Article

Not peer-reviewed version

Prompt Engineering for Structured Data A Comparative Evaluation of Styles and LLM Performance

Jules White , Ashraf Elnashar * , Douglas Schmidt

Posted Date: 24 June 2025

doi: 10.20944/preprints202506.1937.v1

Keywords: Structured Data Generation; LLMs; Prompt Engineering; JSON; YAML; CSV Formats; Token Efficiency; Data Validation; Cost-Effective AI

Article

Prompt Engineering for Structured Data A Comparative Evaluation of Styles and LLM Performance

Ashraf Elnashar [1], Jules White [1,*] and Douglas C. Schmidt [2]

[1] Department of Computer Science, Vanderbilt University, Nashville, TN, USA
[2] Department of Computer Science, William & Mary, Williamsburg, VA, USA
* Correspondence: ashraf.elnashar@vanderbilt.edu

Abstract

The effectiveness of prompt engineering strategies for structured data generation remains an open challenge, especially as the capabilities and architectures of large language models (LLMs) continue to evolve. While prior research has examined a limited set of prompt styles using GPT-4o [8], this study expands the scope significantly by comparing six prompt styles—JSON, YAML, CSV, function calling APIs, simple prefixes, and a hybrid CSV/prefix format—across three state-of-the-art LLMs: ChatGPT-4o, Claude, and Gemini. Building upon our earlier findings, which focused on evaluating three prompt styles within GPT-4o, this study introduces a broader prompt set and performs a comparative analysis across multiple models to generalize and extend prior conclusions. These datasets are used to evaluate each prompt style for these LLMs across three critical metrics: accuracy in reproducing the expected data attributes, token cost for API usage, and time needed to generate data. Our methodology incorporates structured data validation and analysis through Python utilities that ensure precise comparison and document each style's performance. We visualize the results via Technique vs. Accuracy, Technique vs. Token Cost, and Technique vs. Time graphs. Our results reveal trade-offs between prompt complexity and performance, suggesting that simpler formats may provide efficiency benefits with minimal loss in accuracy, while more flexible formats offer enhanced versatility for handling complex data structures. Our extended findings demonstrate that prompt selection substantially affects both quality and resource efficiency. Claude consistently yields the highest accuracy, ChatGPT-4o is the most token- and time-efficient, and Gemini offers a balanced trade-off across metrics. These results extend earlier single-model evaluations and provide practical guidelines for choosing prompt styles based on model capabilities and application-specific constraints. This work advances the field of prompt engineering by offering a comprehensive, multi-model framework for optimizing structured data generation.

Keywords: structured data generation; LLMs; prompt engineering; JSON; YAML; CSV formats; token efficiency data validation; cost-effective AI

---

1. Introduction

Structured data extraction remains a persistent challenge in LLM applications. Modern LLMs show remarkable capabilities in generating structured outputs from unstructured text, offering transformative potential in domains like business intelligence, healthcare, and e-commerce [7,17]. Yet, the quality and efficiency of structured data generation depend heavily on how prompts are designed. Prompt styles vary in structure, verbosity, and interpretability—factors that significantly influence the output's accuracy, token cost, and generation time [16]. Our earlier study explored these dynamics using GPT-4o and three prompt styles. This paper builds directly on our prior work, which evaluated three prompt styles—JSON, YAML, and Hybrid CSV/Prefix—using GPT-4o alone for structured data generation tasks [8]. In this extended study, we significantly broaden both the prompt design space

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2025 doi:10.20944/preprints202506.1937.v1

and the model spectrum, introducing a multi-model comparison framework to generalize and refine earlier insights. This work broadens that investigation across more models and formats to better understand generalizable strategies for prompt-based structured data generation.

While prior studies have explored these differences, there are no comprehensive, cross-model analyzes. This paper extends our previous work by broadening both the range of prompt formats and the diversity of LLMs evaluated. Specifically, we examine six prompt styles—JSON, YAML, CSV, function calling APIs, simple prefixes, and a hybrid CSV/prefix format—across three state-of-the-art models: ChatGPT-4o, Claude, and Gemini. This multi-model, multi-format evaluation establishes a comprehensive and generalizable framework for codifying how prompt design affects the accuracy, efficiency, and cost of structured data generation.

Prompting beyond hierarchy: Exploring diverse representations. Hierarchical formats like JSON and YAML are commonly used for their structural rigor and compatibility with downstream systems. However, they can be verbose and computationally expensive, especially when used with large or nested datasets. In contrast, simpler formats—such as CSV and prefix-based prompts—offer more compact representations that are faster and cheaper to process, albeit with potential trade-offs in structure and semantic clarity. Hybrid approaches, such as CSV with prefixed rows, attempt to bridge this gap by retaining structural cues in a tabular format. Moreover, function-calling APIs, now natively supported in many LLM platforms, provide a formal schema-driven interface that enables strict data validation and integration with programmatic workflows.

Despite the potential of structured data generation [19] with LLMs, there is limited guidance on the optimal prompt styles for achieving high-quality outputs with leading LLMs. Many practitioners prompt via JSON or API function calls due to their structured format and familiarity. These prompt styles, while widely adopted, may not be optimal in terms of token usage or processing time—especially when dealing with complex and/or large datasets. Alternative prompting techniques, such as simple prefixes, CSV, YAML, or hybrid formats, may offer similar or even better results that consume fewer resources.

Our approach → Evaluating the impact of prompt styles on structured data generation. This paper systematically evaluates six prompting strategies—ranging from hierarchical formats (JSON, YAML) to lightweight representations (CSV, simple prefixes), as well as function calling APIss and hybrid approaches—across three leading LLMs: ChatGPT-4o, Claude, and Gemini. We assess the performance of these six prompting styles across three metrics (accuracy, token cost, and generation time), comparing results across the selected LLMs. Our results quantify the efficiency of each technique and establish practical recommendations for prompt design, particularly where token usage and response time are key considerations.

This paper provides the following contributions to research on optimized prompt techniques for efficient and accurate structured data generation using LLMs:

- We designed an experiment that generates randomized datasets in three distinct scenarios (personal stories, medical records, and receipts) to evaluate how effectively each prompt style captures the required structure across diverse and context-specific scenarios. Datasets for personal stories contain attributes representing individual characteristics and paired them with corresponding valid narratives. Datasets for medical records contain medical attributes to produce valid and realistic entries. Datasets for receipts contain attributes to create valid examples reflecting real-world purchases, following the dataset construction methodology from our previous study [8].
- We used these three datasets to compare the outputs generated by leading LLMs (ChatGPT-4o, Claude, and Gemini) with the expected results. Accuracy was measured based on strict adherence to the original data attributes and values, ensuring the generated structured data matched the intended formats, including JSON, YAML, and CSV.
- We developed an automated validation framework to measure output fidelity, token consumption, and generation latency. These metrics are visualized through comparative graphs—*Technique vs. Accuracy*, *Technique vs. Token Cost*, and *Technique vs. Time*—highlighting the trade-offs across prompt styles and LLMs.

 doi:10.20944/preprints202506.1937.v1

Paper organization. The remainder of this paper is organized as follows: Section 2 summarizes the open research questions addressed in our study and outlines our technical approach; Section 3 explains our experiment design, datasets, and testbed environment; Section 4 analyzes the results of experiments that evaluate the efficiency, accuracy, and cost-effectiveness of different prompting strategies for structured data generation using LLMs; Section 5 provides a comparative analysis of the performance of ChatGPT-4o, Claude, and Gemini on our datasets; Section 6 compares our research with related work; and Section 7 presents lessons learned from our study and outlines future work.

2. Summary of Research Questions

Four research questions guide our study by evaluating the effectiveness of different prompt styles on structured data generation by leading LLMs. Each question addressed a specific aspect of prompt performance, focusing on accuracy, efficiency, and cost-effectiveness, as follows.

These questions build upon the research framework introduced in our earlier study using GPT-4o [8], which focused on prompt-style evaluation for structured data generation. In this paper, we extend those questions across a broader range of prompt styles and evaluate them using multiple state-of-the-art LLMs to gain deeper, more generalizable insights.

Q1: Which prompt style for each LLM produces the most accurate structured data outputs in terms of attribute completeness and correctness? This question investigated the accuracy of each prompt style by assessing each LLM's ability to generate structured data that closely matched the expected attributes and values in our test datasets. We measured accuracy by comparing each generated output with a predefined set of attributes, ensuring all required fields were represented correctly. This analysis identified which prompt style(s) best captured the intended structure and values on each LLM we evaluated.

Q2: What is the token cost associated with each prompt style for each LLM and which style offers the most cost-effective data generation? Given that token usage is directly linked to API costs when working with LLMs, this question focused on understanding the token efficiency of each prompt style per LLM. By tracking token consumption for each generated output, we evaluated which prompt style minimized token usage without sacrificing accuracy. This insight is crucial for applications where cost-effectiveness is a priority since lower token counts translate to reduced expenditures for LLMs.

Q3: How does each prompt style perform in terms of generation time and which is the fastest technique for producing structured data per LLM? This question addressed the response time each prompt style needed to generate structured outputs, identifying the prompt style(s) that achieved the best performance for each LLM. We recorded the time taken for each LLM interaction to compare the relative speed of each style. This metric is valuable in scenarios (such as real-time applications or batch processing tasks) where rapid data generation is essential.

Q4: Are there specific scenarios or data types (*e.g.*, personal stories, receipts, or medical records) where one prompt style performs better than others per LLM? Since different data types may impose different requirements on structure and completeness, this question explored whether certain prompt styles were better suited for specific scenarios per LLM. By applying each prompt style to varied datasets, we analyzed if data types impact prompt performance, providing practical guidance for selecting prompt styles based on data type, context, and LLM.

