Improving structured details extraction from short text passages with LLMs

Scott Salisbury
salisbury.100
The Ohio State University
Columbus, Ohio, USA

Kaveen Hiniduma
hiniduma.1
The Ohio State University
Columbus, Ohio, USA

Rumali Perera
perera.62
The Ohio State University
Columbus, Ohio, USA

Mengfei ('Matt') Li
li.12527
The Ohio State University
Columbus, Ohio, USA

ABSTRACT

The rapid advancement of large language models (LLMs) has enhanced their capability to process and analyze free-text data. However, consistently extracting structured information from short text passages remains challenging, particularly for schema-driven data extraction applications. This project investigates methodologies to improve the reliability and accuracy of structured data extraction from short text passages using advanced LLMs. By employing a robust pipeline integrating schema-based data synthesis, multi-model validation, and iterative refinement, we developed a comprehensive dataset for evaluating LLMs across diverse scenarios. Evaluation focused on metrics such as correct inclusion rates and hallucination rates in order to judge the relative strength of different models and the impact of techniques like few-shot prompting and Chain-of-Thought (CoT) reasoning. Results revealed that CoT reasoning significantly reduces hallucination rates and enhances overall performance. Few-shot prompting further improved accuracy, demonstrating the positive influence of in-context learning. This work provides a scalable framework for bridging the gap between unstructured text and structured data requirements, with applications in legal analytics, customer support, and beyond. Detailed evaluation results and performance breakdowns are publicly available to support further research and transparency [1].

ACM Reference Format:
Scott Salisbury, Kaveen Hiniduma, Rumali Perera, and Mengfei ('Matt') Li. 2024. Improving structured details extraction from short text passages with LLMs. In *Proceedings of (CSE 5525 Final Project Report).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The rapid advancement of large language models (LLMs) has transformed the ability to process and analyze free-text data across various domains. Despite their capabilities, consistently and reliably extracting structured information from short text passages remains a significant challenge. This task is critical for enabling nuanced indexing, metadata enrichment, and streamlined workflows in applications that require complex data interpretation and schema-driven structuring. One major application of structured extraction is enriching datasets. For instance, legal contracts often include key information such as governing laws, signing dates, and parties involved. Extracting these details into structured schemas can enable advanced analytics, search, and operational improvements. Similarly, structured metadata can enhance datasets of free-text documents in various fields, supporting sophisticated indexing, categorization, and query responses. Beyond dataset enrichment, improved structured extraction can streamline operations in real-time scenarios. In customer support, for example, LLMs can be used to identify which product a ticket refers to, the operating system associated with the issue, and other contextual details. These capabilities enhance the prioritization and routing of requests and facilitate faster resolutions and better customer experiences. This project aims to improve the reliability and accuracy of structured detail extraction from short text passages using LLMs. By designing methods that align with predefined schemas, the project seeks to bridge the gap between unstructured text and structured data requirements, empowering organizations to better leverage the information embedded in free-text formats.

2 METHODOLOGY

To validate the effectiveness of structured data extraction from short text passages, we implemented a multi-phase process involving iterative scenario design, data generation, and model evaluation. Initially, we used advanced non-OpenAI/Meta models, such as Claude 3.5 Sonnet and Gemini 1.5 Pro, to develop a diverse range of scenarios where structured data extraction could provide significant utility. For each scenario, we generated a JSON schema, tailored to represent structured information across varied contexts, and populated these schemas with multiple diverse JSON objects that incorporated optional fields at varying completion levels. These objects served as the foundation for generating corresponding text passages designed to encapsulate the schema information while introducing contextually-realistic but schema-irrelevant details. A rigorous verification and iteration process ensured the quality of generated text and schema alignment, providing a robust dataset.

CSE 5525 Final Project Report, Fall 2024,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

We used this dataset to evaluate GPT and LLaMA models under different x-shot prompting settings (from zero-shot to few-shot and many-shot) and with Chain-of-Thought 'reasoning' enabled or disabled. This evaluation enabled us to analyze model performance across various prompting strategies, record summary statistics, and identify areas for iterative improvement.

