Cleaning Semi-Structured Errors in Open Data Using Large Language Models

Manuel Mondal
Université de Fribourg, Switzerland
manuel.mondal@unifr.ch

Julien Audiffren
Université de Fribourg, Switzerland
julien.audiffren@unifr.ch

Ljiljana Dolamic
armasuisse W+T, Switzerland
ljiljana.dolamic@armasuisse.ch

Gérôme Bovet
armasuisse W+T, Switzerland
gerome.bovet@armasuisse.ch

Philippe Cudré-Mauroux
Université de Fribourg, Switzerland
philippe.cudre-mauroux@unifr.ch

***Abstract*—Many datasets suffer from errors, rendering data cleaning, the process of rectifying these issues, very time-consuming. The most commonly studied errors encompass inaccuracies in data values or labels (data errors) and issues with data formatting that hinder parsing (structured errors). We focus on a distinct category of errors known as Semi-Structured Errors (SSEs), which occur when both the data values and the structure are correct, but the values are misplaced within the structure, requiring time-demanding and complex parsing rules (including their exceptions) to account for them. In this work, we explore the capabilities of Large Language Models to clean SSEs and show promising experimental results on a public dataset of Swiss federal law.**

***Index Terms*—Large Language Models, In-context Learning, Data Cleaning, Semi-Structured Errors, Open Data, Legal Data**

I. Introduction

With the rise of Open Data, an increasing number of datasets are published every day. This fast growth in available data has led to great progress in the field of applied machine learning, by allowing the use of increasingly complex models, such as Large Language Models (LLMs). However, most datasets contain multiple errors that impact a substantial subset of the available data. The process of removing these errors, or data cleaning, is often considered the most time-consuming part of the data mining pipeline (see [16] and references therein). These errors include data errors, where the values of some elements of the dataset or their label are incorrect and structured errors, where issues in the data formatting prevent its parsing (for instance, missing closing tags in an XML tree).

In this work, we take particular interest in a third category of error, named Semi-Structured Errors (SSEs). This type of error lies between structured errors and data errors and occurs when the structure of the data is valid, but their values, while correct, are often misplaced in the structure (e.g., the incorrect choice of an XML tag). SSEs are quite frequent among public datasets, in particular in structured data, such as collections of XML documents published by large organizations. Indeed, these collections tend to exhibit SSEs, often as a result of the dataset being aggregated over long periods of time, with formatting standards evolving for new documents without affecting prior ones and past mistakes being prohibitively difficult to correct. In consequence, querying structured content from a dataset contaminated with SSEs is an exceedingly time-consuming task, as each SSE may induce errors further down the data processing pipeline, and may require a posteriori changes to the parser, leading to many exceptions and special cases. Moreover, this process may have to be repeated for each new dataset or version of the same dataset, resulting in a very high cost in human labor.

In this paper, we study the task of using automated, LLM-based methods to address the task of cleaning SSEs, a task named hereafter Semi-Structured Data Cleaning (SSDC). Machine Learning algorithms, and LLMs in particular, have shown promising results when applied to data cleaning tasks (see [5] and references therein). For example, [3] have recently shown that LLMs can be used to remove noise (defined by the authors as "spelling and grammatical errors, emojis, internet slang, and profanities") from the data. Previous work has also successfully applied LLMs to clean XML data: for instance, [2] used LLMs to clean structured errors from an XML dataset, while [10] evaluated LLMs on five data cleaning and integration tasks, centered around data error detection. However, to the best of our knowledge, the task of SSDC has not been specifically addressed before, as previous work focuses on data with structured errors.

Hence, in this work, we take particular interest in the use of LLMs to correct SSEs, by rewriting the SSDC problem as a text generation task. Our contributions are as follows: we first collect a public dataset of Swiss federal law and analyze it to highlight the numerous SSEs it contains. Then, we explore the potential use of LLMs on SSDC and provide a new framework to evaluate them on this task. Our chosen approach leverages a one-shot learning prompt and thus does not require any fine-tuning of the LLMs [4]. Finally, our experiments show that LLMs exhibit promising performance for the SSDC task on our dataset, but that further fine-tuning of the process is required to fully exploit their potential. The code for our analyses and experiments is released at `github.com/eXascaleInfolab/LLMFixer`.

II. Semi-Structured Data Cleaning

A. Problem statement

We address the task of semi-structured data cleaning (SSDC) focusing on XML documents, but our definition can easily be extended to further structured (or semi-structured) data. In our setting, the dataset is made of structured documents (for instance XML trees), for which there exists an ideal template structure. We assume that data errors and structured errors are absent or have already been cleaned, that is to say for each document:

1) the content is correct, i.e., the text (and values) contained in the document, except for the structural elements, is correct;
2) the structure is correct, in the sense that e.g., the document is a valid XML document;
3) Semi-Structured Errors may be present. In other words, there may be a deviation in the structure of the documents compared to the template. For instance, XML tags may be different, or part of the content may be contained in the wrong tags (see Section II-C for examples of such problems we encountered in practice).

