EMNLP Project Report

Team members : Ayan Datta, Hardik Mittal

Evaluating Structured Outputs in NLP: Metrics, Models, and Beyond

Table of Contents

1. Introduction

In recent years, the landscape of Natural Language Processing (NLP) has shifted dramatically with the rise of large language models (LLMs) like GPT-4, LLaMA, and Qwen. These models are no longer limited to producing unstructured text; increasingly, they are expected to output **structured, machine-interpretable formats** such as JSON, YAML, XML, and even programming languages or markup formats.

This transition from free-form generation to **structured output generation** is more than a technical curiosity—it is foundational to a new class of NLP applications where outputs feed directly into APIs, user interfaces, or downstream deterministic systems. From **function calling** in OpenAI models to **enterprise data pipelines** that require schema-conformant outputs, structured generation is a vital interface between LLMs and real-world software systems.

Despite this, **evaluating structured outputs** remains a poorly understood problem. Traditional NLP metrics like BLEU and ROUGE were designed for flat text and fall short in capturing the nuances of structured formats. Newer embedding-based metrics like BERTScore offer better semantic fidelity, but still struggle with hierarchical organization and semantic relationships embedded within tree-like structures.

This project explores the **evaluation of structured outputs**, focusing primarily on JSON, and proposes a set of techniques and metrics that better capture the fidelity, accuracy, and usefulness of generated structured data. Our experiments span multiple models, including large-scale LLMs and lightweight 3B parameter models, and employ both automatic and LLM-based evaluation schemes.

2. Motivation and Applications

2.1 Why Structured Outputs Matter in NLP

As LLMs become integral components of real-world software systems, the ability to **produce structured outputs** is no longer optional—it is a necessity. Unlike free-text generation, structured outputs enable machine-level interpretation, interoperability with APIs, and downstream integration with databases, GUIs, and business logic layers.

A well-structured output format like JSON provides:

- **Deterministic Parsing**: Downstream systems can reliably parse and extract information without relying on fragile regex or natural language cues.
- **Reduced Post-processing Overhead**: Developers no longer need to write brittle code to interpret free-text responses, minimizing technical debt and failure points.
- **Schema Enforcement**: Many systems enforce strict data contracts (e.g., OpenAPI, GraphQL). Structured outputs must conform to these contracts for seamless function invocation or UI rendering.

Recent developments in API-driven language models (e.g., OpenAI's **function calling** with strict schema requirements and **JSON mode** in GPT-4) underscore the importance of this trend. Evaluating how well a modeladheres to these structures is critical not only for performance benchmarking, but for deployment readiness.

2.2 Real-World Applications

Structured output generation has found use across a broad array of NLP-driven systems:

✅ Chatbots and Virtual Assistants

Modern assistants like Alexa or Siri require LLMs to output structured intents, slots, and entity-value pairs. For example:

```
{
  "intent": "book_flight",
  "parameters": {
    "origin": "New York",
    "destination": "London",
    "date": "2024-05-20"
  }
}
```

Evaluating the quality of such outputs is non-trivial—simple string metrics are inadequate for validating nested structure, semantic intent, and value correctness.

📡 Information Extraction Pipelines

Enterprises routinely extract structured data from text (e.g., resumes, reports, customer queries, schemas from websites for SEO). LLMs can automate this, but only if their outputs match expected formats:

- Does the output preserve the hierarchy?
- Are all required fields present?
- Are values semantically faithful?

🧠 Function Calling and Tool Use

OpenAI and others have introduced *function calling* APIs, where LLMs are expected to emit JSON adhering to specified schemas to trigger backend logic. Evaluation becomes a matter of checking:

- Schema conformity
- Value accuracy
- Structural alignment

🗂️ Data Pipelines and ETL Processes

LLMs are increasingly being proposed as front-end processors in ETL (Extract, Transform, Load) pipelines. For example, converting unstructured PDFs or emails into JSON databases. Again, success depends on structure, completeness, and semantic correctness.

🔍 Web and Document Understanding

Tasks like web scraping, HTML parsing, LaTeX-to-JSON conversion, and code documentation often involve structured outputs. Evaluating LLM performance on these tasks demands metrics that understand hierarchy, formatting, and content fidelity.