3. Experiment Design

To address the research questions in Section 2, we developed a multi-stage study involving randomized data generation, prompt formulation [12], interactions with multiple LLMs [21], and validation of generated outputs. This study evaluated the efficiency, accuracy, and cost-effectiveness of six prompting styles (JSON, YAML, CSV, API function calls, simple prefixes, and a hybrid CSV/prefix format) to generate structured data across three data contexts (personal stories, receipts, and medical records). To ensure a comprehensive analysis, we performed this study using three leading LLMs (ChatGPT-4o, Claude, and Gemini).

This experiment builds upon our earlier study, which evaluated structured data generation using three prompt styles (JSON, YAML, and Hybrid CSV/Prefix) with GPT-4o alone [8]. In this work, we significantly extend the experiment design by introducing three additional prompt styles (CSV, API function calls, and Simple Prefixes) and comparing them across three leading LLMs. We also reuse and expand upon the randomized datasets and evaluation framework from our prior work to maintain consistency while enabling multi-model analysis. Some figures and validation methodologies presented in this section are adapted from the original study with updated results and formatting.

Systematically assessing the performance of each prompt style per LLM identified optimal formats for structured data generation-based token usage, processing time, and accuracy metrics. Our experiment was structured into two stages shown in Figure 1 and described below, following and expanding the design of our prior GPT-4o-only study [8].

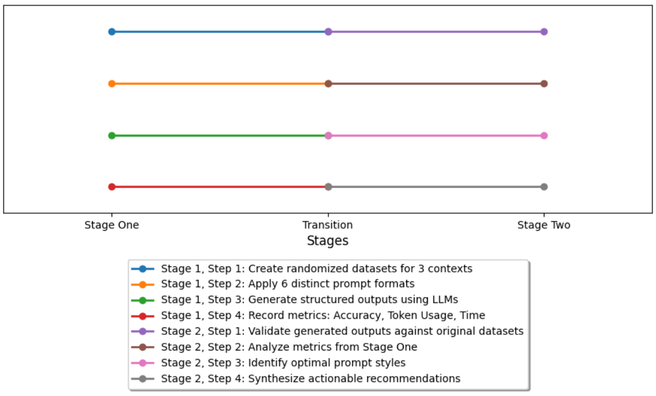

Figure 1. Visualization of Study Stages.

- **Stage One: Data Generation and Prompt Testing**. This first stage created randomized datasets tailored to the three contexts (personal stories, receipts, and medical records) and applied six distinct prompt style (JSON, YAML, CSV, API function calls, Simple Prefixes, and Hybrid CSV/Prefix) to guide the LLMs in generating structured outputs. Three metrics aAccuracy, token usage, and generation time) were recorded for each combination of LLM and prompt style.
- **Stage Two: Assessment and Refinement**. This second stage validated the outputs generated by each LLM against the original datasets to measure accuracy. The metrics collected during Stage One were assessed to identify the most efficient and effective prompt styles. The results were codified into actionable recommendations that highlight the strengths and trade-offs of each prompt style for different data contexts and LLMs.

3.1. Stage One: Data Generation and Prompt Testing

The first stage of our study created and validated diverse datasets that simulate realistic data generation scenarios across three contexts: personal stories, receipts, and medical records. This dataset generation [10] process ensures each context contains a variety of attributes, such as name, age, city, and email, formatted according to specific guidelines. These datasets serve as the foundation for evaluating the effectiveness of each prompt style in Stage Two, as discussed in Section 3.2.

Our dataset generation process began by randomizing attributes to include diversity, such as optional fields like *email*. These attributes were embedded into prompts instructing each LLM to generate a narrative that incorporated all specified details. Each generated story was then validated against the original attributes using pattern-matching to ensure accuracy and completeness.

The validation process ensured every input dictionary attribute was reflected accurately in the generated story. This process extracted relevant paragraphs for each individual attribute using a pattern-matching algorithm and compared each attribute from the input against the generated text to verify inclusion. Any missing attributes were logged for further refinement to ensure generated outputs met the expected requirements of accuracy and completeness.

 doi:10.20944/preprints202506.1937.v1

3.1.1. Personal Stories Dataset and Validation

The *Personal Stories Dataset* evaluates each LLM's ability to generate narrative outputs that faithfully incorporate structured input attributes. This dataset simulates real-world scenarios where structured data is embedded seamlessly within natural language text, while maintaining the integrity of all specified attributes. The generation process followed a systematic method to ensure consistency, accuracy, and validity.

To create this dataset, random attributes were generated for fictional individuals, including fields like *name*, *age*, *city*, and optionally *email*, as shown in Figure 2.

```
{   "people": [ {
      "name": "dkmotp",
      "age": 3
    }, {
      "name": "mlumld",
      "city": "London"
    }, {
      "name": "fanonr",
      "email": "uytgoy@example.com"
    }, {
      "name": "jntsxr",
      "age": 33,
      "city": "New York",
      "email": "xqnmth@example.com"
    } ] }
```

Figure 2. **Random People Generation Example (adapted from [8]).**

These attributes were structured in dictionaries where each entry represents a unique individual. The randomization ensured diversity and tested each LLM's capability to handle variations in input attributes effectively.

Figure 3 shows the LLM prompt used to generate personalized stories. This prompt was designed to ensure each LLM generated a short, single-paragraph narrative that explicitly included every attribute from the input dictionary.

```
## Problem Statement
- Write a short, single-paragraph story
  about the individuals listed below.
## Constraints
- Ensure that every value in the dictionary
  is included exactly as provided.
- The values must be explicitly stated in
  the story so they can be verified using
  ‘.contains({people})‘.
## Input
- ‘people‘: A list of randomly generated
   dictionaries representing individuals.
```

Figure 3. **Prompt Generating Personalized Stories (adapted from [8]).**

Moreover, this prompt enforced the inclusion of all attribute values exactly as provided, enabling rigorous validation of the generated outputs, as shown in Figure 4.

```
## Story
In a twist of fate that only destiny could
orchestrate, dkmotp, a bright-eyed curious
3-year-old, found himself at a bustling
carnival in the heart of London. Just steps
away, mlumld, who had spent her entire life
in this vibrant city, was mesmerized by the
lively atmosphere, unaware her path was about
to cross with others in the strangest of ways.
As she admired the colorful parade, fanonr,
busy typing an email to uytgoy@example.com,
accidentally bumped into jntsxr, a 33-year-
old from New York who was in town to escape
the familiar chaos of his life. He quickly
glanced at his phone to ensure nothing had
happened to his email draft addressed to
xqnmth@example.com. Little did they know,
their lives were about interconnect by the
most unexpected series of events.
```

Figure 4. LLM-generated Personalized Stories (adapted from [8]).

The validated stories were formatted into six distinct styles for further analysis and experimentation: (1) JSON for hierarchical structures with nested attributes, (2) YAML for human-readable formats, (3) CSV for flat tabular data representation, (4) API function calls for encapsulating structured data, (5) Simple prefixes for lightweight labeled fields, and (6) Hybrid CSV/prefix for combining tabular headers with prefixed rows. Section 3.2 assesses each format for accuracy, efficiency, and token usage in subsequent stages of the experiment, providing a comprehensive understanding of their relative strengths and weaknesses.

3.1.2. Medical Record Dataset

The *Medical Record Dataset* evaluates the ability of each LLM to generate structured medical records that accurately represent input attributes. This dataset simulates real-world scenarios where structured patient data is embedded within electronic medical records, while preserving the completeness and correctness of all specified details. The generation process followed a systematic methodology to ensure consistency, accuracy, and validity.

To create the dataset, random attributes were generated for fictional individuals, including fields such as *name*, and optionally *age*, *city*, and *email*. Likewise, medical-specific fields, such as *diagnosis*, *prescriptions*, and *doctor's notes*, were included but are considered outside the scope of this experiment. These attributes were organized into dictionaries, with each entry representing a unique individual. Figure 2 depicts the random person generation process for the *Medical Record Dataset*.

Randomizing attributes ensured diversity and tested each LLM's ability to handle varying input fields effectively. The prompt shown in Figure 5

 doi:10.20944/preprints202506.1937.v1


```
## Problem Statement
Write a structured medical record for each
individual listed below.
## Constraints
- Ensure that every value in the dictionary
  is included  exactly as provided.
- The values must be explicitly stated in
  the record so they can be verified using
  ‘.contains({{people}})‘.
## Input
- ‘people‘: A list of randomly generated
   dictionaries representing individuals.
## Output Format
Write medical records using this format:
## Patient Information: X
- Name: [Name] (required)
  - Age: [Age] (optional)
  - City: [City] (optional)
  - Email: [Email] (optional)
Ensure the records are formatted as shown
above and all names are listed exactly
as provided in the dictionary.
```

Figure 5. Prompt Generating Medical Records.

guided each LLM to generate structured medical records that include every attribute from the input dictionary. This prompt included all specified attributes and ensured the generated medical records followed the required structure and formatting. Figure 6 shows examples of the LLM-generated medical records.

```
## Patient Information: 1
- Name: dkmotp
  - Age: 3
## Patient Information: 2
- Name: mlumld
  - City: London
## Patient Information: 3
- Name: fanonr
  - Email: uytgoy@example.com
## Patient Information: 4
- Name: jntsxr
  - Age: 33
  - City: New York
  - Email: xqnmth@example.com
```

Figure 6. LLM-generated Medical Record.