The objective of SSDC is then to fix the documents of the dataset that are amiss compared to the template. We believe that the SSDC is pervasive in many real-world datasets, as data pieces are often merged from multiple sources, leading to slight inconsistencies, or when data is manually edited. Furthermore, the SSEs may be difficult to detect, as they do not impair the parsing of the document. Thus, finding an automated method to address these errors may help with the building of pipelines to collect and process data.

Here, we choose to address the SSDC problem by reformulating it as a text generation task, specifically, a sequence to sequence problem [11]. By casting the task in this manner, we can leverage recent advances from the field of Natural Language Processing, and in particular Large language models, which have been shown to exhibit strong performance in sequence to sequence tasks [14].

B. Dataset

To illustrate the SSDC problem and explore potential solutions to it, we use a real-world dataset of legal texts of Swiss federal law. The dataset is a collection of all federal acts currently in effect in Switzerland. Each federal act consists of one or more articles, each composed of one or more paragraphs, which are the basic unit of legal reasoning [9]. In the legal context, the structure of an act is considered an essential component of its content, as paragraphs refer to each other by their number and their structure is taken into account during systematic interpretations [15].

Our dataset was collected by downloading the XML version of each federal act from the API of the Federal Chancellery of Switzerland[1]. The (simplified) ideal structure of a federal act is shown in Figure 1. Note that additional elements were

```
<article>(1..n-times)
 <paragraph>(1..m-times)
  <num>{Paragraph number}</num>
  <content>{Paragraph content}</content>
 </paragraph>
</article>
```

Fig. 1. The template of a well-structured article in the Federal Law Dataset.

omitted for simplicity, such as `<section>`, `<chapter>`, and `<title>` tags for grouping articles, `<heading>` tags for article titles, `<blockList>` elements for enumerations inside a paragraph, as well as `<authorialNotes>` used for footnotes. While the extraction of the articles can be handled by a static XML parser, the automated extraction of individual paragraphs is only possible when the template shown above is met. However, the collected dataset contains multiple inconsistencies and SSEs[2] (see below), which may prevent the automatic processing of articles and paragraphs, thereby constituting an SSDC problem.

C. Semi-Structured Errors

To assess the presence of SSEs in the dataset, we performed a repeated screening test with inspection error [6] in batches of 20 articles. As a result, we identified the following two categories of SSE:

a) Numbering Issues: When an article of law contains multiple paragraphs, each of them must be numbered (e.g., using numbers, letters, Roman numerals). A properly formatted article thus contains pairs of `<num><content>` elements, the former containing a paragraph's numbering and the latter its textual content. Examples of numbering SSEs include numbering schemes incompatible with this template, or the misuse of a paragraph's `<num>` element to contain other text elements (e.g., footnotes, superscripts).

b) Text Misplacement Issues: SSEs of this family occur, for instance, when the paragraph numbering is not separated into a distinct structural element. Instead, it may be placed inside the paragraph's `<content>` element, a table, or a list. Other examples include the substitution of the paragraph structure by a blocklist or table structure.

Figure 2 shows two examples of articles with an SSE. In both cases, the SSEs render the automatic extraction of legal paragraphs difficult, and our repeated screening tests revealed that 21% of the 127k articles in the dataset exhibit such errors, which amounts to 73% of the 4593 federal acts containing at least one article with an SSE[3]. While it is technically possible to adapt a parser to account for these issues, such a parser would need to be modified to account for each individual possible past and future SSE in the dataset. This, in turn,

[1]SPARQL endpoint: `fedlex.data.admin.ch`. Queried in Nov. 2023.

[2]Federal law having been digitized over decades and spanning various scopes (e.g., international treaties, federal acts, cantonal constitutions) results in inconsistent paragraph structures.