2.3 Structured Output Formats Beyond JSON

While JSON is a natural choice due to its popularity and readability, structured output evaluation is relevant across a spectrum of formats:

- **YAML** – Popular in configuration files.
- **XML** – Widely used in enterprise integration and legacy systems.
- **HTML** – Critical for web generation, scraping, and UI rendering.
- **LaTeX** – For scientific document generation.
- **Code / Abstract Syntax Trees (ASTs)** – For program synthesis, translation, or documentation.

Our evaluation strategies are format-agnostic in principle—they can be adapted for any structure that can be serialized as a tree or graph. This broad applicability motivates the need for general-purpose, robust evaluation techniques.

---

3. Problem Statement

The central goal of this work is to design and assess **methods for evaluating structured outputs**—primarily in **JSON format**—produced by large language models (LLMs). Unlike free-text generation, where established metrics like BLEU, ROUGE, or BERTScore suffice, structured outputs require evaluation techniques that respect both **hierarchical structure** and **semantic accuracy**.

This challenge can be broken down into three critical evaluation dimensions:

3.1 Core Evaluation Objectives

Let P be the predicted structured output from a model, and R be the reference structured output. We aim to quantify how "good" P is relative to R along the following axes:

Semantic Equivalence to the Reference

This objective ensures that the meaning conveyed by the predicted output is the same as that of the reference. Every key, value, and relationship in the reference should be represented in a way that captures the same intended meaning, even if minor variations (such as synonymous key names) are present.

Structural Hierarchy and Organization

This criterion focuses on the arrangement and grouping of elements. The prediction should have a clear and logical organization that reflects the inherent structure of the reference. In other words, the way elements are nested and grouped in the prediction should mirror the relationships and overall organization found in the reference.

Completeness

The prediction must include every element present in the reference without omitting any key parts. This means that all required keys, nested elements, and data points should appear in the predicted output. Completeness ensures that nothing essential is missing or extraneously added.

Data Consistency and Correctness

This objective assesses whether the data types, naming conventions, and formats in the prediction match those in the reference. Consistency in these aspects is crucial so that the output can be interpreted correctly, maintaining uniformity in how data is represented and understood.

3.2 Types of Evaluation Tasks

Evaluation of structured outputs spans a spectrum from **strict schema enforcement** to **open-ended generation**, depending on the constraints imposed during generation.

A. Schema-Based Tasks

These tasks expect outputs in a **predefined format**, such as:

```
{
  "name": "...",
  "location": "...",
  "date_of_birth": "..."
}
```

In this setting, both the structure and key labels are **fixed in advance**, and the primary goal is to **fill values correctly**.

Evaluation Focus:

- Key presence
- Correct types (e.g., strings, arrays)
- Value accuracy (exact or semantic)
- Order (in some API-sensitive cases)

Evaluation techniques include:

- Exact match
- ROUGE/BERTScore over individual values
- Schema validation (e.g., JSON Schema tools)

B. Schema-Free Tasks

Here, there is **no predefined schema**, and the model has freedom to decide:

- What structure to generate
- What keys to use
- How deeply to nest information

Example task:

> “Given the Wikipedia article on Apollo 8, output structured information about the mission and its crew.”

Model output:

```
{
  "mission": "Apollo 8",
  "crew": [
    {"name": "Frank Borman", "role": "Commander"},
    {"name": "Jim Lovell", "role": "Command Module Pilot"},
    {"name": "William Anders", "role": "Lunar Module Pilot"}
  ]
}
```

There could be **multiple correct outputs**, and strict schema enforcement is **not possible**.

Evaluation Focus:

- Semantic coverage of required concepts
- Logical structure (e.g., grouping by section)
- Absence of hallucinations or irrelevant content

Evaluation techniques include:

- Tree/graph similarity
- Edit distance
- Embedding-based path comparison
- LLM-as-a-judge (human-like scoring)

Note: We noticed that this classification is not a strict one, but rather a spectrum, as some schemas are not completely imposing on the output structure, (For example: schema.org, medical case descriptions ontologies)

3.3 Why Traditional Metrics Fall Short

Traditional metrics like BLEU or ROUGE:

