Evaluating Structured Output Robustness of Small Language Models for Open Attribute-Value Extraction from Clinical Notes

Nikita Neveditsin[1], Pawan Lingras[1], Vijay Mago[2]
[1]Saint Mary's University, Halifax, Canada
[2]York University, Toronto, Canada

Abstract

We present a comparative analysis of the parseability of structured outputs generated by small language models for open attribute-value extraction from clinical notes. We evaluate three widely used serialization formats: JSON, YAML, and XML, and find that JSON consistently yields the highest parseability. Structural robustness improves with targeted prompting and larger models, but declines for longer documents and certain note types. Our error analysis identifies recurring format-specific failure patterns. These findings offer practical guidance for selecting serialization formats and designing prompts when deploying language models in privacy-sensitive clinical settings.

1 Introduction

Structured information extracted from clinical narratives enhances clinical decision-making, streamlines reporting, and facilitates research database development. Small language models (SLMs) (Schick and Schütze, 2021) can be deployed on local hardware and therefore meet privacy requirements, but their utility depends on producing outputs that downstream software can parse automatically.

This work examines **open attribute-value extraction**, a task in which an SLM identifies clinically relevant attribute-value pairs *without a predefined schema* and serializes them in a standard format (Etzioni et al., 2008; Zheng et al., 2018; Li et al., 2023; Brinkmann et al., 2025). We compare three commonly used formats: JSON, YAML, and XML, and assess robustness via *parseability*, defined as the proportion of outputs that can be successfully validated by a standard parser without manual correction. We further analyze how document length, note type, model size, and extraction scope (open vs. targeted for medications, symptoms, and demographics) affect parseability, and report on common structural failure modes and key interactions among these factors.

Our contributions are as follows: (i) to the best of our knowledge, we provide the first comparative analysis of structured output parseability across three widely used serialization formats (JSON, YAML, XML) in the context of open attribute-value extraction from clinical notes; (ii) we demonstrate how model size, prompt specificity, and clinical document characteristics systematically influence structural robustness; (iii) we identify and categorize recurrent structural failure modes, offering practical insights into common format-specific vulnerabilities in SLM-generated outputs.

2 Related Work

Prior work on structured information extraction with transformer-based language models has highlighted both their semantic potential and their syntactic fragility. Research in this area can be broadly categorized by its primary evaluation focus: studies that prioritize the semantic accuracy of the extracted content, and those that more directly engage with the technical challenge of ensuring syntactic validity.

In high-stakes domains such as clinical medicine, the evaluation emphasis is typically on semantic accuracy. For example, Balasubramanian et al. (2025) evaluated the extraction of 51 features from breast cancer pathology reports by comparing model outputs against expert-annotated gold standards. Similarly, Kadhim et al. (2025) measured the correctness of extracted findings in inflammatory bowel disease reports using F1 scores. In both cases, models like LLaMA-3.3 were assessed primarily on their ability to extract correct clinical content. Syntactic validity, such as whether outputs conformed to a given format, was assumed rather than explicitly evaluated. Other studies, such as Elnashar et al. (2025), explored prompt design and

efficiency trade-offs across JSON, YAML, and hybrid CSV formats using GPT-4o. While they validated attribute-level correctness, structural robustness was not a primary focus.

This focus on semantics often coexists with an implicit acknowledgment of the syntactic fragility of unconstrained model outputs. Work in scientific and technical domains has more directly quantified this issue. Dagdelen et al. (2024), in the context of materials science extraction, noted parse failures under token limits. Schilling-Wilhelmi et al. (2024) advocates constrained decoding to restrict the model's vocabulary during generation to enforce structural compliance. While this technique improves parseability, Tam et al. (2024) have shown that tighter constraints may also reduce reasoning flexibility, underscoring a trade-off between structural validity and expressiveness.

These findings indicate a gap in evaluating the syntactic reliability of structured outputs. Our study addresses this by focusing specifically on parseability as the primary evaluation criterion, using small instruction-tuned models.