[3]Note that this dataset, including its structuring issues, is used to generate the content of the Swiss Federation's official publication platform, as can be visually confirmed by e.g., comparing the formatting of `Art. 64 MStG` with `Art. 66 MStG` or see `Art. 8 Abs. 1 ALBAG`.

```
<article eId="art_64">
 <num><b>Art. 64</b></num>
 <paragraph eId="art_64/para">
  <content>
   <p>1.  Wenn mehrere sich {...}</p>
   <p>In leichten Fällen erfolgt {...}</p>
   <p>2.  In Kriegszeiten kann auf {...}</p>
  </content>
 </paragraph>
</article>
```

```
<article eId="art_8">
 <num><b>Art. 8</b></num>
 <heading> Einreichungspflicht {...}</heading>
 <paragraph>
  <content>
   <blockList>
    <item>
     <num><sup>1</sup>Die ESTV kann {...}:</num>
    </item>
    <item><num>a.</num><p>{...}</p></item>
    <item><num>b.</num><p>{...}</p></item>
   </blockList>
  </content>
 </paragraph>
 {....}
</article>
```

Fig. 2. Examples of articles with semi-structured errors (Art. 64 MStG, Art. 8 ALBAG). In the top article, the paragraphs are at the same level of separation as a line break and their numbering is not in its own XML element, while in the bottom article, the numbering and beginning of paragraph 1 are intertwined with the subsequent enumeration.

may require a significant amount of human labor, both for the detection and the correction of each error, hence motivating the present exploration for an automated solution.

III. Experiments

We investigate here the capabilities of LLMs to perform SSDC on the previously built dataset of XML documents.

Models: We evaluated the performance of a total of five models, four of which have publicly available weights. Three small 7 billion parameter models, Llama-2-7b-chat-hf (Llama-2-7B) [13], DeciLM-7B-instruct (DeciLM-7B) [12] and Mistral-7B-Instruct-v0.2 (Mistral-7B) [7] were run on a local cluster of V100-equipped machines. The mid-sized 47 billion parameter model Mixtral-8x7B-Instruct-v0.1 (Mixtral-8x7B) [8] was queried from a hosted inference API[4]. Finally, the model gpt-4-1106-preview (GPT-4 Turbo) [1] of unknown parameter size was queried from the vendor's API[5].

Prompt: We started by comparing multiple prompts on a small validation set and selected the overall best-performing phrasing to carry out the subsequent experiments. The final versions of each model's prompt can be found in the linked repository. The prompt contains general instructions, a problematic example with multiple SSEs, its fixed version matching the ideal template of Figure 1, and finally, the XML data to clean. The problematic example was built by selecting an article, manually verifying its validity, and introducing one occurrence of each of the previously discussed SSE. The resulting prompt was finally adapted to the specifications of each model (e.g., special tokens), following the guidelines of the respective papers or platforms.

SSDC Evaluation: We start by evaluating the performance of the different models to address the SSEs we detected during the analysis of the dataset (see Section II-C). Due to the lack of an available ground truth for the different articles, we built a synthetic evaluation dataset, in which all error-prone samples are paired with an appropriate error-free version. To create this dataset, we first designed automated parsers, that filtered out all the articles having at least one of the aforementioned SSEs (Section II-C). We then randomly selected 200 articles that were considered without errors[6]. Then, all articles were modified to exhibit one of the SSEs, resulting in 1000 altered articles, as well as their error-free counterpart. Each LLM was then tasked with cleaning these articles, and their results were evaluated using four criteria:

1) **ValidXML** Whether the output contains a valid XML tree, with an `<article>` as its root element[7].
2) **NoSSE** Whether the resulting XML still contains any of the known SSEs.
3) **SameXML** Whether the response and the ground truth have the exact same XML tree (e.g. tag names).
4) **SimilarText** The normalized Levenshtein similarity between the text contents of the response XML tree and the ground truth XML tree. This is computed by extracting the textual content of each XML node in the two XML trees and comparing the similarity of the resulting two strings. Text similarity is particularly key in applications on legal data, as even a mild modification (e.g., replacing synonyms) may alter the entire meaning of a law and is not an admissible effect of a data cleaning procedure.

To complete our analysis, we performed a manual assessment of the response generated by each model on a small test set of 10 articles for which no ground truth is available, and which exhibit common issues and problems that may not be captured by automated scoring functions. The objective is to inspect the quality of the data cleaning result and identify which issues remain after processing.

Results

The results of our experiment, presented in Table I, show a heterogeneity between the scores of the models among the different metrics. On the one hand, it should be noted that all models were able to produce valid XMLs trees in the vast majority of cases, as shown by the high share of **ValidXML**. The rare cases where this criterion is not met are generally caused by the models either spending too many tokens of their context on verbal comments and thus providing incomplete answers, or not matching XML tags adequately.

[4]Hugginface Inference API: `huggingface.co/docs/api-inference`.
[5]OpenAI API: `platform.openai.com/docs`.

[6]Note that only the previously detected errors are filtered out with this method, thus additional, rarely occurring issues may have been missed.
[7]In case the answer contains multiple XML trees, we only consider the last complete XML tree.