- Ignore **structure and nesting**
- Penalize semantically equivalent rewordings
- Are highly sensitive to word order and formatting
- Operate on flat sequences, unsuitable for hierarchical outputs

Thus, new approaches are needed that:

- Respect the **tree nature** of structured data
- Align **semantic units**, not just surface tokens
- Allow **partial credit** for partial matches or fuzzy equivalence

---

4. Related Work and Existing Evaluation Methods

Evaluating JSON outputs from language models is an emerging challenge, and several lines of work and tools have begun addressing it. Evaluation methods fall into three broad categories:

4.1 Schema Validation

- **Purpose**: Ensure the output is syntactically valid JSON and follows a predefined schema.
- **Common Techniques**:
  - JSON schema validation tools (e.g., using `jsonschema` or Pydantic)
  - Key presence and type checks
- **Use Cases**: Widely used in API generation, function calling, and form-based tasks.
- **Tools**: Promptfoo (`is-json` checks), DeepEval (schema compliance score)

4.2 Exact and Field-Level Matching

- **Whole-JSON Exact Match**: Simple but brittle; any small formatting change results in failure.
- **Key-Value Matching**: Compares individual fields across prediction and reference.
  - Supports partial scoring using per-field precision, recall, and F1.
- **Per-Field Custom Checks**: Different criteria for different field types (e.g., string vs numeric).

4.3 Semantic and Structural Comparison

- **Tree-Based Methods**:
  - Treat JSON as a tree and use **tree edit distance** to compare structure and content.
  - More robust to reordering and minor formatting changes.
- **Semantic Key-Value Matching**:
  - Flatten JSON into paths (e.g., `user.name: "Alice"`) and compare semantically using string similarity or embeddings.
  - Allows partial credit for close matches (e.g., "NewYork" ≈ "New York").
- **Embedding-Based Comparisons**:
  - Linearize JSON via tree traversal and compute ROUGE or BERTScore on the resulting text.
  - Captures content similarity while discarding rigid structure constraints.

4.4 Evaluation Tools and Benchmarks

- **Promptfoo** and **DeepEval**: Support both schema validation and semantic assertions.
- **Distancia**, **JYCM**, and other libraries: Compute tree or structure-based similarity.

- **JSONSchemaBench** and **StructuredRAG**: Benchmarks specifically for structured outputs; assess schema compliance and generation quality.

🔍 Takeaway:

There's a growing recognition that **flat metrics like BLEU and ROUGE are insufficient** for JSON evaluation. Instead, a combination of:

- **Schema checks**
- **Key-value accuracy**
- **Tree similarity**
- **Semantic matching**

...is required for robust and reliable evaluation. These ideas have begun to materialize in both academic work and real-world tooling for structured output generation.

---

5. Proposed Methodology

To address the shortcomings of traditional metrics in evaluating structured outputs, we propose a methodology that treats JSONs as **structured trees** rather than flat text. Our approach consists of two complementary strategies:

1. **Tree-Based Linearization and Traversal-Based Evaluation**
2. **Node-to-Node Semantic Similarity**

These strategies are designed to be **schema-agnostic**, allowing evaluation of both schema-constrained and schema-free outputs.

5.1 Tree-Based Linearization

🔍 Motivation

Standard evaluation metrics like ROUGE and BERTScore operate on sequences, not trees. However, JSON is inherently a **hierarchical data structure**. To bridge this gap, we transform each JSON into a **linear string** via tree traversals. This preserves structural order while enabling the use of well-understood NLP evaluation metrics.

🧩 How It Works

We parse each JSON into a tree structure and apply three common traversal orders:

- **Pre-order (Root → Left → Right)**

  Preserves top-down intent of JSONs, listing parent keys before children.

- **Post-order (Left → Right → Root)**

  Captures nested details before wrapping them in parent structure.

- **In-order (Left → Root → Right)**

  Although more common for binary trees, it offers a middle-ground sequence for balancing hierarchical and terminal nodes.

Each traversal results in a **flattened string** that preserves structural cues.