2.1 Dataset Creation Process

There were of course no existing labeled datasets for this purpose, so we needed to create one. Manually assembling such a dataset would've been prohibitively time-consuming, and it would've been difficult to maintain quality when doing such mind-numbingly tedious work. We used two frontier LLM's (Gemini 1.5 Pro 002 and Claude 3.5 Sonnet (New)) to assist with brainstorming of use-cases/JSON-schemas and to mostly automate the creation of the dataset. Including two different frontier LLM's made the data more diverse and also enabled a more robust automated-validation stage in the data synthesis pipeline.

2.1.1 Use Case Brainstorming and Schema Creation. We went through several rounds of coming up with use cases, iterating on the prompts we used with frontier LLM's (e.g. using good previous outputs for few-shot prompting), discussing what we were looking for in these use cases and associated schemas.

While we initially intended to focus on scenarios with 1-4 sentence text passages (and would evaluate SLM's like GPT-4o-mini or Llama-3.1-8B on the dataset), we found that it was difficult to find enough diverse and plausible use cases that fit such a tight length constraint. As a result, we changed scope fairly early on to use cases involving structured detail extraction from 1-4 paragraph text passages and to primarily focus on larger LLM's like GPT-4o and LLama 3.1 405B.

While some of the generated use cases and associated schemas that we kept were fully aligned with our purposes, we also manually edited details in a large minority of the use case descriptions and/or schemas where that seemed more straightforward than additional rounds of LLM prompting.

2.1.2 Data Synthesis Pipeline. The goal for this project's synthetic data pipeline was ensuring that a given text passage (an input to be used for testing models on this task) had all of the pieces of information in its associated JSON object (the expected output to be used when testing models on this task) and that that text passage had no other pieces of information that would fit within the current scenario's schema.

We maximized the odds of the former desideratum by first having an LLM generate a JSON object containing plausible values for the current use case and then separately prompting the LLM to create a text passage that was realistic for that use case and contained the information from that JSON object.

The key trick was that we then had the other LLM that was being used for data synthesis (but which hadn't produced the current text passage and expected JSON object pair) try to extract information from that text passage according to the current use case's schema. We then used custom logic for comparing two JSON objects (with many safeguards against identifying a mismatch that wasn't really there) to both check for the second model not being able to find expected pieces of information in the text passage and for the second model finding pieces of schema-relevant information in the text passage that weren't expected. This involvement of the second model (trained by a different company) reduces the likelihood that an extra (not in the JSON object) schema-relevant piece of information would be added to the text passage without being noticed (e.g. because some model had an idiosyncratic flaw related to certain topics or types of details that caused it to both be more likely to add such a detail into a text passage despite instructions and less likely to extract that schema-relevant detail from a text passage, again despite instructions).

In practice (and averaging across the 15 use cases), a large majority of generated JSON-object/text-passage pairs passed the above test (that used the other frontier LLM plus old-fashioned rules-based logic) without issue. Meanwhile, the data synthesis pipeline collected extensive information about the cases where a pair failed that validation (including the Chain of Thought analysis from the original JSON object creation, the CoT analysis from the text passage creation, the CoT analysis from the other LLM's attempt to reconstruct the object from the text passage, and the list of problems that the JSON comparison logic identified) and stored all information about such cases in a report file for later human review, where they could be manually corrected and then included in the dataset. The CoT outputs were collected and stored to make human error analysis more convenient and allow more informed changes of the JSON-object and text-passage generation prompts.

The creation of this pipeline involved a number of experiments and revisions. A couple rounds of initial experiments with prompt engineering were done by manually prompting the models in their providers' API playgrounds. One contributor then implemented a script for having one LLM's text passages be reconstructed into objects by the other LLM, and another contributor came up with and implemented large parts of the JSON-object-comparison logic. The former contributor then finished the script logic for the whole pipeline (from generating batches of JSON objects and making a text passage for each one to having another LLM extract information from a given text passage and using the aforementioned JSON-object-comparison logic to confirm whether the current JSON-object/text-passage pair could safely be saved as part of the dataset) while dramatically expanding on the object-comparison logic to reduce incorrect flagging of mismatches and to produce more nuanced evaluation metrics. This was all done with just 2-3 use cases to allow faster and cheaper iterations.

Once the basic pipeline was working, it was enhanced to be more robust in many ways, e.g. automatically re-prompting the LLM if it generated a JSON object that didn't even conform to the schema, using 2-shot prompting in all system prompts, making later runs of the pipeline build on rather than overwrite the outputs of earlier runs, and automatically retrying API calls that failed due to temporarily-overloaded servers or overzealous content-filtering.

