain- and schema-specific*: generating standard-compliant XML requires strict adherence to hierarchical structure, tag ordering, and attribute constraints. While the baseline model can often produce plausible natural language, it struggles to consistently emit valid, standardized XML without additional task-specific supervision.

Cross-entropy variants do not transfer cleanly. The three cross-entropy fine-tuning variants (Token Weight, Tag Mask, and Example Weight) yield no meaningful gains over the baseline (Over-

all $\approx$ 15.87%–16.13%, Tag Accuracy $\approx$ 3.48%–3.50%). This suggests that standard NLP fine-tuning pipelines require adaptation when applied to power outage tracker report generation, where success depends primarily on *structured correctness* rather than surface-level fluency. Reweighting tokens, masking tags, or upweighting examples may be insufficient when local token likelihood is weakly correlated with structural validity and schema compliance.

Preference optimization depends on negative quality. Direct Preference Optimization (DPO) remains close to the cross-entropy baselines (Overall 16.16%, Tag Accuracy 3.43%), implying that the preference signal may not yet be informative enough. A likely explanation is that negative example generation needs improvement: if negatives are too trivial, too noisy, or not targeted toward *near-miss* structural failures, the model receives limited guidance about which corrections matter. Constructing harder negatives that preserve semantic plausibility while violating schema constraints (e.g., missing wrappers, invalid attributes, swapped tag order) may yield a stronger ranking signal and better alignment.

Rules remain a strong structural prior. Rule-based conversion (Overall 61.46%, Tag Accuracy 79.00%) still outperforms the LLM-only cross-entropy and DPO baselines by a wide margin. This reinforces that deterministic rules excel at enforcing structural constraints even when they lack semantic flexibility.

5.6 Analysis on quality of Ground-Truth Labels

To assess the quality of the generated labels, we recruited two domain experts to manually evaluate 50 samples randomly selected from the POTracker dataset. Each expert was asked to assign a score from 1 to 5, where 1 indicates the lowest quality and 5 indicates the highest quality. The evaluation rubric was designed to measure the meaningfulness and validity of the generated data points. Based on this analysis, the generated labels achieved an average expert rating of **4.03** across the 100 randomly selected data points, with a median score of **5**. We provide the detailed analysis results in the Appendix and supplemental materials.

6 Related Work

Pretrained architectures such as BERT (Devlin et al., 2019b), CodeT5 (Wang et al., 2021), and LLaMA (Touvron et al., 2023) are now widely used for tasks requiring rich contextual representations and fine-grained prediction. Supervised fine-tuning (SFT) is a strong and practical approach for adapting these models to domain-specific settings (Parthasarathy et al., 2024), including low-resource scenarios and cases where training labels are synthetically constructed (Mecklenburg et al., 2024). Beyond SFT, preference optimization techniques—such as Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) and Direct Preference Optimization (DPO) (Rafailov et al., 2023)—have been shown to improve alignment with human preferences in text generation.

7 Limitations

While the POTracker framework demonstrates promising results in structured report generation, several limitations should be acknowledged. First, our study focuses exclusively on *domain-specific problems* related to power outage tracking, and does not generalize to related domains such as weather forecasts. Second, the fine-tuning process relies heavily on *synthetic datasets* generated via prompting and weak supervision, which may introduce alignment artifacts or limit diversity compared to real-world annotated data.

8 Conclusion

We introduce *POTracker*, a domain-specific AI assistant that supports power outage tracking via LLM-based reasoning. POTracker standardizes heterogeneous, free-form outage reports into ODIN-compliant XML (CIM-IEC-61968-3), with a focus on improving the reliability of compact open LLMs for high-precision structured generation. We incorporate multiple adaptation strategies, including supervised fine-tuning and preference-based optimization via Direct Preference Optimization (DPO), and compare against a rule-based conversion baseline. Across our experiments, POTracker improves over the base LLM by up to 53% of overall accuracy.

References

2024. Enhancing xml-based compiler construction with large language models: A novel approach. *Mesopotamian Journal of Big Data*, 2024:23–39.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. https://www.anthropic.com/news/claude-3-family. Accessed: 2025-04-05.

Pablo Berdaguer, Alcino Cunha, Hugo Pacheco, and Joost Visser. 2007. Coupled schema transformation and data conversion for xml and sql. In *Proceedings of the 9th International Conference on Practical Aspects of Declarative Languages*, PADL'07, page 290–304, Berlin, Heidelberg. Springer-Verlag.