3.1.3. Receipt Dataset

The *Receipt Dataset* evaluates each LLM's ability to generate structured receipts that represent input attributes accurately. This dataset simulates real-world scenarios where structured information about transactions or individuals is embedded within formal receipt templates, while preserving the completeness and correctness of all specified details. The generation process applied a systematic prompt style to ensure consistency, accuracy, and validity.

To create the dataset, random attributes were generated for fictional individuals, including fields such as *name*, and optionally *age*, *city*, and *email*. These attributes were organized into dictionaries, with each entry representing a unique individual. Figure 2 depicts the random person generation process for the *Receipt Dataset*.

Randomizing attributes ensured diversity and tested each LLM's ability to handle varying input fields effectively. The prompt shown in Figure 7 guided each LLM to generate structured receipts that explicitly included every attribute from the input dictionary.

 doi:10.20944/preprints202506.1937.v1


```
## Problem Statement
- Write a structured receipt for each
  individual listed below.
## Constraints
- Ensure that every value in the dictionary
  is included exactly as provided.
- The values must be explicitly stated in the
  receipt so they can be verified using
  '.contains({{people}})'.
## Input
- 'people': A list of randomly generated
   dictionaries representing individuals.
## Output Format
Write receipts using the following format:
## Receipt for Person: X
  - Name: [Name] (required)
  - Age: [Age] (optional)
  - City: [City] (optional)
  - Email: [Email] (optional)
Ensure all names are listed exactly as shown
in the dictionary and only include optional
fields if they are provided.
```

Figure 7. Prompt Generating Receipt Records.

This prompt enforced the inclusion of all specified attributes and ensured the generated receipts follow the required structure and formatting. Figure 8 shows examples of LLM-generated receipts.

```
## Receipt for Person 1
- Name: dkmotp
  - Age: 3
## Receipt for Person 2
- Name: mlumld
  - City: London
## Receipt for Person 3
- Name: fanonr
  - Email: uytgoy@example.com
## Receipt for Person 4
- Name: jntsxr
  - Age: 33
  - City: New York
  - Email: xqnmth@example.com
```

Figure 8. LLM-generated Receipt Records.

3.2. Stage Two: Assessment and Refinement

The second stage in our study validated and assessed outputs generated by the three LLMs (ChatGPT-4o, Claude, and Gemini) against the datasets described in Section 3.1. This stage measured the accuracy, efficiency, and token cost for each prompt style and LLM combination by building on the metrics collected during Stage One and codifying actionable insights into the performance of each prompt style. Our analysis in Section 4 highlights the strengths, limitations, and trade-offs of all six prompt styles across the *Personal Stories*, *Medical Records*, and *Receipts* datasets.

Stage Two employed six distinct prompt styles (JSON, YAML, CSV, API function calls, Simple Prefixes, and Hybrid CSV/Prefix), each designed to produce structured data in a specific format. Each prompt style was tailored uniquely to represent the data while testing the capabilities of each LLM to generate outputs accurately and efficiently. These prompt styles served as input instructions to each LLM, as described below.

The JSON prompt Figure 9 instructs each LLM to create a structured JSON output adhering to a predefined schema.

 doi:10.20944/preprints202506.1937.v1


```
Create valid JSON output using this schema:
{   "people": [ {
            "name": "<name>",
            "age": <age>,
            "city": "<city>",
            "email": "<email>"
        },
    ... # Repeat for other people in story
    ]
}
If any attributes (name, age, city, email)
are not present for a person in the story,
omit them from that person’s JSON object.
```

Figure 9. Prompt in JSON Format.

This format is hierarchical and suitable for applications requiring nested structures. The YAML prompt shown in Figure 10 emphasizes human-readability while maintaining strict formatting standards, making it a versatile option for both human and machine interpretation.

```
Create valid YAML output using this schema:
  people:
    - name: <name>
    - age: <age>
    - city: <city>
    - email: <email>
    # Repeat for other people in story
If any attributes (name, age, city, email)
aren’t present for a person in the story,
omit that attribute from that YAML object.

**Ensure the YAML output begins with
’‘‘‘yaml’ and ends with ’‘‘‘’.**
```

Figure 10. Prompt in YAML Format.

The CSV prompt shown in Figure 11 represents the data in a flat, tabular format, ensuring simplicity and compatibility with data analysis tools.

```
Create a CSV file based on this schema:
        Name, Age, City, Email

If any attributes (Name, Age, City, Email)
are not present for a person in the story,
leave that field blank in the CSV output.

**Ensure the CSV output begins with
’‘‘‘csv’ and ends with ’‘‘‘’.**
```

Figure 11. Prompt in CSV Format.

The API function call prompt in Figure 12 simulates an API interaction by formatting the data as function arguments in JSON, challenging each LLM to meet schema requirements precisely.

The Simple Prefix prompt shown in Figure 13 generates structured text using labeled fields (*e.g.*, ‘Name:‘, ‘Age:‘), focusing on human-readability and lightweight representation. Finally, the Hybrid CSV/Prefix prompt shown in Figure 14 combines elements of tabular data and prefixed text, requiring each LLM to maintain placeholders for all attributes, even when values are absent.

Each of these six prompts was crafted carefully to evaluate how effectively each LLM interpreted and adhered to the provided instructions. The outputs generated from these prompts were then validated against the three three *Personal Stories*, *Medical Records*, and *Receipts* databases to measure accuracy. Each dataset was generated using prompts that were identical across ChatGPT-4o, Claude, and Gemini to provide a fair comparison by avoiding any LLM-specific tailoring. This approach

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2025 doi:10.20944/preprints202506.1937.v1

ensured our evaluations reflected each LLMs' inherent capabilities without introducing bias from prompt tailoring [6].

```
Use OpenAI’s API to invoke a function named
‘answer‘ with the following schema:
{ "$schema":
  "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
     "people": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
             "name": {
                "type": "string"
             },
            "age": {
                "type": "string"
             },
             ... # Repeat for city & email
          },
          "required": ["name"]
        } } },
    "required": ["people"]
}
For each person in the story, include their
information as an object within the "people"
array. If any attributes (name, age, city,
email) aren’t present for a person in the
story, omit that parameter in their object.

**Ensure the API function call output begins
 with ’‘‘‘json’ and ends with ’‘‘‘’.**
```

Figure 12. Prompt in API Function Call Format.

```
Create structured text output using prefixes
based on this schema where available:
    Name: <name>
    Age: <age>
    City: <city>
    Email: <email>
If any attributes (Name, Age, City, Email)
are not present for a person in the story,
omit that line in the output.

**Ensure each person’s details are
separated in output.**
```

Figure 13. Prompt in Simple Prefix Format.

```
Create structured output using a hybrid of
CSV and simple prefixes based on this
schema,:
        row: name, age, city, email
Ensure the first row includes the header and
each column has a placeholder, even if the
value is not present.

For each person in the story, extract their
name, age, city, and email, and format it
as a CSV string prefixed by 'row'. The first
line must always be "row: name, age, city,
email". For subsequent rows, if any of these
attributes are not present for a person,
leave that field empty while keeping the
comma as a placeholder.

**Ensure the Hybrid CSV/Prefix output begins
with '```hybrid' and ends with '```'.**

The output should retain all columns in the
order specified, even if some values are
missing. Use the information extracted from
this story:
{ In a twist of fate that only destiny could
orchestrate, dkmotp, a bright-eyed curious
3-year-old, found himself at a bustling
carnival in the heart of London. Just steps
away, mlumld, who had spent her entire life
in this vibrant city, was mesmerized by the
lively atmosphere, unaware her path was about
to cross with others in the strangest of ways.
As she admired the colorful parade, fanonr,
busy typing an email to uytgoy@example.com,
accidentally bumped into jntsxr, a 33-year-
old from New York who was in town to escape
the familiar chaos of his life. He quickly
glanced at his phone to ensure nothing had
happened to his email draft addressed to
xqnmth@example.com. Little did they know,
their lives were about interconnect by the
most unexpected series of events.}
```

Figure 14. Prompt in Hybrid CSV Simple Prefix Format.

We evaluated the performance of these six prompt styles systematically by applying them to the three datasets for all three LLMs. Our assessment identified the strengths and weaknesses of each prompt style by comparing LLM-generated outputs against expected data via the following three measures:

- *Accuracy measures*. which calculated the percentage of attributes correctly included in the generated output,
- *Token usage measures*, which evaluated the number of tokens consumed by each prompt style for each LLM, as token efficiency directly correlates with cost,
- *Time efficiency measures*, which computed response times for generating outputs to assess the suitability of each LLM for real-time or batch processing tasks.

Our schema validation [4] process ensured every attribute from the input datasets was reflected accurately in the generated LLM outputs. As discussed in Section 4 below, the findings from Stage Two codified actionable recommendations when selecting the most effective LLM and prompt style combinations for different structured data generation tasks.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2025 doi:10.20944/preprints202506.1937.v1

4. Analysis of Experiment Results

This section compares the performance of ChatGPT-4o, Claude, and Gemini's in accordance with the assessment process described in Section 3.2. This analysis builds upon our previous GPT-4o-focused evaluation [8], which explored the trade-offs among three prompt styles (JSON, YAML, Hybrid CSV/Prefix) using a single-model setup. In this extended study, we broaden the scope significantly by evaluating six prompt styles across three advanced LLMs. While the datasets and visualization strategies (e.g., prompt style vs. accuracy, token cost, and generation time) retain elements from our prior methodology, all results, graphs, and comparisons have been re-executed with the expanded set of prompt styles and LLMs to ensure comprehensive, model-specific insights.