Once the pipeline was ready, it was used to produce several hundred records (object-passage pairs) across the 15 use cases.

We also updated a few details of the schemas and datasets (and scripts) when we identified problems later on. For example, we realized that the 'correct' value for one field in one use-case's schema would change every year, so that field was removed. We also realized that a handful of fields in 3 use-cases's schemas needed to

be flagged to the LLM as areas where the text should be faithfully copied in every detail rather than being paraphrased or summarized (e.g. one wouldn't trust an LLM tasked with this detail extraction work to reliably summarize the nuances and implications of the termination clause of a contract, and also such inexact extraction behavior made automated validation very unreliable for those use-cases' records).

Because Claude (Anthropic) and Gemini (Google DeepMind) models were involved in the inclusion of every single record in the dataset (either in the generation step or the validation/filtering step), it was only appropriate to use this dataset for evaluating other companies' models on this task of extracting details from free-text according to a specified structure.

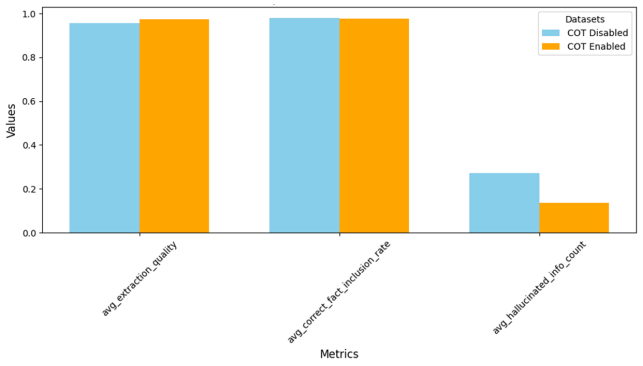

Figure 1: Comparison of Overall Metrics when CoT enabled/disabled

3 EVALUATING LLMS ON STRUCTURED DETAIL EXTRACTION

3.1 Data Splitting

The dataset splits were generated by first consolidating text passages and corresponding JSON objects produced by two chatbot models, Claude and Gemini, for multiple scenarios. Each scenario was associated with a schema that defined the structure of the extracted information. The combined dataset was then shuffled and divided into three parts: a reserved split of examples for few-shot/many-shot prompting, a validation set, and a test set. Few-shot examples were selected first, followed by allocating a fixed proportion of the remaining data to validation and test sets.

3.2 Evaluation Metrics: Extraction Quality, Inclusion Rate, Hallucination Rate, and Retries Used

The extraction quality was assessed through a detailed framework that considered both the accuracy of correctly extracted information and the occurrence of hallucinations—extraneous or incorrect details not relevant to the original schema. The impact of hallucinations was scaled relative to the expected number of information pieces. For instance, in scenarios expecting many pieces of information, a few hallucinations were less detrimental than in scenarios expecting only a small number of details.

A harmonic mean was used to balance the correct inclusion rate with the complement of the hallucination rate. This approach allowed the evaluation to reward extractions with high precision and low hallucination rates, while heavily penalizing outputs with high hallucination rates, regardless of the correct inclusion rate. In cases where hallucinations outnumbered the expected pieces of information, the extraction was deemed unreliable, as any given piece of extracted data was at least as likely to be incorrect as correct. Conversely, low hallucination rates were rewarded, especially if the correct inclusion rate was further from perfect accuracy than the hallucination ratio was from zero.

Also, because the evaluation mechanism automatically re-queries models when they produce invalid or non-schema-conforming JSON objects, the evaluation also tracks the number of retries that the model had to use before it produced a possibly-correct output.

3.3 Evaluations

The evaluation process in this system involves assessing model outputs based on their extraction quality, factual accuracy, and the presence of hallucinated information. It uses the above set of metrics to gauge the performance of models, comparing actual outputs to expected results. This process evaluates models in zero-shot, few-shot, and meny-shot settings, randomly sampling examples from the 'few-shot' split of the dataset for the few-shot and many-shot prompts. It also evaluates models with or without chain-of-thought (CoT) reasoning (i.e. "without CoT" means the json object must be the entire model output). These results are grouped and aggregated to provide an overall performance summary for each model and evaluation scenario. Additionally, the system aggregates individual evaluations' records and generates reports that summarize performance across different evaluation models, scenarios, and configurations.