Sebastian Bordt, Ben Lengerich, Harsha Nori, and Rich Caruana. 2024. Data science with llms and interpretable models. *Preprint*, arXiv:2402.14474.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186.

Johannes Fürnkranz and Eyke Hüllermeier. 2010. *Preference Learning*. Springer.

Noah Hollmann, Samuel Müller, and Frank Hutter. 2023. Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

International Electrotechnical Commission. 2021. IEC 61968-3:2021 application integration at electric utilities – system interfaces for distribution management – part 3: Interface for network operations. Accessed: 2025-04-06.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Preprint*, arXiv:1907.10529.

Ehsan Kamalloo, Nouha Dziri, Charles L. A. Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. *Preprint*, arXiv:2305.06984.

Thomas Kudrass and Tobias Krumbein. 2003. Rule-based generation of xml dtds from uml class diagrams. In *Advances in Databases and Information Systems*, pages 339–354, Berlin, Heidelberg. Springer Berlin Heidelberg.

Oak Ridge National Laboratory. 2023. ODIN Developer Guide. Accessed: 2025-04-06.

Oak Ridge National Laboratory and U.S. Department of Energy Office of Electricity. 2023. Outage data initiative nationwide (odin). https://odin.ornl.gov/. Accessed: 2025-04-05.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection. *Preprint*, arXiv:1708.02002.

Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, Tianjiao Zhao, Amit Panalkar, Dhagash Mehta, Stefano Pasquali, Wei Cheng, Haoyu Wang, Yanchi Liu, Zhengzhang Chen, Haifeng Chen, and 5 others. 2024. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *Preprint*, arXiv:2305.18703.

Quazi Ishtiaque Mahmud, Ali TehraniJamsaz, Hung Phan, Le Chen, Mihai Capotă, Theodore Willke, Nesreen K. Ahmed, and Ali Jannesari. 2025. Autoparllm: Gnn-guided context generation for zero-shot code parallelization using llms. *Preprint*, arXiv:2310.04047.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Nick Mecklenburg, Yiyou Lin, Xiaoxiao Li, Daniel Holstein, Leonardo Nunes, Sara Malvar, Bruno Silva, Ranveer Chandra, Vijay Aski, Pavan Kumar Reddy Yannam, Tolga Aktas, and Todd Hendry. 2024. Injecting new knowledge into large language models via supervised fine-tuning. *Preprint*, arXiv:2404.00213.

Nathalia Nascimento, Everton Guimaraes, Sai Sanjna Chintakunta, and Santhosh Anitha Boominathan. 2024. Llm4ds: Evaluating large language models for data science code generation. *Preprint*, arXiv:2411.11908.

OpenAI. 2025. Introducing GPT-5.2.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang and 1 others. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. 2024. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *Preprint*, arXiv:2408.13296.

Hung Phan, Anurag Acharya, Rounak Meyur, Sarthak Chaturvedi, Shivam Sharma, Mike Parker, Dan Nally, Ali Jannesari, Karl Pazdernik, Mahantesh Halappanavar, Sai Munikoti, and Sameera Horawalavithana. 2024. Examining long-context large language models for environmental review document comprehension. *Preprint*, arXiv:2407.07321.

Python Software Foundation. difflib — helpers for computing deltas. `https://docs.python.org/3/library/difflib.html`. Accessed: 2026-01-28.

Qwen Team. 2024a. Qwen2.5: A party of foundation models! `https://qwenlm.github.io/blog/qwen2.5/`. Accessed: 2026-01-29.

Qwen Team. 2024b. Qwen/qwen2.5-7b-instruct (model card). `https://huggingface.co/Qwen/Qwen2.5-7B-Instruct`. Accessed: 2026-01-29.

Rafael Rafailov, Yoonho Lee, Archit Sharma, Tengyu Ma, and Stefano Ermon. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2019. Learning to reweight examples for robust deep learning. *Preprint*, arXiv:1803.09050.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. *Preprint*, arXiv:1902.07379.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8696–8708.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations (ICLR)*.

An Yang, Qwen Team, and 1 others. 2025. Qwen2.5 technical report.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2023. Benchmarking large language models for news summarization. *Preprint*, arXiv:2301.13848.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024. Multilingual machine translation with large language models: Empirical results and analysis. *Preprint*, arXiv:2304.04675.