3 Methodology

3.1 Models

To assess the impact of output format on small language models, we evaluate seven open-weight instruction-tuned models (Table 1).

Model	Vendor	Params (B)	Ctx. Window
`Phi-4` (Abdin et al., 2024b)	Microsoft	14	16K
`Phi-3.5-mini` (Abdin et al., 2024a)	Microsoft	3.8	128K
`Llama-3.2-3B` (Grattafiori et al., 2024)	Meta	3	128K
`Llama-3.1-8B` (Grattafiori et al., 2024)	Meta	8	128K
`Mistral-8B` (Jiang et al., 2023)	Mistral AI	8	128K
`Qwen3-4B` (Qwen Team, 2024)	Alibaba	4	32K
`Qwen3-14B` (Qwen Team, 2024)	Alibaba	14	128K

Table 1: SLMs evaluated in this study.

We selected 7 models from 4 vendors (Microsoft, Meta, Mistral, Alibaba), some of which contributed more than one model. This allowed us to reduce provider-specific bias while also covering a range of model sizes (3–14B parameters) and context window capacities (ranging from 16K to 128K tokens, as shown in Table 1). All models are openly available, support local deployment, and are widely used in the open-source community, ensuring relevance, reproducibility, and suitability for privacy-sensitive clinical use.

3.2 Data

We use the **EHRCon** (Goldberger et al., 2000; Kwon et al., 2025) dataset, a standardized, open, and ethically compliant subset of MIMIC-III (Johnson et al., 2016) that supports reproducible research. It includes 105 randomly selected, de-identified clinical notes with 4,101 annotated entities mapped to 13 structured EHR tables. Derived from a large critical care database, EHRCon captures the complexity of real-world clinical documentation. Its public availability and prior ethical clearance make it suitable for secondary analysis without requiring additional ethical review. EHRCon is well-suited for evaluating structural parseability, and its detailed attribute-level annotations offer opportunities for future research on semantic validity, though we do not pursue that direction in this work.

The dataset includes three note types: discharge summaries, nursing notes, and physician notes, each with distinct content and length characteristics (Table 2). Discharge summaries, the longest (avg. 1300 words, 2700 tokens), provide a comprehensive account of the hospital stay. Physician notes, of moderate length, focus on assessments and treatment plans. Nursing notes, the shortest, document vitals, patient behavior, and routine care.

Type	# Documents	Avg. Words	Avg. Tokens
Discharge	38	1306.47	2764.46
Nursing	36	490.33	1153.63
Physician	33	669.91	1914.93

Table 2: Descriptive statistics of clinical note types.

Token counts are computed by applying each model's tokenizer to every document and averaging across models from Table 1.

3.3 Experimental Setup

We assess SLMs in two extraction scenarios. The *open* format scenario prompts the model to extract any medically relevant information it can infer from a note without relying on a predefined schema. This reflects exploratory or retrospective use cases where schema coverage may be incomplete or unavailable. The *targeted* scenario narrows the prompt to a specific category: medications, symptoms, or demographics. These categories are commonly prioritized in clinical information extraction for their central role in decision support and downstream clinical tasks (Sohn et al., 2013; Wang et al., 2018). This allows us to assess whether more constrained prompts yield more structurally

consistent outputs.

Figure 1 illustrates the overall workflow. A clinical note is processed under one of the two prompting conditions, passed to an SLM, and rendered in JSON, YAML, or XML. The output is then evaluated for parseability using a standard parser.

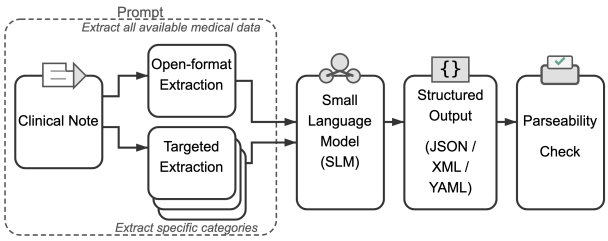

Figure 1: Workflow for evaluating structured output generation

In both scenarios, we focus on parseability; we do not evaluate content accuracy. Formally, for a given model, prompt type, and a set of documents D, we define the **parseability rate** as

$$\rho(D) = \frac{n_v}{|D|},$$

where n_v denotes the number of documents in D whose outputs were successfully parsed by a standard parser under that model and prompt type. To support our findings, we apply appropriate statistical tests. Appendix A provides additional details on the experimental setup.