The models we evaluated on the task of extracting the structured data were: gpt-4o-2024-11-20, gpt-4o-mini-2024-07-18, Llama-3.3-70B and Llama-3.1-405B.

Below are the plots illustrating the overall performance of the models across various configurations, including settings with Chain-of-Thought (CoT) enabled/disabled, different few-shot configurations (ranging from 0 to 50) and overall performance of the models per scenario. Additional evaluation reports such as per model evaluations with CoT enabled/disabled can be accessed via our project's GitHub repository [1].

As anticipated, Figure 1 demonstrates a clear improvement in metrics when CoT is enabled compared to when it is disabled, which aligns with expectations. For instance, hallucinations are reduced by nearly 50% with CoT enabled.

Figure 2 illustrates that enabling CoT generally enhances performance across almost all scenarios for overall extraction quality and for hallucination rate while having a very mixed (mostly negative) effect on the correct fact inclusion rate. This slight (0.46 percentage point) decline could be attributed to the additional complexity of the reasoning steps in CoT (i.e. perhaps the brittle LLM 'reasoning' overcomplicates things or otherwise 'goes off the rails' some of the time).

Figure 3 and Figure 4 demonstrates that, as anticipated, performance across metrics improves with an increase in the number of

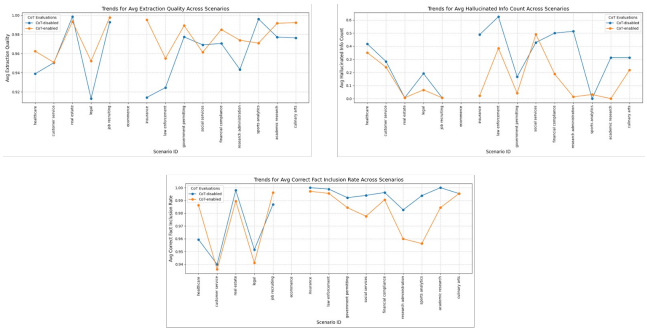

Figure 2: Evaluation metric trends based on scenarios

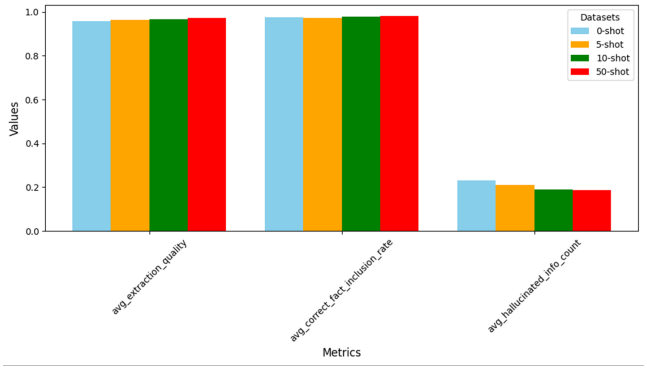

Figure 3: Comparison of overall metrics for different few shot settings

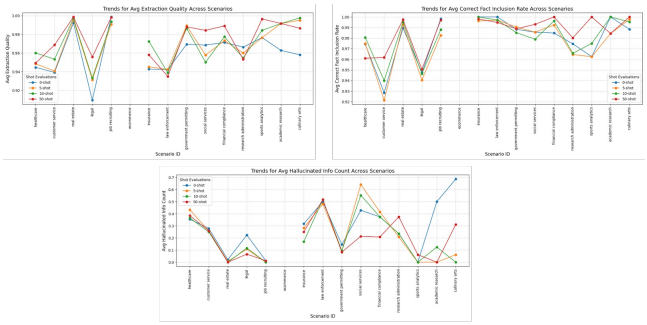

Figure 4 : Evaluation metric trends based on scenarios for few shot settings

examples provided for prompting. The average count of hallucinated information shows a modest but noticeable and relatively consistent decrease as more examples were introduced. Meanwhile, the correct fact inclusion rate actually decreases slightly with 5 few-shot examples but improves (relative to zero-shot) in the large-few-shot (10 examples) and many-shot (50 examples) cases.