A Appendix

A.1 Algorithm of Token Weight Cross Entropy

Algorithm 2 Token-Weighted Cross Entropy Loss (Causal LM)

Require: Logits $\mathbf{Z} \in \mathbb{R}^{B\times T\times V}$, labels $\mathbf{Y} \in \{-100, 0, \ldots, V-1\}^{B\times T}$
Require: Vocabulary weights $\mathbf{w}^{(v)} \in \mathbb{R}^{V}$, ignore index -100, normalize flag $norm$
Ensure: Scalar loss $\mathcal{L}$

$\mathbf{Z}' \leftarrow \mathbf{Z}_{:,\,1:T-1,\,:}$ {SHIFT LOGITS LEFT}
$\mathbf{Y}' \leftarrow \mathbf{Y}_{:,\,2:T}$ {SHIFT LABELS RIGHT}
$\mathbf{M} \leftarrow \mathbb{I}[\mathbf{Y}' \neq -100]$ {VALID-TOKEN MASK, SHAPE $(B, T-1)$}
$\boldsymbol{\ell} \leftarrow \textsc{CrossEntropyNoReduce}(\mathbf{Z}', \mathbf{Y}', \texttt{ignore} = -100)$ $\{\boldsymbol{\ell} \in \mathbb{R}^{B\times(T-1)}\}$
$\mathbf{S} \leftarrow \mathbf{Y}'$; $\mathbf{S}[\mathbf{S} = -100] \leftarrow 0$ {SAFE GATHER INDICES}
$\mathbf{W} \leftarrow \mathbf{w}^{(v)}[\mathbf{S}] \odot \mathbf{M}$ {TOKEN WEIGHTS PER POSITION}
if $norm$ **then**
  $d \leftarrow \max\left(\sum \mathbf{W},\ 1\right)$
  $c \leftarrow \max\left(\sum \mathbf{M},\ 1\right)$
  $\mathbf{W} \leftarrow \mathbf{W} \cdot \frac{c}{d}$ {KEEP LOSS SCALE CE-LIKE}
end if
$c \leftarrow \max\left(\sum \mathbf{M},\ 1\right)$
$\mathcal{L} \leftarrow \frac{\sum(\boldsymbol{\ell}\odot\mathbf{W})}{c}$
$\mathcal{L}$

A.2 Algorithm of Tag Mark Cross Entropy

Algorithm 3 Tag-Mask Weighted Cross-Entropy (Tag Mask CE)

Require: Model logits $Z \in \mathbb{R}^{B\times T\times V}$, labels $Y \in \mathbb{Z}^{B\times T}$ (with -100 as ignore),
Require: tag mask $M \in \{0,1\}^{B\times T}$, weights w_{tag}, w_{text}, normalize flag $\textsc{Normalize}$
Ensure: Scalar loss L

$Z' \leftarrow Z[:, 0{:}T{-}1, :]$ $\{B \times (T{-}1) \times V\}$
$Y' \leftarrow Y[:, 1{:}T]$ $\{B \times (T{-}1)\}$
$M' \leftarrow M[:, 1{:}T]$ $\{B \times (T{-}1)\}$
$CE \leftarrow \textsc{CrossEntropyNoReduce}(Z', Y', \textsc{ignore} = -100)$ $\{B \times (T{-}1)\}$
$Vmask \leftarrow \mathbb{I}[Y' \neq -100]$ $\{B \times (T{-}1)\}$
$W \leftarrow \big(M' \cdot (w_{\text{tag}} - w_{\text{text}}) + w_{\text{text}}\big) \odot Vmask$
if $\textsc{Normalize}$ **then**
  $\mathit{denom} \leftarrow \max\big(\sum W,\ 1\big)$
  $\mathit{target} \leftarrow \max\big(\sum Vmask,\ 1\big)$
  $W \leftarrow W \cdot (\mathit{target}/\mathit{denom})$
end if
$L \leftarrow \dfrac{\sum(CE \odot W)}{\max(\sum Vmask,\ 1)}$
L

A.3 Algorithm of Example Weight Cross Entropy

A.4 Algorithm for bad response generation for DPO training.

Algorithm 4 Example-Weighted Cross-Entropy Loss (POTracker-style)

Require: logits $L \in \mathbb{R}^{B\times T\times V}$, labels $Y \in \mathbb{Z}^{B\times T}$ (ignore index $= -100$)
Require: tokenizer $\mathcal{D}$, $\textsc{CombinedSimilarity}(\cdot,\cdot,\alpha)$
Require: hyperparams: λ (potracker_weight), N (potracker_every), mode $\in \{\texttt{1-minus}, \texttt{inv}\}$, ϵ, clip range $[w_{\min}, w_{\max}]$, EMA factor β
Require: state: call_count, EMA baseline μ (initialized as `None`)
Ensure: scalar loss