🧪 Example

For input JSON:

```
{"a": [1, 2, 3], "b": {"c": 4, "d": 5}}
```

Tree representation:

```
{}
  a
    []
      1
      2
      3
  b
    {}
      c
    4
   d
    5
```

Traversals yield sequences like:

- **Pre-order**: `{}` `a` `[]` `1` `2` `3` `b` `{}` `c` `4` `d` `5`
- **Post-order**: `1` `2` `3` `[]` `a` `4` `c` `5` `d` `{}` `b` `{}`
- **In-order**: `1` `2` `3` `[]` `a` `4` `c` `5` `d` `{}` `b` `{}`

Once linearized, we compare these sequences using:

- **ROUGE-1, ROUGE-2, ROUGE-L**: To measure n-gram and longest-match overlap.
- **BERTScore**: To capture semantic similarity between linearized key-value strings.

🌳 Real-World Example

Consider:

Prediction:

```
{
  "document": {
    "title": "Natural Language Processing",
    "sections": [
      {"header": "Introduction", "content": "NLP is a field of AI."},
      {"header": "Applications", "content": "NLP is used in chatbots and translati
on."}
    ]
  }
}
```

Reference:

```
{
  "title": "Natural Language Processing",
  "content": {
    "Introduction": "NLP is a field of AI.",
    "Applications": "NLP is used in chatbots and translation."
  }
}
```

Their **pre-order traversals** become:

- Prediction:

  `{}` `document` `{}` `title` `Natural Language Processing` `sections` `[]` `{}` `header` `Introduction` `content` `NLP is a field of AI.` `{}` `header` `Applications` `content` `NLP is used in chatbots and translation.`

- Reference:

  `{}` `title` `Natural Language Processing` `content` `{}` `Introduction` `NLP is a field of AI.` `Applications` `NLP is used in chatbots and translation.`

These traversals are then evaluated with ROUGE and BERTScore.

5.2 Node-to-Node Semantic Similarity

🔍 Motivation

Traversal-based evaluation loses granularity. A good local match might be drowned out by global structural differences. To solve this, we compare **individual nodes** of the predicted and reference trees directly.

🧩 Method

For every node in the reference JSON tree, we compute its similarity to **all nodes** in the predicted tree using BERT-based embeddings.

We calculate three scores:

- **Maximum similarity**: Best semantic match for each node
- **Minimum similarity**: Worst-case deviation
- **Average similarity**: Overall alignment quality

This gives a fine-grained view of how well individual facts or fields are reproduced—even if misplaced in the structure.

Mathematically, for nodes $ri \in R, r_i \in R$ and $pj \in P, p_j \in P$:

$$\mathrm{Sim}(r_i, P) = \max_j \left(\mathrm{cosine}(\mathrm{BERT}(r_i), \mathrm{BERT}(p_j))\right)$$

The **average similarity** across all reference nodes provides a score for prediction quality:

$$\mathrm{AvgSim}(R, P) = \frac{1}{|R|} \sum_{i=1}^{|R|} \mathrm{Sim}(r_i, P)$$

🎯 Benefits

- Handles semantic shifts ("NLP" ≈ "Natural Language Processing")
- Robust to structural reordering
- Offers both **summary-level** and **granular** performance insights

---

Together, these methods offer a **hybrid evaluation strategy**:

- Traversals for global structure + surface alignment
- Node-to-node BERT for local semantic accuracy

In the next section, we describe how we constructed a dataset and ran experiments using these methods across models.

---

6. Experimental Setup

To rigorously evaluate our proposed methodologies, we designed a structured pipeline comprising **dataset construction**, **model selection**, **JSON transformation**, and **metric evaluation**. This section outlines how we built our test set, generated outputs using multiple models, and applied our evaluation techniques to assess model performance.

6.1 Dataset Construction

📄 Source Content

Our input data consisted of **Wikipedia pages** covering a variety of factual and descriptive topics. These pages were selected for their rich information content.

Only the first 5000 tokens were used to construct the dataset, while preserving sentence boundaries.

🧠 Reference Generation with LLM

We used **Qwen2.5-14B-Instruct (Int-8 Quant)**, a high-performing instruction-tuned model, to convert Wikipedia pages to structured representations (JSONs) . Each page was passed through a custom prompt asking the model to **structure the content as a JSON document**.

These outputs were treated as **pseudo-ground-truth references**, under the assumption that the larger model was more capable of producing high-quality structured data.