4.1. Analysis of ChatGPT-4o Experiment Results

We first analyze the results of applying ChatGPT-4o across our three *Patient Information*, *Personal Story*, and *Receipt* datasets for each of the three accuracy, token usage, and time efficiency measures.

4.1.1. Accuracy Analysis for ChatGPT-4o

Figure 15 depicts ChatGPT-4o's accuracy for all six prompt styles and three datasets.

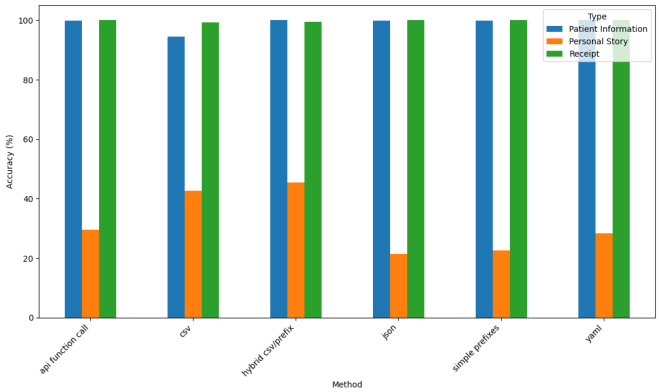

Figure 15. ChatGPT-4o Accuracy by Prompt Style and Type (extended from [8]).

The results indicate ChatGPT-4o achieves consistently high accuracy (nearly 100%) for structured datasets (*i.e.*, *Patient Information* and *Receipt*) across all prompt styles. Its performance declines significantly, however, for the narrative-style *Personal Story* dataset, where its accuracy ranges between 20%-40%.

ChatGPT-4o's disparity in performance highlights its challenges when structuring narrative data compared to well-structured formats. The API Function Call, YAML, and Hybrid CSV/Prefix prompt styles exhibit strong performance, particularly for structured datasets. JSON demonstrates variability in accuracy, however, particularly for *Personal Story* data. These results underscore ChatGPT-4o's strengths in handling structured data formats, while revealing its limitations in unstructured and narrative contexts.

4.1.2. Token Usage Analysis for ChatGPT-4o

Figure 16 shows ChatGPT-4o's token usage for all six prompt styles and three datasets.

 doi:10.20944/preprints202506.1937.v1

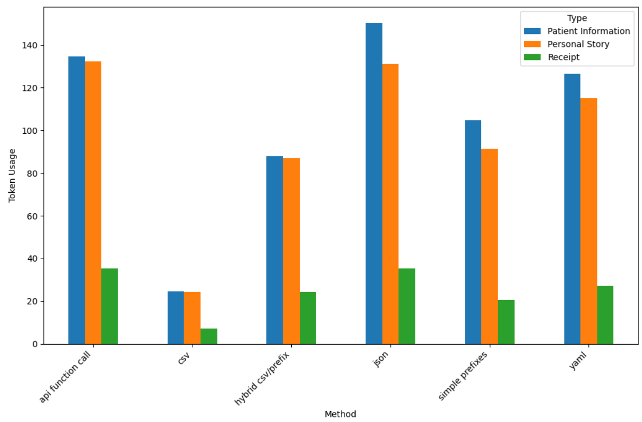

Figure 16. ChatGPT-4o Token Usage by Prompt Style and Type (extended from [8]).

These results show the API Function Call and JSON prompt styles consume the highest number of tokens, particularly for *Patient Information* and *Personal Story* datasets, which generate verbose and structured outputs that increase token usage.

In contrast, the CSV prompt style exhibits the lowest token usage across all dataset types, demonstrating ChatGPT-4o's efficiency in producing concise tabular outputs. Among the dataset types, the *Receipt* dataset consistently shows lower token usage for all prompt styles. This result reflects the reduced complexity and structured nature of ChatGPT-4o for transactional data.

The Hybrid CSV/Prefix and Simple Prefixes prompt styles balance token usage and maintain moderate verbosity. These results highlight ChatGPT-4o's ability to minimize token consumption while ensuring high-quality outputs for structured and semi-structured datasets. ChatGPT-4o does exhibit notable variability, however, influenced by both prompt style and dataset type.

4.1.3. Time Analysis for ChatGPT-4o

Figure 17 visualizes the processing times of ChatGPT-4o across all the prompt styles and dataset types.

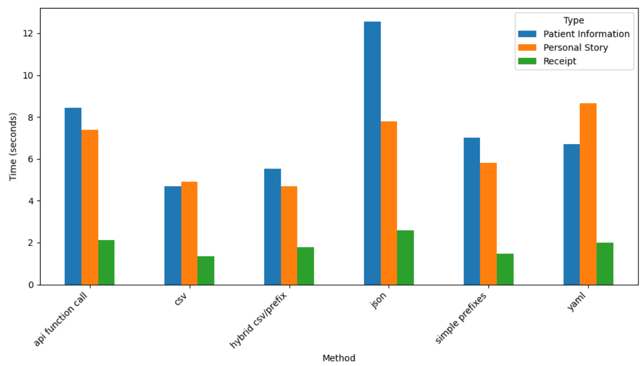

Figure 17. ChatGPT-4o Time Taken by Prompt Style and Type (extended from [8]).

These results indicate that the API Function Call and JSON prompt styles exhibit the highest processing times, particularly for *Patient Information* and *Personal Story* datasets. JSON processing for *Patient Information* reaches over ten seconds, reflecting the complexity of generating detailed structured outputs.

In contrast, the CSV prompt style and *Receipt* dataset show the lowest processing times across all prompt styles, with times often below three seconds. These results show ChatGPT-4o generates concise outputs efficiently for transactional data in simple formats. The Hybrid CSV/Prefix and YAML prompt styles show moderate processing times across datasets, balancing complexity and output quality.

In general, processing times for *Patient Information* datasets remain consistently higher than for other types, highlighting the intricacy of these structured inputs. These findings underscore ChatGPT-

 doi:10.20944/preprints202506.1937.v1

4o's efficiency in simpler formats and transactional data while revealing its challenges in handling complex, hierarchical outputs.

ChatGPT-4o's performance across accuracy, token usage, and time metrics show its strengths and limitations in handling different datasets and prompt styles. It achieves high accuracy for structured datasets like *Patient Information* and *Receipt*, especially when using the API Function Call and YAML prompt styles, but struggles with the narrative-style *Personal Story* dataset. It efficiently minimizes token usage for simpler formats like CSV, making it suitable for tasks with strict token constraints, but requires many more tokens for verbose prompt styles, such as API Function Call and JSON.

Finally, ChatGPT-4o exhibits competitive time efficiency by processing simpler formats like CSV and *Receipt* datasets quickly. However, more complex prompt styles and datasets, such as *Patient Information* and JSON, demand significantly longer processing times. These results reaffirm and extend our earlier findings on ChatGPT-4o [8], confirming its strength in structured data generation and identifying new trends when evaluated across a broader spectrum of prompt styles and datasets.

4.2. Analysis of Experiment Results for Claude

This section analyzes Claude's performance across the three *Personal Stories*, *Medical Records*, and *Receipt Records* datasets described in Section 3.1. Claude was evaluated using customized prompts designed to leverage its strengths in generating structured outputs while adapting to the varying complexities of the datasets. As with ChatGPT-4o, accuracy, token usage, and generation time metrics were analyzed to quality how well the Claude LLM (1) captured the essential attributes of each dataset and (2) how effectively it handled the trade-offs between verbosity, efficiency, and output quality. This analysis provides insights into Claude's ability to handle structured and narrative data, offering a comparative perspective against other LLMs.

4.2.1. Accuracy Analysis for Claude

Figure 18 depicts Claude's accuracy across various prompt styles and dataset types.

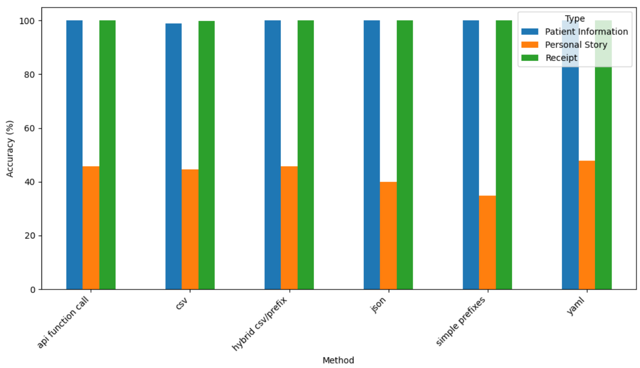

Figure 18. Claude Accuracy by Prompt Style and Type.

These results reveal that Claude achieves consistently high accuracy (close to 100%) for structured datasets, such as *Patient Information* and *Receipt* across all prompt styles, demonstrating its strength in handling structured data effectively. Its performance drops significantly, however, for the narrative-style *Personal Story* dataset, with accuracy stabilizing at around 40% across all prompt styles.

This analysis highlights Claude's uniform performance across different prompt styles (such as API Function Call, YAML, and CSV), reflecting its robustness in generating output for structured datasets. For less structured datasets, such as *Personal Story*, Claude's accuracy remains consistent regardless of prompt style, indicating that dataset complexity, rather than prompt structure, plays a more significant role in its performance. These findings underscore Claude's reliability in structured tasks, while identifying areas for improving its narrative-style data generation.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2025 doi:10.20944/preprints202506.1937.v1

4.2.2. Token Usage Analysis for Claude

Figure 19 depicts the token usage of the Claude LLM across various prompt styles and dataset types.