$L' \leftarrow L[:, 1{:}T, :]$; $Y' \leftarrow Y[:, 2{:}T]$ {causal shift (teacher forcing)}
$ce_{\text{tok}} \leftarrow \textsc{CrossEntropy}(L', Y', \texttt{reduction=none}, \texttt{ignore} = -100)$ $\{B \times (T{-}1)\}$
$m \leftarrow \mathbb{I}[Y' \neq -100]$; $c \leftarrow \max(1, \sum_t m)$ {valid-token mask/count per example}
$ce_{\text{ex}} \leftarrow \frac{\sum_t (ce_{\text{tok}} \odot m)}{c}$ {per-example CE, shape $(B,)$}
call_count $\leftarrow$ call_count $+1$
do_score $\leftarrow$ (`call_count` mod $N = 0) \wedge (\lambda > 0)$
if do_score **then**
  $\hat{Y} \leftarrow \arg\max_v L$ {token-level argmax predictions, shape (B, T)}
  for $i = 1$ to B **do**
    $S_i \leftarrow \{t \mid Y_{i,t} \neq -100\}$ {positions used for scoring}
    $y_i \leftarrow \mathcal{D}(\, Y_{i,S_i}\,)$; $\hat{y}_i \leftarrow \mathcal{D}(\, \hat{Y}_{i,S_i}\,)$ {decode to text/XML}
    $s_i \leftarrow \textsc{CombinedSimilarity}(y_i, \hat{y}_i, \alpha)[\texttt{combined_similarity}]$ {typically in $[0, 1]$}
    if mode == `inv` **then**
      $p_i \leftarrow \frac{1}{s_i + \epsilon}$
    else
      $p_i \leftarrow 1 - s_i$
    end if
  end for
  $\bar{p} \leftarrow \frac{1}{B} \sum_i p_i$
  if μ is `None` **then**
    $\mu \leftarrow \bar{p}$
  else
    $\mu \leftarrow \beta\mu + (1 - \beta)\bar{p}$
  end if
  $w_i \leftarrow \textsc{Clip}\big(1 + \lambda(p_i - \mu),\, w_{\min},\, w_{\max}\big)$
  $w \leftarrow \textsc{StopGrad}(w)$ {detach weights from backprop}
else
  $w \leftarrow \mathbf{1}$ {no reweighting this batch}
end if
$\frac{1}{B} \sum_{i=1}^{B} \big(w_i \cdot ce_{\text{ex},i}\big)$ {important: mean($w \cdot ce$), not normalized by $\sum w$}

Algorithm 5 High-Level Bad-Response Generation for DPO (XML Corruption)

Require: Chosen XML string x^+, RNG $\mathcal{R}$, optional other chosen XML x^{other}
Ensure: Rejected XML string x^-