4 Results

Table 3 presents parseability rates across JSON, YAML, and XML for all models listed in Table 1, evaluated on the full clinical document set. Each model appears in two rows, corresponding to the open-ended and targeted extraction settings (the *Setting* column).

Parseability tends to improve with model size. To assess this effect, we grouped models by parameter count into three categories: *Small* (3-4B), *Medium* (8B), and *Large* (14B). A Chi-squared test of independence confirmed a significant association between model size and parseability ($\chi^2 = 106.72$, $p \ll 0.05$). Average parseability rates rose with size: Large models achieved 90.3%, followed by Medium (82.6%) and Small (80.9%). The effect size, measured by Cramér's $V = 0.11$, suggests a statistically significant but modest association between model size and parseability.

Prompt specificity was also a significant factor. Targeted prompts substantially boosted parseability across all formats, especially for YAML, which performs poorly in the open setting. A Chi-squared test confirmed a strong association between prompt type and parseability ($\chi^2 = 1579.41$, $p \ll 0.05$). Cramér's $V = 0.42$ indicates a medium-to-large impact of prompt type on structural validity.

Model	Setting	JSON	XML	YAML
Llama-3.1-8B	Open	**59.8**	54.2	*23.4*
Llama-3.1-8B	Targeted	**97.8**	96.9	*92.2*
Llama-3.2-3B	Open	**73.8**	41.1	*29.9*
Llama-3.2-3B	Targeted	**94.4**	81.6	*75.1*
Mistral-8B	Open	**81.3**	57.9	*47.7*
Mistral-8B	Targeted	**96.0**	89.1	*80.4*
Phi-3.5-mini	Open	**83.2**	*43.0*	52.3
Phi-3.5-mini	Targeted	**99.4**	94.7	*83.5*
Phi-4	Open	**100.0**	61.7	*44.9*
Phi-4	Targeted	**100.0**	98.4	*97.8*
Qwen3-14B	Open	**98.1**	*43.0*	47.7
Qwen3-14B	Targeted	**99.4**	*97.2*	97.5
Qwen3-4B	Open	**95.3**	39.3	*29.0*
Qwen3-4B	Targeted	**97.2**	94.4	*86.3*

Table 3: Parseability rates (%) by model and output format across the full document set. Each model appears in two rows, corresponding to open-ended and targeted extraction settings (prompt types). **Bold** indicates the highest parseability per row; *italic* indicates the lowest.

To test for the statistical significance of differences in parseability across output formats, we conducted paired McNemar's tests and report the results in Table 4.

Comparison	χ^2	p-value
JSON vs YAML	167.607	$\ll 0.05$
JSON vs XML	69.351	$\ll 0.05$
YAML vs XML	32.411	$\ll 0.05$

Table 4: Paired McNemar's test results comparing parseability outcomes across formats

All comparisons yield statistically significant results, with JSON significantly outperforming both YAML and XML ($p \ll 0.05$ in both cases). The difference between YAML and XML is also significant ($p \ll 0.05$), though comparatively smaller in effect size.

Figure 2 illustrates the relationship between document length (in words) and parseability, separately for the open and targeted extraction scenarios. In both scenarios, documents that failed to parse tend to be longer, with noticeably higher medians and more dispersed distributions compared to parseable documents.

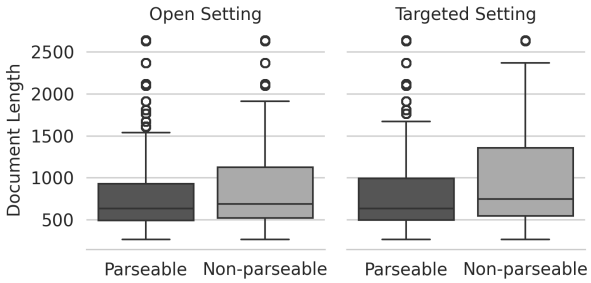

Figure 2: Boxplot showing the distribution of document lengths (in words) for parseable and non-parseable outputs.