On the other hand, the behaviors of the different models vary significantly according to the other metrics. Indeed, the smaller models, i.e., Llama-2-7B, DeciLM-7B, and Mistral-7B, are generally not able to repair the SSEs they are tasked to handle, resulting in rather low **NoSSE** scores. Even the larger Mixtral-8x7B model is able to clean all issues in less than two-thirds of the samples. Conversely, the results returned by GPT-4 Turbo appear to be SSE-free in over 93% of cases. However, according to our manual analysis, SSE-free outputs are not always perfect results. For instance, in some responses, the model properly detects that paragraphs contained within the text ought to be split into multiple elements, but creates too many paragraphs. The resulting XML data thereby exhibits a new type of SSE and does not match the ideal template.

Similarly, the proportion of exact matches in the tree structure between the model's responses and ground truth (**SameXML**) is small, even for GPT-4 Turbo. By investigating the responses, we observed two main causes for this discrepancy: (i) models returning only the provided input without modifications and (ii) the removal of elements that the models considered secondary (e.g., footnotes, section headers, metadata). While the latter case can sometimes be an admissible response (e.g., when italics are removed), it may also result in the loss of essential legal information.

Finally, the comparison of the output and ground truth text content strings (**SimilarText**) reveals that all models tend to alter the sample's content. As stated above, certain modifications, such as changes to the punctuation of the paragraph numbers, are unproblematic. However, looking further into the results, we discovered that smaller models often change terms, translate phrases, and omit parts of sentences, which is not an admissible output in our context. This behavior is significantly less frequent in larger models.

	ValidXML	**NoSSE**	**SameXML**	**SimilarText**
Llama-2-7B [13]	**0.999**	0.392	0.206	0.692
DeciLM-7B [12]	0.947	0.189	0.229	0.784
Mistral-7B [7]	0.995	0.594	0.107	0.753
Mixtral-8x7B [8]	0.973	0.631	0.257	**0.897**
GPT-4 Turbo [1]	**0.999**	**0.937**	**0.352**	0.878

TABLE I
SCORES OF THE FIVE LLM MODELS ON THE SSDC TASK.

IV. CONCLUSION

While the results of our experiments are promising according to multiple metrics, a close inspection of the results reveals three types of incorrect answers: a result that did not fix any (or not all) of the SSEs, an output XML tree where elements were moved or omitted, and a response with significant changes in the text. In particular, it should be noted that our **NoSSE** metric only measures the presence of previously discussed SSEs (Section II-C) and that the example provided in the prompt only exhibits these SSEs, making it particularly challenging for LLMs to handle previously unseen types of SSEs.

However, the net performances of LLMs on our task remain promising, as for instance GPT-4 Turbo was able to fix most of the provided SSEs. Given that none of the studied models has been fine-tuned towards data cleaning, legal articles, or XML in general, and that they were only provided with a one-shot example of the task, our results highlight the potential of these methods. It is therefore the authors' belief that with further training and adaptation, LLMs could be a viable tool to help with the challenging SSDC task.

REFERENCES

[1] Josh Achiam et al. "GPT-4 Technical Report". In: *arXiv preprint arXiv:2303.08774* (2023).

[2] Simran Arora et al. "Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes". In: *arXiv preprint arXiv:2304.09433* (2023).

[3] Quinten Bolding et al. "Ask Language Model to Clean Your Noisy Translation Data". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*.

[4] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020).

[5] Pierre-Olivier Côté et al. "Data Cleaning and Machine Learning: A Systematic Literature Review". In: *arXiv preprint arXiv:2310.01765* (2023).

[6] Mauro Gasparini, Harald Nusser, and Jeffrey Eisele. "Repeated screening with inspection error and no false positive results with application to pharmaceutical pill production". In: *Journal of the Royal Statistical Society Series C: Applied Statistics* 53.1 (2004).

[7] Albert Q Jiang et al. "Mistral 7B". In: *arXiv preprint arXiv:2310.06825* (2023).

[8] Albert Q Jiang et al. "Mixtral of Experts". In: *arXiv preprint arXiv:2401.04088* (2024).

[9] Ernst A Kramer. *Juristische Methodenlehre*. 1998.

[10] Avanika Narayan et al. "Can Foundation Models Wrangle Your Data?" In: *Proc. VLDB Endow.* 16.4 (2022).

[11] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems* 27 (2014).

[12] DeciAI Research Team. *DeciLM-7B*. 2023. URL: https://huggingface.co/Deci/DeciLM-7B.

[13] Hugo Touvron et al. "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (2023).

[14] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[15] Rolf Wank. "Juristische Methodenlehre: eine Anleitung für Wissenschaft und Praxis". In: Academia Iuris (2020).

[16] Steven Euijong Whang et al. "Data collection and quality challenges in deep learning: A data-centric ai perspective". In: *The VLDB Journal* 32.4 (2023).