$x^+ \leftarrow \textsc{Trim}(x^+)$
$(ok, T) \leftarrow \textsc{TryParseXML}(x^+)$
if $\neg ok$ **then**
  $\textsc{FallbackCorrupt}(x^+, \mathcal{R})$
end if
$E \leftarrow \textsc{AllElements}(T)$
$P \leftarrow \textsc{BuildParentMap}(T)$
$L \leftarrow \textsc{LeafTextNodes}(T)$ {LEAVES WITH NONEMPTY TEXT}
$O \leftarrow \emptyset$
if $x^{\text{other}} \neq \emptyset$ **then**
  $(ok_2, T_2) \leftarrow \textsc{TryParseXML}(\textsc{Trim}(x^{\text{other}}))$
  if ok_2 **then**
    $O \leftarrow \textsc{LeafTexts}(T_2)$ {POOL OF PLAUSIBLE TEXT VALUES}
  end if
end if
$\mathcal{S} \leftarrow [\textsc{SwapLeafText}, \textsc{ReplaceLeafText}, \textsc{RemoveElement}, \textsc{NumericPerturb}]$
$s \leftarrow \textsc{RandomChoice}(\mathcal{R}, \mathcal{S})$
if $s = \textsc{SwapLeafText}$ **and** $|L| \geq 2$ **then**
  $(a, b) \leftarrow \textsc{RandomSample2}(\mathcal{R}, L)$
  $\textsc{SwapText}(a, b)$
else if $s = \textsc{ReplaceLeafText}$ **and** $|L| \geq 1$ **then**
  $t \leftarrow \textsc{RandomChoice}(\mathcal{R}, L)$
  if $|O| \geq 1$ **then**
    $\textsc{SetText}(t,\ \textsc{RandomChoice}(\mathcal{R}, O))$
  else
    $\textsc{SetText}(t,\ \textsc{RandomChoice}(\mathcal{R}, \{\texttt{UNKNOWN}, \texttt{N/A}, \texttt{0}, \texttt{999999}, \texttt{Pending Investigation}\}))$
  end if
else if $s = \textsc{RemoveElement}$ **then**
  $C \leftarrow \textsc{NonRootRemovable}(E, P)$
  if $|C| \geq 1$ **then**
    $v \leftarrow \textsc{RandomChoice}(\mathcal{R}, C)$
    $\textsc{RemoveChild}(P[v], v)$
  else if $|L| \geq 1$ **then**
    $\textsc{SetText}(\textsc{RandomChoice}(\mathcal{R}, L),\ \texttt{UNKNOWN})$
  end if
else if $s = \textsc{NumericPerturb}$ **and** $|L| \geq 1$ **then**
  $t \leftarrow \textsc{RandomChoice}(\mathcal{R}, L)$
  if IsNumeric(GetText(t)) **then**
    $v \leftarrow \textsc{ToFloat}(\textsc{GetText}(t))$
    $v \leftarrow v + \textsc{RandomSign}(\mathcal{R}) \cdot \textsc{Uniform}(\mathcal{R}, 1, 1000)$
    $\textsc{SetText}(t,\ \textsc{FormatNumber}(v))$
  else
    $\textsc{SetText}(t,\ \textsc{RandomChoice}(\mathcal{R}, \{\texttt{0}, \texttt{UNKNOWN}, \texttt{999999}\}))$
  end if
end if
$x^- \leftarrow \textsc{SerializeXML}(T)$
$x^- \leftarrow \textsc{EnsureXMLDeclaration}(x^-)$
$x^+_{decl} \leftarrow \textsc{EnsureXMLDeclaration}(x^+)$
if $\textsc{Trim}(x^-) = \textsc{Trim}(x^+_{decl})$ **or** $\textsc{Length}(x^-) < 60$ **then**
  $x^- \leftarrow \textsc{FallbackCorrupt}(x^+, \mathcal{R})$
end if
x^-

Table 2: Set of rules for conversion from non-standard to standard outage report

Nonstandard outage report	Standard outage report
Outage/mRID	<ns0:mRID>OUTAGE- {area}-0001</ns0:mRID>
Outage/communityDescriptor	<ns0:communityDescriptor> {area}</ns0:communityDescriptor>
Outage/metersAffected	<ns0:metersAffected> {metersAffected} </ns0:metersAffected>
Outage/EstimatedRestorationTime/ert	<ns0:ert>{etr}</ns0:ert>
Outage/OutageArea/metersServed	<ns0:metersServed>{metersServed} </ns0:metersServed>
Outage/OutageArea/outageAreaKind	<ns0:outageAreaKind>zipcode </ns0:outageAreaKind>
Outage/Incident/cause	<ns0:cause>Pending Investigation</ns0:cause>
Outage/Incident/Location/ geoInfoReference	<ns0:geoInfoReference>{area} </ns0:geoInfoReference>
Outage/Incident/Location/zoneKind	<ns0:zoneKind>zipcode< /ns0:zoneKind>

Table 3: Settings for Qwen-2.5-7B-Instruct in POTracker's experiment

Generation Parameter	Default
do_sample	true
temperature	0.7
top_p	0.8
top_k	20
repetition_penalty	1.05

B Rule-based approach

Table 2 shows the subset of rules we use for the rule-based approach for converting from non-standard to standard outage report.

C Settings for Qwen-2.5-7B-Instruct in POTracker's experiment

Table 3 shows the parameters we set as default for POTracker's experiment.

D Human Study on Quality of Ground-truth Labels

E Accuracy on varying weight for POTracker

We adjust the weight of POTracker_Loss as α from 0 to 1 and see the change on overall accuracy. We get the accuracy as of POTracker as shown in Table 5.

Table 4: Distribution of human rating scores for generated ground-truth standard reports. Domain experts rated each label on a 0–5 scale.

Human Rating Score	**Number of Data Points**
0	1
1	0
2	12
3	20
4	16
5	51
Average	**4.03**
Median	**5**

Table 5: Accuracy of POTracker under different values of the weighting factor α.

α	**Accuracy**
0.1	83.00%
0.2	79.54%
0.3	76.07%
0.4	72.60%
0.5	69.14%
0.6	65.67%
0.7	62.21%
0.8	58.74%
0.9	55.27%
1.0	51.81%