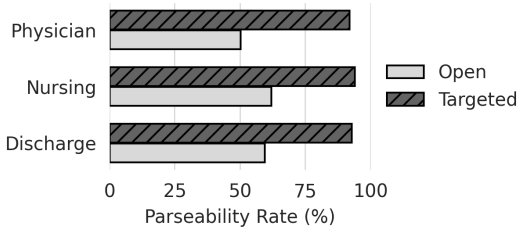

Figure 3: Parseability rates by document type for open and targeted extraction settings. Bars show the percentage of successfully parsed documents within each type.

To quantify the relationship between document length and parseability, we computed the point-biserial correlation. Across all documents, the correlation was weak but statistically significant ($r = -0.081, p \ll 0.05$). When analyzed by scenario, the negative correlation was slightly stronger in the open setting ($r = -0.118, p \ll 0.05$) compared to the targeted setting ($r = -0.077, p \ll 0.05$). These results suggest that longer documents are consistently less likely to be parsed successfully, especially in open-ended generation scenarios. However, despite statistical significance, the small effect size and substantial overlap in length distributions between parseable and non-parseable documents (Figure 2) indicate that length alone does not strongly determine parseability. This suggests the presence of potential confounding factors such as note type, which we examine further.

Figure 3 shows parseability rates across the three clinical document types, separated by extraction scenario. Targeted prompting consistently improves parseability for all types, with the most pronounced gain observed in physician notes. Nursing notes achieve the highest parseability overall, while physician notes lag behind in the open setting. These differences likely reflect variations in document complexity and length, as shown in Table 2, where physician notes are among the longest on average. To assess whether document type is significantly associated with parseability, we conducted a chi-squared test of independence, yielding $\chi^2 = 23.93$, $p \ll 0.05$. This confirms that the observed differences across note types are unlikely to be due to chance, though the corresponding Cramér's $V = 0.05$ indicates a small effect size.

To isolate the effects of document type and length on parseability, we fit a logistic regression with parseability as the binary outcome. Results show that discharge notes, though longer on average, are more parseable than nursing notes ($\beta = 0.550$, $p < 0.05$), while physician notes are less parseable ($\beta = -0.204$, $p < 0.05$). Length itself negatively impacts parseability ($\beta = -0.0008$, $p < 0.05$). These findings suggest that document type affects parseability independently of length, likely due to semantic and structural differences.

To understand the structural differences suggested by the regression analysis, we performed a qualitative analysis of the notes. This analysis reveals distinct structural patterns that explain these findings. Discharge notes are more consistently templated, with consistent section headers and enumerated lists that facilitate structured parsing, even in longer documents. In contrast, physician notes are rich in semantically dense content and frequently include compact representations of clinical data, such as vitals and lab panels (e.g., *Ca^{++}: 8.3 mg/dL, Mg^{++}: 2.7 mg/dL, PO_4: 5.0 mg/dL*), that pose specific challenges for structured formatting. These notations often combine numbers, units, and symbols in complex strings that can break parsing when not properly quoted or escaped. Nursing notes fall in between, mixing structured elements like vitals and interventions with narrative descriptions of patient events. These semantic and structural distinctions, not length alone, appear to drive parseability differences across note types.

5 Error Analysis

We categorize parse errors into two broad groups. First, extraction-related errors (see Figure 4, "Extraction-related" portion) occur when a standard regular expression fails to extract a structured object from the model output. Notably, our analysis revealed that the majority of extraction-related errors stemmed from infinite repetitions (Holtzman et al., 2020) in the generated text.