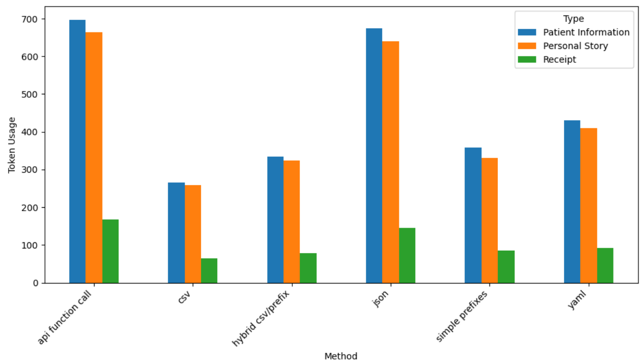

Figure 19. Claude Token Usage by Prompt Style and Type.

These results indicate that API Function Call and JSON prompt styles consume the highest number of tokens, particularly for *Patient Information* and *Personal Story* datasets, where token usage exceeds 700 tokens. This finding reflects the verbose and detailed outputs required for these structured tasks.

In contrast, the CSV and Simple Prefixes prompt styles show the lowest token usage for Claude. In particular, the *Receipt* dataset consistently shows minimal token consumption. This efficiency highlights Claude's adaptability in generating concise outputs for simpler datasets and formats.

The moderate range of token usage for the Hybrid CSV/Prefix and YAML prompt styles reveal how Claude balances verbosity and efficiency. Across all prompt styles, *Patient Information* datasets consistently require the most tokens, followed by *Personal Story*, while *Receipt* datasets remain the least token-intensive. These results show Claude's ability to balance output quality and verbosity based on dataset complexity and output requirements.

4.2.3. Time Analysis for Claude

Figure 20 shows Claude's time performance across the prompt styles and dataset types.

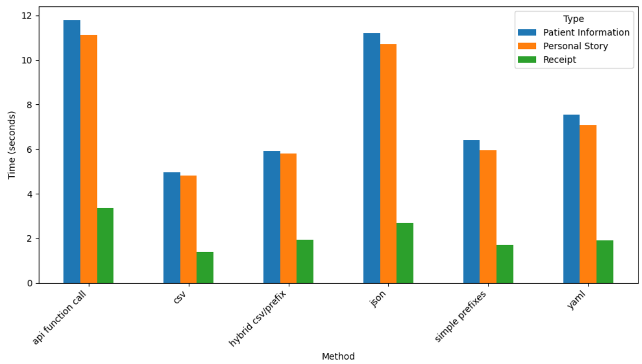

Figure 20. Claude Time Taken by Prompt Style and Type.

These results indicate the API Function Call and JSON prompt styles require the most time for processing, with *Patient Information* datasets taking over twelve seconds on average. This finding reflects the complexity of generating detailed and structured outputs for these prompt styles and datasets.

Conversely, the CSV prompt style demonstrates the fastest processing times, particularly for the *Receipt* dataset, where times fall below 2 seconds. This efficiency highlights Claude's ability to handle simple tabular formats and transactional data effectively.

 doi:10.20944/preprints202506.1937.v1

Hybrid CSV/Prefix and YAML prompt styles exhibit moderate processing times, balancing the trade-offs between complexity and efficiency. Across all prompt styles, *Patient Information* consistently requires the most processing time, followed by *Personal Story*, while *Receipt* datasets remain the fastest to process. These results underscore Claude's adaptability in balancing processing time with output complexity and dataset characteristics.

Claude's performance across the accuracy, token usage, and time metrics show its strengths in handling structured data and its limitations with narrative-style datasets. Claude achieves near-perfect accuracy for structured datasets, such as *Patient Information* and *Receipt*, regardless of the prompt style used. It performance declines significantly, however, for the narrative-style *Personal Story* dataset, where accuracy stabilizes around 40% across prompt styles.

Claude's token usage demonstrates efficiency for simpler prompt styles like CSV and Simple Prefixes, especially for the *Receipt* dataset. It requires substantially more tokens, however, for verbose prompt styles like API Function Call and JSON, particularly for *Patient Information* and *Personal Story*.

Claude's processing times are similarly dataset-dependent, with *Patient Information* and *Personal Story* requiring the longest times for API Function Call and JSON prompt styles, which exceed twelve seconds. Conversely, simpler prompt styles like CSV and datasets like *Receipt* exhibit faster processing times, often below two seconds. These results emphasize Claude's capability for structured data tasks while revealing areas to improve its efficiency and performance for narrative or unstructured data.

4.3. Gemini Analysis of Experiment Results

This section examines the performance of the Gemini LLM across the three datasets described in Section 3.1 in accordance with the accuracy, token usage, and processing time metrics described in Section 3.2. This analysis assessed Gemini's adaptability and effectiveness in handling structured, semi-structured, and narrative data and quantified Gemini's strengths in producing structured outputs while identifying its trade-offs in computational efficiency and output quality across diverse datasets.

4.3.1. Accuracy Analysis for Gemini

Figure 21 visualizes the accuracy of Gemini across the various prompt styles and dataset types.

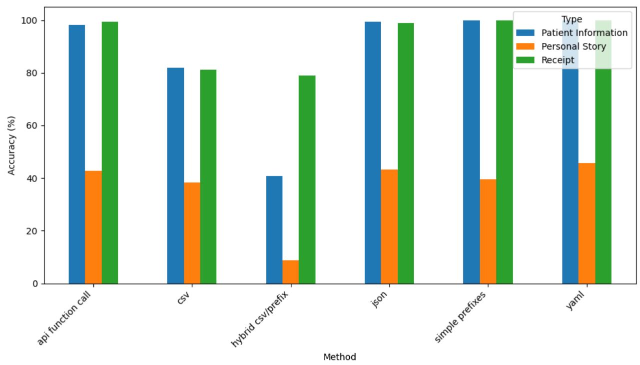

Figure 21. Gemini Accuracy by Prompt Style and Type.

These results indicate consistently high accuracy for structured datasets like *Patient Information* and *Receipt* across most prompt styles, with scores nearing 100%. This finding demonstrates Gemini's capability to generate reliable outputs for well-structured data.

In contrast, Gemini struggled with the narrative-style *Personal Story* dataset, where accuracy stabilized at ∼40% across all prompt styles, reflecting limitations in structuring less formal, narrative inputs. Among the prompt styles, API Function Call and YAML exhibit the highest accuracy for structured datasets, while Hybrid CSV/Prefix shows a slight decline in accuracy for *Patient Information*, suggesting challenges in mixed-format data generation. Overall, Gemini displays strong adaptability to various prompt styles, excelling in structured tasks while maintaining consistent but moderate performance for narrative data.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2025 doi:10.20944/preprints202506.1937.v1

4.3.2. Token Usage Analysis for Gemini

Figure 22 shows Gemini's token usage across various prompt styles and dataset types.

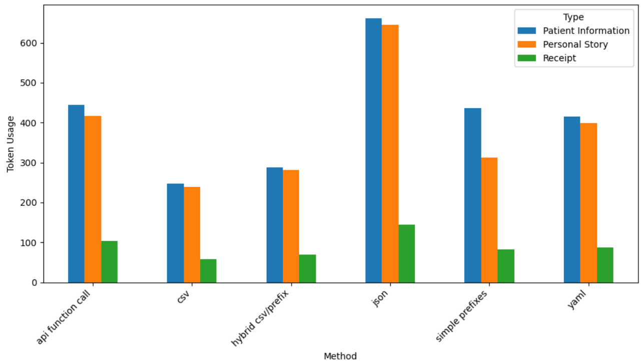

Figure 22. Gemini Token Usage by Prompt Style and Type.

These results indicate that API Function Call and JSON prompt styles require the highest number of tokens, particularly for *Patient Information* and *Personal Story*, where token usage exceeds 500 tokens. This finding reflects the verbose and detailed outputs generated by Gemini for structured and semi-structured datasets.

In contrast, the CSV prompt style demonstrates the lowest token usage, particularly for the *Receipt* dataset, where usage remains consistently below 100 tokens. This efficiency highlights Gemini's ability to adapt its verbosity to simpler formats and transactional data.

Hybrid CSV/Prefix and YAML prompt styles fall within a moderate range of token usage, balancing output verbosity and efficiency effectively. Across all prompt styles, *Patient Information* consistently requires the most tokens, followed by *Personal Story*, while *Receipt* datasets remain the least token-intensive. These findings emphasize Gemini's ability to balance token consumption while maintaining high output quality across diverse datasets and prompt styles.

4.3.3. Time Analysis for Gemini

Figure 23 depicts Gemini's time performance across all the prompt styles and dataset types.

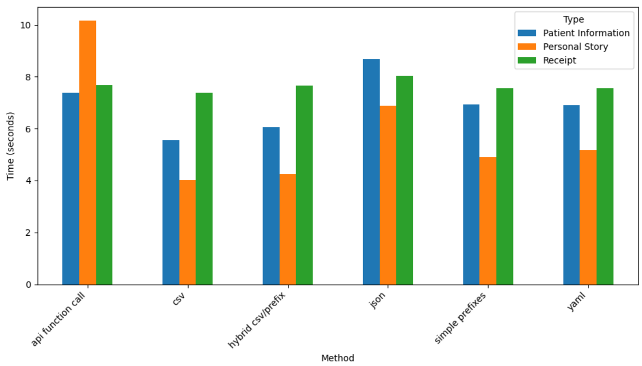

Figure 23. Gemini Time Taken by Prompt Style and Type.