Finally, we found that larger and more widely-well-regarded models did better at this. At the same time, it is somewhat surprising how close to reliable even the SLM GPT-4o-mini is at this task.

4 FUTURE WORK

First of all, given how well GPT-4o-mini did, it would be interesting to see how well an even smaller model like Gemma 2-9B or Llama 3.1-8B would do on this task. It's also an intriguing open question whether the additional RL training and CoT-style preliminary output of OpenAI's o1-mini model would allow it to outperform the nearly-as-expensive GPT-4o on this task. Once o1 is made available on the API, it would be interesting for certain purposes (i.e. cases where the tolerance for errors is extremely low and cost-management is less critical) to see just how low its oversight and hallucination rates are on this task relative to 'GPT-4-level' models.

We could also extend the data pipeline:

- Evaluate whether self-review with self-consistency at the json-object-generation stage and again at the text passage generation stage improves the quality of generated records.
  - Self-review with self-consistency might involve 3-5 additional API calls per json object or text passage to scrutinize it for rule-following and plausibility, with the object or passage being regenerated if a majority of the self-review API calls judged it to not be following all instructions fully.
  - If frontier models can actually associate points of criticism with locations in the JSON object (i.e. JSONPath strings) or text passage (e.g. sentence numbers) reliably enough, we might even be able to include feedback in the regeneration prompt (i.e. feedback from critic LLM responses) based on which pieces of feedback referred to sections of the thing being critiqued that a majority of critic responses took issue with.
- Make multiple reconstruction attempts with nonzero temperature (in the new-data-validation stage) to reduce the risk that imperfect records are saved as part of the dataset because the validating LLM happened to miss the extra schema-relevant detail some of the time.

Three refinements would improve both the convenience of the automated data synthesis pipeline and the reliability, usefulness, or practical application of the evaluation benchmark (i.e. augmentations that allow models to get higher scores on the benchmark might make this technique more useful in practical settings):

- Explore the addition of SLM API calls for semantic similarity checks in the string-vs-string part of the JSON-object-comparison code. This could reduce the rate of records being falsely flagged as bad. However, it raises tricky questions about how to define rules for different strings being truly equivalent (which might be context-dependent) or where to set the threshold for a similarity score.
- Add a self-consistency component to the structured-extraction-from-text-passage code. This could involve making multiple (3, 5, or even 10) queries per text passage and only including a given piece of information in the final result object if it was found in a majority of the queries' outputs.
- Add a self-review with self-consistency component (as described above) to the structured-extraction-from-text-passage code.

We could also dramatically scale up dataset size (to 3-5k records) and change the dataset split to fewshot/train/validation/test. Once

Metric	GPT-4o	GPT-4o-mini	Llama 3.1 405B	Llama 3.3 70B
Extraction Quality	0.9826	0.9479	0.9711	0.9598
Correct Fact Inclusion Rate	0.9867	0.9664	0.9773	0.9789
Hallucination Rate	0.105	0.277	0.156	0.278
Retry Usage	0.014	0.046	0.035	0.051

Table 1: Model Comparison

that was done, we could attempt SFT or even RL with various models to see how much better they could be made at this task.

5 CONCLUSION

In conclusion, this project demonstrates the potential of large language models to reliably extract structured details from short text passages, bridging the gap between unstructured text and schema-driven data requirements. By employing a rigorous methodology that integrates schema-based data synthesis, cross-model validation, and iterative refinement, we successfully created a robust dataset for evaluating LLMs across diverse use cases. The evaluation process highlighted the strengths and limitations of leading LLMs, including GPT and LLaMA, in their ability to align text with predefined schemas under different prompting strategies and x-shot settings. The evaluation framework, which balances the recall of relevant facts against hallucinations, provides insights into model performance. This work lays the foundation for advancing LLM-based structured data extraction, enabling applications in fields ranging from legal analytics to customer support. Additionally, detailed per-model (GPT and Llama) evaluation reports, including metrics and performance breakdowns, are available in our GitHub repository [1] for further review and transparency.

REFERENCES

[1] BareBeaverBat. 2024. CSE5525 Structured Data Extraction. https://github.com/BareBeaverBat/CSE5525_Structured_Data_Extraction/tree/main Accessed: 2024-12-10.