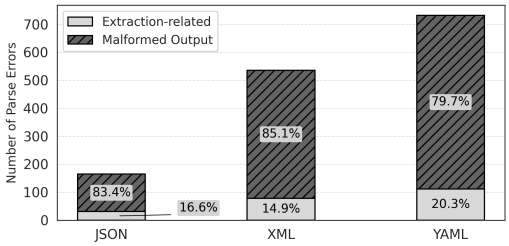

Figure 4: Breakdown of parse errors across JSON, XML, and YAML formats. Bars show the number of extraction-related and malformed output errors per format.

Second, malformed output errors, which arise when the output is syntactically invalid and cannot be parsed after successful extraction. Figure 4 shows the distribution of these error types across formats. A more detailed breakdown is provided in Appendix B.

To quantify the association between model size and types of parse errors, we grouped failed generations by model size. Among these, Large (14B) models produced only 2.4% extraction-related errors, compared to 21.0% and 19.0% for Medium (8B) and Small (3-4B) models, respectively. A Chi-squared test confirmed a statistically significant association between model size and error type ($\chi^2 = 45.52$, $p \ll 0.05$), with a Cramér's $V = 0.18$ indicating a small to moderate effect size. These findings suggest that extraction errors are more typical in smaller models, though they are not exclusive to them.

We also examined whether the type of parse error varied with prompt type. Open prompts resulted in extraction-related errors only 2.4% of the time, while targeted prompts produced extraction errors in 45.5% of failures. A Chi-squared test revealed a statistically significant association between prompt type and error type ($\chi^2 = 420.62$, $p \ll 0.05$), and Cramér's $V = 0.54$ indicated a large effect size. This suggests that extraction errors are a dominant failure mode under targeted prompting conditions.

Conclusion

We conducted a systematic evaluation of the structural robustness of SLM-generated outputs for open attribute-value extraction from clinical notes. Across three common formats, JSON significantly outperformed YAML and XML in parseability. Parseability improved with model size and prompt specificity, and targeted prompting yielded especially large gains for YAML. However, performance declined on longer documents, and physician notes were particularly error-prone. Error analysis revealed two dominant failure modes: infinite repetition and syntactic malformations, particularly missing quotation marks around numerals embedded in non-numeric fields (e.g., blood pressure values like "128/68"), unescaped special characters, and malformed list structures. These issues were most frequent in smaller models and underscore the need for decoding strategies that promote format-conformant output.

Our findings underscore the importance of aligning prompt and format design with generation strategies that ensure structural reliability, particularly in resource-constrained or privacy-sensitive clinical NLP settings. Future work should explore automatic post-processing techniques to detect and correct structural errors, extend parsers to better handle common irregularities in LLM-generated outputs, conduct more extensive evaluations on diverse clinical corpora, and support joint analysis of syntactic and semantic validity to better assess the clinical utility of structured outputs.

Limitations

While our study offers detailed insights into the structural robustness of SLM outputs, it has several limitations. First, the evaluation is based on the EHRCon dataset, which, although diverse in note types, contains only 105 documents and may not capture the full variability of clinical narratives. Second, all experiments were conducted using a single decoding configuration (greedy decoding without sampling), which may not generalize to alternative generation settings. Third, we evaluated a limited set of open-weight models. Future work should include domain-specific clinical language models and additional parameter sizes to capture broader trends. Finally, our analysis focused exclusively on syntactic parseability, without assessing the semantic accuracy or clinical correctness of the extracted information, which is an important direction for future research.

Ethics Statement

This study uses the EHRCon dataset, which is derived from the publicly available and de-identified MIMIC-III database. As no personally identifiable information is included in the data, and no new data collection was conducted, the study does

not require approval from an institutional ethics board. We do not publish any content that could potentially identify individuals. To promote transparency and reproducibility, we rely exclusively on open-source models and datasets, and provide detailed descriptions of our experimental setup and evaluation methodology.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024a. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024b. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Jeya Balaji Balasubramanian, Daniel Adams, Ioannis Roxanis, Amy Berrington de Gonzalez, Penny Coulson, Jonas S Almeida, and Montserrat García-Closas. 2025. Leveraging large language models for structured information extraction from pathology reports. *arXiv preprint arXiv:2502.12183*.