These results show that the API Function Call prompt style requires the longest processing time across all datasets, particularly for *Personal Story*, which exceeds ten seconds. This finding reflects the additional complexity and verbosity required for narrative-style data in this prompt style.

Conversely, the CSV and Hybrid CSV/Prefix prompt styles demonstrate the shortest processing times, especially for the *Receipt* dataset, where times consistently remain below four seconds. This finding shows Gemini's ability to handle structured and simple formats efficiently.

YAML and JSON prompt styles exhibit moderate processing times, balancing output quality and computational demand. Across all prompt styles, *Patient Information* and *Receipt* datasets display

 doi:10.20944/preprints202506.1937.v1

consistent performance, whereas *Personal Story* remains the most variable and often the most time-intensive. These results emphasize Gemini's efficiency for structured data and its adaptability to diverse prompt styles and dataset types.

Gemini demonstrates strong performance across the accuracy, token usage, and time metrics, with clear strengths in handling structured data and moderate adaptability for narrative-style inputs. It achieves near-perfect accuracy for structured datasets like *Patient Information* and *Receipt* across most prompt styles, while its performance for the narrative-style *Personal Story* dataset is consistent but lower, stabilizing around 40%. Gemini's token usage balances verbosity and efficiency effectively, with prompt styles like CSV and Hybrid CSV/Prefix requiring minimal tokens, particularly for simpler datasets like *Receipt*.

However, verbose prompt styles such as API Function Call and JSON exhibit higher token usage, especially for complex datasets like *Patient Information* and *Personal Story*. Processing times further reflect this trend, with API Function Call being the most time-intensive, particularly for *Personal Story* datasets. In contrast, the CSV and Hybrid CSV/Prefix prompt styles demonstrate the fastest times, especially for the *Receipt* dataset. These results highlight Gemini's adaptability and efficiency in structured data tasks, while also revealing areas for improvement in handling narrative or semi-structured data with less verbosity and faster response times.

5. Comparison of ChatGPT-4o, Claude, and Gemini LLM Models

This section provide a comparative analysis of the performance of GPT-4o, Claude, and Gemini on the three datasets described in Section 3.1. This analysis builds directly upon our prior study of prompt style efficiency using GPT-4o and three prompt formats (JSON, YAML, and Hybrid CSV/Prefix) [8]. In this extended work, we introduce three additional prompt styles (CSV, API Function Call, and Simple Prefixes) and evaluate them across three advanced LLMs. While we reuse elements of the experimental design and visualization formats (such as accuracy/token/time comparisons), all results have been independently re-generated and expanded to support model-level comparative analysis. As before, our evaluation focuses on the accuracy, token usage, and processing time metrics to highlight the pros and cons of each LLM across various prompt styles and data types. By examining their adaptability to structured, semi-structured, and narrative datasets, this comparison quantifies unique characteristics of each LLM and identifies trade-offs between efficiency, quality, and computational demands.

5.1. Comparing the Accuracy of ChatGPT-4o, Claude, and Gemini

The comparative accuracy performance of ChatGPT-4o, Claude, and Gemini across various prompt styles is shown in Figure 24.

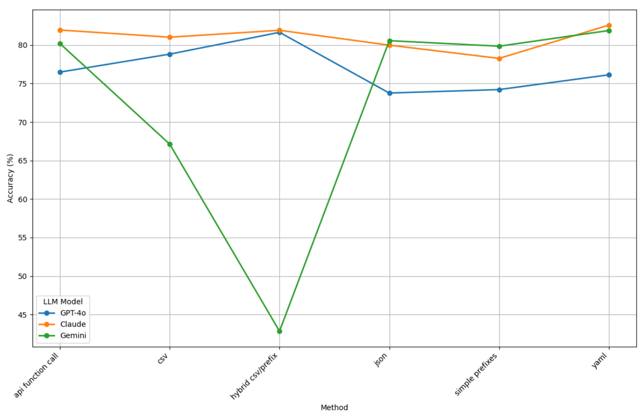

Figure 24. Accuracy Comparison Across LLMs (extended from [8]).

This figure shows that Claude is the most accurate LLM, consistently achieving accuracy levels above 80% across all prompt styles. Its ability to handle diverse input structures and datasets under-

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2025 doi:10.20944/preprints202506.1937.v1

scores its robustness. ChatGPT-4o also demonstrates steady performance, with accuracy levels ranging between 75% and 80%, reflecting its adaptability to various prompt styles.

In contrast, Gemini exhibits variable accuracy, performing well in structured prompt styles like YAML and API Function Call but showing significant declines in Hybrid CSV/Prefix, where its accuracy dips below 65%. This finding highlights potential challenges in managing mixed-style prompts for certain datasets. Among the prompt styles, API Function Call and YAML show high accuracy for all models, while Hybrid CSV/Prefix poses difficulties for Gemini. These results highlight Claude's superior versatility, ChatGPT-4o's stability, and Gemini's specialized strengths with notable areas for improvement.

5.2. Comparing the Token Usage of ChatGPT-4o, Claude, and Gemini

Figure 25 compares the token usage of ChatGPT-4o, Claude, and Gemini across various prompt styles.

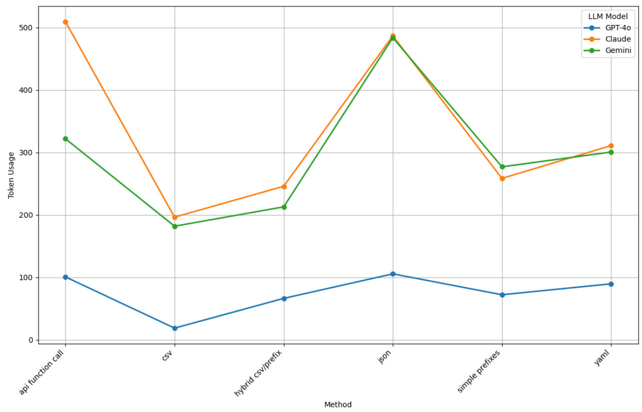

Figure 25. Token Usage Comparison Across LLMs (extended from [8]).

These results show ChatGPT-4o consistently consuming the fewest tokens across all prompt styles, underscoring its efficiency in generating concise outputs. This efficiency is particularly notable in prompt styles like CSV, Simple Prefixes, and Hybrid CSV/Prefix, where ChatGPT-4o's token usage remains much lower than Claude and Gemini's usage.

In contrast, Claude exhibits the highest token usage for most prompt styles, indicating a propensity to produce more verbose responses. This trend is particular evident in the API Function Call and JSON prompt styles, where Claude's token consumption surpasses ChatGPT-4o and Gemini by a substantial margin. Gemini positions itself between ChatGPT-4o and Claude, balancing verbosity and token efficiency, and demonstrates moderate token usage across all prompt styles.

These observations suggest that ChatGPT-4o is the most token-efficient model, which is advantageous in scenarios (such as applications with strict token limits or cost constraints) where minimizing token consumption is critical. Claude's higher token usage may be beneficial for use cases that require more detailed and elaborate responses, but could be less suitable in contexts where token economy is essential. Gemini offers a middle ground, providing a balance between detailed output and token efficiency, making it a versatile option for a variety of applications.

5.3. Comparing Time Performance of ChatGPT-4o, Claude, and Gemini

Figure 26 depicts the comparative time performance of ChatGPT-4o, Claude, and Gemini across various prompt styles.

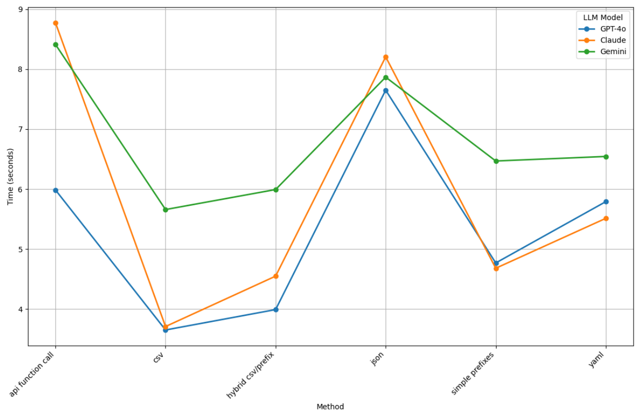

Figure 26. Time Comparison Across LLMs (extended from [8]).

This figure shows that Claude consistently demonstrates the longest processing times, particularly for API Function Call and JSON, where times approach nine seconds. This finding reflects its detailed output generation process, which prioritizes verbosity over efficiency.

In contrast, ChatGPT-4o consistently delivers the fastest processing times across all prompt styles, maintaining a range between four and six seconds. Even for more complex prompt styles like JSON, ChatGPT-4o's time efficiency highlights its streamlined approach to output generation. Gemini exhibits variable performance, balancing efficiency for prompt styles like CSV and Hybrid CSV/Prefix while showing spikes in time for JSON, where its processing time aligns with Claude's.

For simpler prompt styles, such as CSV, all three models demonstrate shorter processing times, with ChatGPT-4o leading in efficiency. These results emphasize ChatGPT-4o's time efficiency, Claude's verbosity at the cost of longer processing times, and Gemini's balanced approach with occasional variability Figure 26.

5.4. Comparing ChatGPT-4o, Claude, and Gemini Performance Across All Metrics

A comparative analysis of ChatGPT-4o, Claude, and Gemini shows distinct trade-offs in performance across the accuracy, token usage, and time efficiency metrics. We summarize these results through two complementary visualizations, each offering unique insights into the data.