Alexander Brinkmann, Roee Shraga, and Christian Bizer. 2025. Extractgpt: Exploring the potential of large language models for product attribute value extraction. In *Information Integration and Web Intelligence*, pages 38–52, Cham. Springer Nature Switzerland.

John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S. Rosen, Gerbrand Ceder, Kristin A. Persson, and Anubhav Jain. 2024. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418.

Ashraf Elnashar, Jules White, and Douglas C Schmidt. 2025. Enhancing structured data generation with gpt-4o evaluating prompt efficiency across prompt styles. *Frontiers in Artificial Intelligence*, 8:1558938.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open information extraction from the web. *Commun. ACM*, 51(12):68–74.

Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3(1):160035.

Alex Z Kadhim, Zachary Green, Iman Nazari, Jonathan Baker, Michael George, Ashley Heinson, Matt Stammers, Christopher M Kipps, R Mark Beattie, James J Ashton, and 1 others. 2025. Application of generative artificial intelligence to utilise unstructured clinical data for acceleration of inflammatory bowel disease research. *medRxiv*, pages 2025–03.

Yeonsu Kwon, Jiho Kim, Gyubok Lee, Seongsu Bae, Daeun Kyung, Wonchul Cha, Tom Pollard, Alistair Johnson, and Edward Choi. 2025. Ehrcon: Dataset for checking consistency between unstructured notes and structured tables in electronic health records. *PhysioNet*.

Yanzeng Li, Bingcong Xue, Ruoyu Zhang, and Lei Zou. 2023. Attgen: Attribute tree generation for real-world attribute joint extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2139–2152.

Alibaba Cloud Qwen Team. 2024. Qwen3 language model. `https://huggingface.co/Qwen`. Accessed 2024-05-12.

Timo Schick and Hinrich Schütze. 2021. It‘s not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Mara Schilling-Wilhelmi, Martiño Ríos-García, Sherjeel Shabih, María Victoria Gil, Santiago Miret, Christoph T Koch, José A Márquez, and Kevin Maik Jablonka. 2024. From text to insight: large language models for materials science data extraction. *arXiv preprint arXiv:2407.16867*.

Sunghwan Sohn, Kavishwar B Wagholikar, Dingcheng Li, Siddhartha R Jonnalagadda, Cui Tao, Ravikumar Komandur Elayavilli, and Hongfang Liu. 2013. Comprehensive temporal information detection from clinical text: medical events, time, and tlink identification. *Journal of the American Medical Informatics Association*, 20(5):836–842.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1218–1236, Miami, Florida, US. Association for Computational Linguistics.

Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad, Sungrim Moon, Feichen Shen, Naveed Afzal, Sijia Liu, Yuqun Zeng, Saeed Mehrabi, Sunghwan Sohn, and 1 others. 2018. Clinical information extraction applications: a literature review. *Journal of biomedical informatics*, 77:34–49.

Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. Opentag: Open attribute value extraction from product profiles. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1049–1058.

A Additional Details on Experimental Setup

Software Versions

Experiments were conducted using Python 3.10.12 (main, Nov 20 2023, 15:14:05) compiled with GCC 11.4.0. Table 5 lists the versions of key libraries used in our experiments.

Library	Version
transformers	4.51.3
PyYAML	6.0.1
statsmodels	0.14.2
scipy	1.13.1
numpy	1.26.4
json	Standard Library (Python 3.10)
xml	Standard Library (Python 3.10)

Table 5: Versions of software and libraries used in the experiments.

Model Configuration

All models were queried using the HuggingFace pipeline interface with parameters listed in Table 6. Generation was deterministic and capped at 8192 tokens. For consistency across models, the "thinking" mode was disabled for Qwen models.

Parameter	Value
`max_new_tokens`	8192
`do_sample`	False
`top_p`	None
`temperature`	None

Table 6: Model generation parameters used in all decoding runs.

Regular Expressions

If initial parsing failed, we attempted to extract structured content from fenced code blocks using regular expressions. Table 7 summarizes the patterns used for each format.

Prompts

For open-ended attribute-value extraction, we used format-specific prompts that instructed the model to generate structured data in either JSON, YAML, or XML. Each prompt asked the model to produce a valid, well-structured output using the appropriate syntax and meaningful field names. Additionally, models were explicitly instructed to use proper serialization fences to support regex-based extraction.

The general prompt template is shown below, where `<FORMAT>` is replaced with the target format (`JSON`, `YAML`, or `XML`):

Format	Regex Pattern	Description
JSON	“‘(?:json)?\s *\n (.*?)“‘	Matches a fenced code block optionally labeled as `json`. Extracts everything between the triple backticks.
YAML	“‘(?:yaml|yml)?\s *\n (.*?)“‘	Matches a fenced code block optionally labeled as `yaml` or `yml`. Captures the inner content.
XML	“‘(?:xml)?\s *\n (.*?)“‘	Matches a fenced code block optionally labeled as `xml`. Content inside is captured for parsing.

Table 7: Regular expressions used to extract structured content from fenced code blocks.

Open Extraction Prompt

Given the following document: \n <*document text*>. Extract all data in <FORMAT> format. Make sure that the <FORMAT> document is valid, provide reasonably detailed names for fields.
Make a proper fence for <FORMAT> so that it can be extracted from the response with a regular expression.

For targeted extraction scenario, we used prompts that explicitly instructed the model to extract specific categories: demographics, medications, or symptoms, in a specified structured format. Prompts were adjusted dynamically based on both the target concept and the desired output format (JSON, YAML, or XML). If no relevant information was found, the model was instructed to return an empty object.

The generalized prompt template is shown below, where <CONCEPT> refers to the target category (e.g., “patient demographics” or “medications”) and <FORMAT> specifies the output format.

Targeted Extraction Prompt

Given the following document: \n <*document text*>. Extract all mentioned <CONCEPT> from the text below in valid <FORMAT> format. If no <CONCEPT> are found, return an empty <FORMAT> object. Make sure that the <FORMAT> document is valid, provide reasonably detailed names for fields.
Make a proper fence for <FORMAT> so that it can be extracted from the response with a regular expression.

B Additional Details on Error Analysis

B.1 Extraction-Related Errors

Extraction-related errors arise when neither direct parsing nor regular expression matching succeeds in recovering a structured object from the model output. Initially, we attempt to parse the output as-is, assuming the model produces a complete structured object without serialization fences; if that fails, we apply format-specific regular expressions to extract fenced content (Appendix A). These errors predominantly stem from infinite repetitions in the generated text. Table 8 summarizes the extraction-related failures across all formats. Notably, `Phi-4` was the only model that consistently avoided these failures.

Format	Total Cases	Infinite Repetitions	Broken Fence (Non-repetitive)
JSON	31	31	0
XML	78	78	0
YAML	112	109	3

Table 8: Summary of extraction-related failures due to regular expression mismatches.

The repetition block length varied, ranging from short fragments such as:

```
"Hepatic dysfunction",
"Hepatic dysfunction",
"Hepatic dysfunction",
"Hepatic dysfunction",
"Hepatic dysfunction"
```

to much longer blocks like:

```
"shortness of breath or respiratory distress (not explicitly
stated but implied by SpO2: 100%)",
"chest pain or discomfort (not explicitly stated but implied
by clear lungs on CXR)",
"fever or chills (not explicitly stated but implied by WBC:
12.4 and 13.8)",
"abdominal pain or discomfort (epigastric region)",
"nausea or vomiting (not explicitly stated but implied by
NPO status)",
"abdominal distension (nondistended)",
"abdominal tenderness (TTP in all quadrants)",
"abdominal guarding (voluntary guarding)",
"abdominal masses or organomegaly (not explicitly stated
but implied by TTP in all quadrants)",
"shortness of breath or respiratory distress (not explicitly
stated but implied by SpO2: 100%)",
"chest pain or discomfort
```

B.2 Malformed Output Errors

Malformed output errors occur when the internal content of a model’s generation is structurally invalid, resulting in failed parsing despite the successful extraction of the object.