5.4.1. Metric-by-Metric Comparison Using Bar Charts

Figure 27 presents a multi-metric bar chart, expanding on the visualization techniques introduced in [8] to support cross-model comparison across six prompt styles. This figure highlights overall trends, emphasizing how each LLM aligns with specific use cases. For example, Claude achieves the highest accuracy (85%), excelling in tasks that demand complex data structures, such as JSON and YAML. While ChatGPT-4o is slightly less accurate at 78%, it provides the best token and time efficiency, making it the most practical choice for cost-sensitive tasks. Gemini, at 76%, offers balanced performance, with moderate results across all metrics.

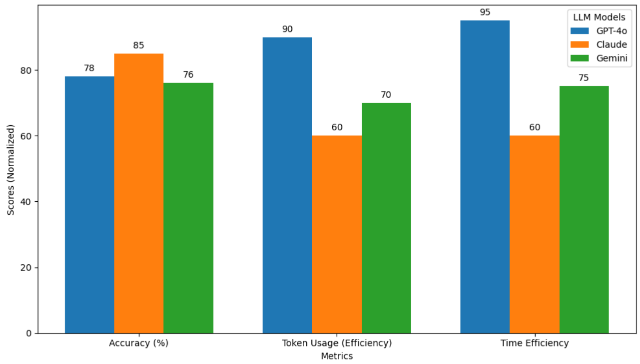

Figure 27. Metric-by-Metric Performance of ChatGPT-4o, Claude, and Gemini (extended from [8]).

 doi:10.20944/preprints202506.1937.v1

5.4.2. Holistic Trade-Off Analysis Using Radar Charts

Figure 28 provides a radar chart summarizing the same metrics, offering a holistic view of the trade-offs among the three LLMs.

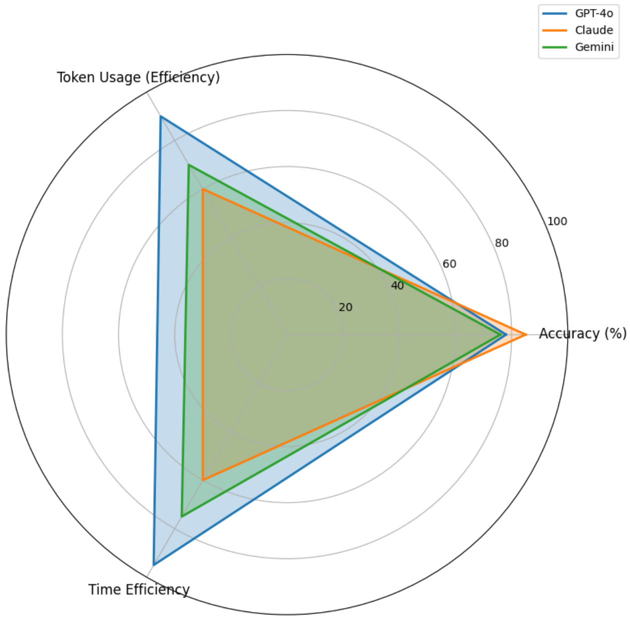

Figure 28. Aggregate Comparison of ChatGPT-4o, Claude, and Gemini Performance (extended from [8]).

Unlike the bar chart in Figure 27, this figure visualizes LLM strengths and weaknesses via a unified profile, allowing a more intuitive understanding of how each LLM balances its capabilities. For instance, Claude's focus on accuracy is visually distinct from ChatGPT-4o's dominance in token and time efficiency, whereas Gemini shows a steady but less pronounced performance across all axes.

We intentionally included both Figure 27 and 28 since each serves a different analytical purpose. The bar chart is ideal for detailed, metric-specific comparisons, enabling precise quantification of each LLM's strengths and weaknesses. In contrast, the radar chart facilitates a high-level overview, which helps stakeholders grasp LLM trade-offs quickly for practical decision-making.

5.4.3. Performance Analysis

Claude achieves the highest accuracy at 85%, particularly excelling in tasks requiring complex hierarchical representations, such as JSON and YAML prompts. This finding highlights Claude's strong ability to preserve the structural integrity of outputs. ChatGPT-4o, with an accuracy of 78%, offers consistent performance across most prompt styles but falls short of Claude's precision in highly structured outputs. While Gemini trails slightly at 76%,it exhibits competitive accuracy, showing potential for handling structured tasks effectively, but with some room for refinement.

Token efficiency places ChatGPT-4o as the most resource-efficient model, achieving a score of 90%, significantly outperforming Claude (60%) and Gemini (70%). This find demonstrates ChatGPT-4o's capability to generate concise and compact outputs while maintaining acceptable accuracy levels. Both Claude and Gemini show higher token usage, particularly in verbose formats such as JSON, indicating a trade-off between verbosity and precision.

ChatGPT-4o again emerges as the time efficiency leader with a score of 95%, delivering faster response times across a majority of prompt styles. Gemini follows with a moderate time efficiency score of 75%, balancing speed and accuracy reasonably well. Claude lags behind at 60%, reflecting its computational overhead and longer processing times, particularly for resource-intensive tasks.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2025 doi:10.20944/preprints202506.1937.v1

5.4.4. Trade-Offs and Practical Recommendations

Our results underscore key trade-offs among ChatGPT-4o, Claude, and Gemini, offering practical recommendations for selecting models and prompt styles based on application constraints. Claude prioritizes accuracy, making it ideal for tasks that demand high-fidelity structured outputs, particularly in complex formats such as JSON and YAML. This model excels in preserving nested hierarchies and attribute completeness, albeit at the cost of increased token usage and slower processing times.

ChatGPT-4o, by contrast, offers the best balance between performance and resource efficiency. It consistently outperforms the other models in token usage and processing speed while maintaining acceptable levels of accuracy. These properties make ChatGPT-4o particularly well-suited for cost-sensitive or real-time applications where latency and token limits are primary concerns.

Gemini demonstrates balanced performance across metrics, making it a versatile candidate for general-purpose structured data generation. However, its reduced accuracy in prompt styles such as Hybrid CSV/Prefix and increased variability in time and token metrics for verbose prompts suggest that further prompt tuning or model alignment may be needed for more specialized tasks.

This expanded multi-model trade-off analysis builds on the insights from our prior work [8], which focused on a single-model (GPT-4o) comparison. By introducing model-specific constraints and behaviors, our findings offer a more comprehensive framework for practitioners aiming to optimize structured data workflows in LLM applications.

5.4.5. Summarizing Insights in a Comparative Table

Table 1 provides a comparative analysis of ChatGPT-4o, Claude, and Gemini across key accuracy, token usage, and time efficiency metricsx. This table highlights the distinct strengths and weaknesses of each LLM model when evaluated against multiple methods. Specifically, the table summarizes the overall accuracy trends, the best-performing methods for accuracy, token usage efficiency, and time efficiency, revealing trade-offs and areas where each LLM excels or falls short.

This summary table expands upon the single-model comparison in our earlier work [8] by offering LLM-specific insights that capture model-wise trade-offs between accuracy, token cost, and time performance.

Table 1. Comparative Analysis of ChatGPT-4o, Claude, and Gemini.

Metric	GPT-4o	Claude	Gemini
Accuracy (Overall)	Consistently between 75-80% across methods	Highest accuracy (>80%) across most methods	Variable; excels in YAML but struggles in Hybrid CSV/Prefix
Best Accuracy Method	API Function Call & YAML	JSON & YAML	YAML
Token Usage (Overall)	Lowest across all methods (<100 tokens)	Highest for verbose methods (>500 tokens)	Moderate token usage with spikes for JSON
Best Token Usage Method	CSV & Simple Prefixes	Hybrid CSV/Prefix	YAML
Time Efficiency	Fastest processing time (4-6 seconds)	Longest for verbose methods (7-9 seconds)	Moderate; aligns with Claude for JSON and faster for CSV

As shown in Table 1, Gemini demonstrates a balanced performance profile, particularly with prompt styles like YAML and Simple Prefixes. However, its reduced accuracy and elevated variability with Hybrid CSV/Prefix suggest optimization opportunities for mixed-format prompt handling. The JSON and YAML prompts, shared across all three models, are particularly effective for tasks requiring hierarchical data representation but result in higher token consumption due to their verbosity. Meanwhile, CSV and Simple Prefix prompts provide lightweight, cost-effective solutions with shorter processing times but sacrifice some flexibility for representing complex data structures. The Hybrid

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2025 doi:10.20944/preprints202506.1937.v1

CSV/Prefix format attempts to balance tabular and prefixed strengths, showing potential for diverse applications but presenting challenges in ensuring consistent performance across all three models.

In summary, Claude excels in accuracy for structured and verbose outputs. In contrast, ChatGPT-4o prioritizes efficiency and time, whereas Gemini offers a balanced approach while revealing areas for refinement in mixed-format tasks. These findings provide actionable insights for selecting the most suitable model and format based on specific application requirements.

6. Related Work

Recent advancements in large language models (LLMs) such as ChatGPT-4o, Claude, and Gemini have significantly expanded their applicability to structured data generation tasks in healthcare, e-commerce, and personalized content synthesis. Prior work has extensively explored prompt engineering strategies, performance metrics, and domain-specific applications. This section surveys these foundations and contextualizes our contributions in relation to existing research, as shown in Figure 29.