Because these issues are tightly coupled to the specific requirements of each format, we analyze them separately for JSON, XML, and YAML.

Table 9 summarizes the most common sources of malformed JSON, including unquoted values, missing delimiters, improperly structured lists, and misnested objects. Many of these errors stem from the model emitting raw numerical data, units, or complex expressions without enclosing them in quotes.

Table 10 highlights XML-specific issues such as invalid tag names, unescaped reserved characters (e.g., &, <), and improper tag nesting. Additional problems arise when tags encode entire phrases or when outputs terminate prematurely, leaving the structure incomplete.

Table 11 details YAML parsing failures, which are frequently caused by incorrect use of aliases, inconsistent indentation, missing colons, or unescaped colons within long strings. YAML is particularly sensitive to formatting errors, making minor deviations from proper structure likely to result in failure.

Category	Description	Example
Unquoted numeric values	Common vitals (e.g., 128/68, 96%) were emitted without quotes, causing syntax errors.	"blood_pressure": 128/68,
Unquoted units or ranges	Values with units (300mg, 20-60cc/hr) appeared as raw text.	"dose": 300mg,
Improper list or array formatting	Lists with non-JSON-safe elements (e.g., slashed values) were incorrectly serialized.	"BP": [121/63, 75],
String concatenation or unescaped expressions	Attempted concatenation or strings with internal quotes broke JSON structure.	"Range": "10 - 20" + " insp/min",
Missing delimiters	Adjacent fields were emitted without commas.	"hematocrit": 37.3 "platelets": 126 K,

Standalone strings	Free text like "Levofloxacin" appeared without a key, resembling list items.	"medications": {
		"Levofloxacin"
		}

Multiple top-level objects	More than one top-level JSON object or extraneous content after the main object.	{
		"History": "..."
		}
		{
		"PMH": {...}
		}

Unescaped control characters	Strings included invalid characters or unmatched quotes.	"date": "s/p lobectomy '[**33**]'

Table 9: Summary of prevalent JSON formatting errors in model outputs.

Category	Description	Example
Invalid tag names	Tags contain digits, punctuation, or special characters, violating XML naming rules.	<123_BP>120/80</123_BP>
Unescaped characters	Raw XML-reserved characters (<, >, &) appear unescaped in text content.	<symptom>nausea & vomiting</symptom>
Mismatched or misnested tags	Opening and closing tags are misaligned or improperly nested.	<heart><rate>88</heart></rate>
Improper structural nesting	Structural templates are reused in invalid contexts or nested inconsistently.	<24_hour_events><note>...</24_hour_events></note>
Free-text as tag name	Sentence-length strings or clinical statements are incorrectly placed as tag names.	<Patient is alert and oriented>yes</Patient is alert and oriented>

Table 10: Summary of prevalent XML formatting errors in model outputs.

Category	Description	Example
Alias misinterpretation	Placeholders in `[**...**]` format are misinterpreted as YAML aliases, which require alphanumeric characters.	`attending_md: [**Doctor Last Name**] [**Doctor First Name**] C.`
Invalid nested mappings	Multiple colons in a single line without proper quoting create ambiguous mappings.	`- Cardiovascular: (S1: Normal), (S2: Normal)`
Improper scalar values	Misuse of block scalars (e.g., >) or unescaped strings leads to format violations.	`- SpO2: >95\%`
Unclosed or broken blocks	Incomplete sequences or mappings with missing indentation or block terminators.	`- Fentanyl: "2192-9-17" 08:10 AM`
Malformed collections	Lists with poor indentation or unexpected formatting cannot be resolved by the parser.	`- "not feeling well" (1 day prior to admission)`
Improper question mark usage	Use of ? outside mapping syntax breaks YAML interpretation.	`?look into the suprapubic area.`
Unescaped strings with colons	Long unquoted strings containing multiple colons (e.g., copied EHR text) are misparsed.	`title: Chief Complaint: respiratory failure, PEA arrest`

Table 11: Summary of prevalent YAML formatting errors in model-generated outputs.