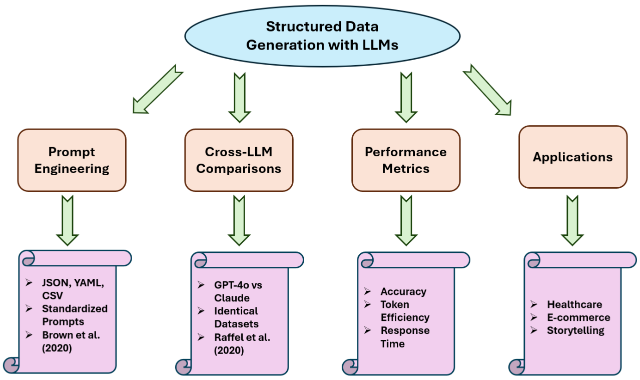

Figure 29. Structured Data Generation with LLMs(adapted from [8]).

Prompt engineering: Crafting effective and structured outputs. Prompt design plays a critical role in harnessing the capabilities of LLMs to generate accurate and structured outputs. Prior studies have investigated prompt styles such as JSON, YAML, CSV, and prefix-based prompt styles, analyzing their efficacy in guiding model behavior. Notably, research by Brown et al. (2020) [6] introduced prompt-based prompt styles to instruct generative models, highlighting the importance of input formatting. Most studies tailored prompts for specific LLMs, however, potentially biasing cross-model evaluations. Our work contributes to this field by employing a standardized set of prompts across GPT-4o, Claude, and Gemini, ensuring consistency and fairness in comparative analysis.

Cross-LLM comparisons: Evaluating LLMs under consistent conditions. Existing work often focuses on the isolated performance of a single LLM, such as OpenAI's ChatGPT models or Anthropic's Claude. Few studies conduct direct comparisons across multiple LLMs under standardized conditions. Raffel et al. (2020) [18] compared various transformer-based models in the T5 framework, emphasizing the need for benchmark datasets to assess model robustness. Similarly, Zhang et al. (2021) [11] explored cross-model evaluations for task-specific applications but relied on tailored prompts. In contrast, our study evaluates three leading LLMs on identical datasets using consistent prompts, providing a more equitable basis for comparison.

Performance metrics: Measuring efficiency, accuracy, and cost-effectiveness. Measuring LLM performance requires a multi-faceted approach, incorporating key metrics, such as accuracy, token usage, and response time. Studies by Vaswani et al. (2017) [20] and later refinements by OpenAI [1] and Anthropic [2] have emphasized token efficiency [3] as a key factor in determining LLM scalability and cost-effectiveness. Other research has explored accuracy in generating structured outputs, particularly in domains like medical records and financial receipts that require hierarchical data representation. Our work extends this prior research by analyzing these metrics comprehensively across diverse

 doi:10.20944/preprints202506.1937.v1

datasets and prompt styles, highlighting the trade-offs between verbosity, efficiency, and generation quality.

Applications of LLMs in healthcare, e-commerce, and storytelling. LLMs have increasingly being adopted and applied to generate structured data in domains like healthcare, personalized storytelling, and transactional record management. For instance, prior studies have demonstrated the use of LLMs in generating synthetic medical records [14] for data augmentation and training machine learning models. Similarly, e-commerce applications have leveraged LLMs for generating product descriptions [13] and receipt summaries. Our datasets, including *Personal Stories*, *Medical Records*, and *Receipt Records*, align with these use cases, enabling a realistic evaluation of LLM performance in relevant scenarios.

Extending Prior Findings through a Multi-Model, Multi-Prompt Framework. While our prior work [8] focused on evaluating three prompt styles with GPT-4o, the present study significantly expands the prompt and model space. By conducting comparative evaluations across ChatGPT-4o, Claude, and Gemini using six prompt styles, we generalize previous conclusions and identify nuanced model-specific behaviors and trade-offs. This broader framework provides a more actionable foundation for prompt design and model selection in structured data applications.

7. Concluding Remarks

This paper presented a detailed methodology, comprehensive analysis, and practical recommendations stemming from a study evaluating the impact of prompt styles on structured data generation tasks with LLMs. Our study results contribute to the growing body of knowledge on prompt engineering and structured data generation with LLMs. The following are lessons we learned from conducting the research presented in this paper:

- **Trade-offs in prompt design are context-dependent**. Understanding the trade-offs associated with different prompting techniques helps developers and data scientists make informed choices that balance accuracy, efficiency, and cost-effectiveness. For instance, hierarchical formats like JSON and YAML [9] offer superior accuracy but at a higher token cost, whereas CSV and simple prefixes [5] provide cost-efficient alternatives with reduced flexibility for complex structures.
- **Alternative formats deliver unique advantages**. Alternative formats, such as simple prefixes and hybrid approaches, can offer high accuracy with reduced token costs in certain use cases. These formats are less verbose and strike a balance between clarity and conciseness, making them valuable for semi-structured data, such as receipts or transactional records.
- **Consistent prompts enhances evaluation fairness**. Using consistent prompts across different LLMs ensures fairness in evaluation and highlights the inherent capabilities of each LLM. Our findings demonstrate that standardized prompts provide a reliable baseline for performance comparison to avoid biases caused by prompt tailoring for specific LLMs.
- **Efficiency gains vary across LLMs**. The study revealed significant differences in token usage and processing times among ChatGPT-4o, Claude, and Gemini. While ChatGPT-4o had the highest efficiency in both token consumption and time, Claude excelled in accuracy and Gemini struck a balance between the two. These insights guide LLM selection for specific use cases where speed or cost constraints are critical.
- **Applications influence prompt selection**. The selection of prompt stlyes can vary significantly based on application domain. For example, our results demonstrate how JSON and YAML formats are better suited for domains like healthcare requiring hierarchical data representation. Conversely, CSV and simple prefixes excel in domains like e-commerce where token efficiency and processing speed are critical. While our datasets simulate healthcare (*e.g.*, medical records) and e-commerce (*e.g.*, receipt records), our results generalize to similar structured data across these domains.
- **Prompt design impacts outcomes**. While our study does not focus directly on iterative prompt refinement [15], the results indicate that careful initial prompt design plays a key role in deter-

mining the accuracy and efficiency of LLM outputs. This finding underscores the importance of selecting prompt styles that align with the dataset's complexity and intended use case. Our future work will explore the role of iterative refinement in enhancing LLM output quality, which is beyond the scope of this paper.

Overall, this paper extends our earlier work [8] by expanding the prompt design space, model diversity, and evaluation depth. These contributions support a broader understanding of how LLMs can be effectively leveraged for structured data generation in real-world applications.

Acknowledgments: We used ChatGPT-4o's Advanced Data Analysis (ADA) capability to generate code for the data visualizations and filter the datasets. Moreover, Claude and Gemini's outputs provided a comparative evaluation of LLM performance across structured data generation tasks. Their contributions enriched the analysis and insights presented in this study.

References

1. Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
2. Adebowale Jeremy Adetayo, Mariam Oyinda Aborisade, and Basheer Abiodun Sanni. Microsoft copilot and anthropic claude ai in education and library service. *Library Hi Tech News*, 2024.
3. Ramon Maria Garcia Alarcia and Alessandro Golkar. Optimizing token usage on large language model conversations using the design structure matrix. *arXiv preprint arXiv:2410.00749*, 2024.
4. Jo Inge Arnes and Alexander Horsch. Schema-based priming of large language model for data object validation compliance. *Available at SSRN 4453361*, 2023.
5. Alexander Ball, Lian Ding, and Manjula Patel. Lightweight formats for product model data exchange and preservation. In *PV 2007 Conference*, pages 9–11, 2007.
6. Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-shot Learners. *arXiv preprint arXiv:2005.14165*, 33:1877–1901, 2020.
7. John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S Rosen, Gerbrand Ceder, Kristin A Persson, and Anubhav Jain. Structured Information Extraction from Scientific Text with Large Language Models. *Nature Communications*, 15(1):1418, 2024.
8. Ashraf Elnashar, Jules White, and Douglas C. Schmidt. Enhancing structured data generation with gpt-4o evaluating prompt efficiency across prompt styles. *Frontiers in Artificial Intelligence*, Volume 8 - 2025, 2025.
9. Malin Eriksson and Victor Hallberg. Comparison between json and yaml for data serialization. *The School of Computer Science and Engineering Royal Institute of Technology*, pages 1–25, 2011.
10. Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.
11. William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.
12. Lin Guo. The effects of the format and frequency of prompts on source evaluation and multiple-text comprehension. *Reading Psychology*, 44(4):358–387, 2023.
13. Shashank Kedia, Aditya Mantha, Sneha Gupta, Stephen Guo, and Kannan Achan. Generating rich product descriptions for conversational e-commerce systems. In *Companion Proceedings of the Web Conference 2021*, pages 349–356, 2021.
14. Gleb Kumichev, Pavel Blinov, Yulia Kuzkina, Vasily Goncharov, Galina Zubkova, Nikolai Zenovkin, Aleksei Goncharov, and Andrey Savchenko. Medsyn: Llm-based synthetic medical text generation framework. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 215–230. Springer, 2024.
15. Zhexin Liang, Chongyi Li, Shangchen Zhou, Ruicheng Feng, and Chen Change Loy. Iterative prompt learning for unsupervised backlit image enhancement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8094–8103, 2023.
16. Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

17. Max Moundas, Jules White, and Douglas C. Schmidt. Prompt Patterns for Structured Data Extraction from Unstructured Text. In *Proceedings of the 31st Pattern Languages of Programming (PLoP) conference*, Columbia River Gorge, WA, October 2024.
18. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-text Transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
19. Sindhu Tipirneni, Ming Zhu, and Chandan K. Reddy. Structcoder: Structure-aware transformer for code generation, 2024.
20. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
21. Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*, 2023.