⚠️ Data Filtering

To make evaluation tractable and fair across models:

1. We used the same Wikipedia inputs with two smaller models:
    - **Phi-3-mini-128k-instruct**
    - **Qwen2.5-3B-Instruct**
2. We discarded any samples where:
    - The output was not valid JSON
    - The structure was drastically malformed
    - One or more models failed to generate a parseable output

This resulted in a **clean evaluation set of 340 examples**, each containing:

- A Wikipedia source
- A high-quality reference JSON (from Qwen-14B)
- Predictions from Qwen-3B and Phi-3B

6.2 JSON Transformation and Traversals

Each JSON (reference and prediction) was parsed into a **tree structure** using a recursive walk:

- Keys became internal nodes
- Values became leaves
- Arrays and objects were preserved recursively

We then generated **three traversal strings**:

- **Pre-order**
- **Post-order**
- **In-order**

Each string was treated as a flattened text representation of the structured data, allowing evaluation using conventional NLP similarity metrics.

6.3 Models Evaluated

Model	Size	Source
Qwen2.5-14B	14B	Alibaba
Qwen2.5-3B-Instruct	3B	Alibaba
Phi-3-mini-128k-instruct	3B	Microsoft

- Qwen-14B outputs were treated as reference-quality.
- Qwen-3B and Phi outputs were evaluated against these references.

6.4 Evaluation Metrics Applied

We applied the following metrics across all examples:

✅ Traversal-Based Evaluation

- **ROUGE-1**, **ROUGE-2**, **ROUGE-L**

  Applied to each of the traversal strings (pre/post/in-order).

- **BERTScore (F1, Precision, Recall)**

Captures semantic similarity in key-value contexts.

🧠 Node-Level Semantic Similarity

- Each node in the reference was compared with all nodes in the predicted tree.
- We recorded:
  - **Maximum similarity** (best local match)
  - **Minimum similarity** (worst-case divergence)
  - **Average similarity** (overall alignment quality)

All semantic similarity was computed using **BERT embeddings** and **cosine similarity**.

🧪 Summary

This setup enabled us to:

- Test different **evaluation paradigms** (exact match vs. semantic alignment)
- Compare models of varying sizes and architectures
- Understand the effect of **structure-preserving evaluation techniques** across tasks

Next, we'll present detailed results from these evaluations—including ROUGE, BERTScore, and node-to-node similarity—and compare them against human-aligned judgments from GPT-4o-mini.

7. LLM-as-a-Judge: Model-Aligned Evaluation

Given the lack of human-annotated reference scores for structured outputs, we adopt a **"model-as-a-judge" approach**, using **GPT-4o-mini** to simulate human-level evaluation. This method provides a grounded benchmark to assess how well our automated metrics reflect real-world semantic correctness, structure, and utility.

7.1 Motivation

Automated metrics like ROUGE and BERTScore offer valuable signal, but they may:

- Miss deeper **semantic nuances**
- Penalize harmless **structural variations**
- Reward shallow matches over truly **meaningful information**

By using a state-of-the-art LLM to **rate the predictions against references**, we obtain an approximation of **human judgment**, which we can then use as a benchmark to validate our automated methods.

7.2 Evaluation Rubric

We prompt GPT-4o-mini to score each predicted JSON output against the reference using the following **4-point rubric**, with scores ranging from **1 (poor)** to **5 (perfect)** on each criterion:

Criterion 1: Semantic Equivalence

- Does the output convey the same meaning as the reference?
- Are all key concepts represented accurately?

Criterion 2: Structural Hierarchy and Organization

- Is the information organized in a logical and meaningful structure?
- Are nesting and relationships consistent with the reference?

Criterion 3: Completeness

- Are all relevant fields from the reference included?
- Are any critical components missing or extraneous?

Criterion 4: Data Consistency and Correctness

- Are types, key formats, and values used consistently?
- Does the output follow coherent and usable formatting?

7.3 Prompt Format

Each comparison was framed with:

```
Reference:
<reference JSON here>

Prediction:
<predicted JSON here>

Evaluate the prediction on the following criteria: ...
Return a JSON with four scores between 1 and 5.
```

We parsed the scores automatically and aggregated them into:

- Per-criterion averages
- Overall average score (mean of 4 criteria)
- Correlation with metric scores (see below)

---

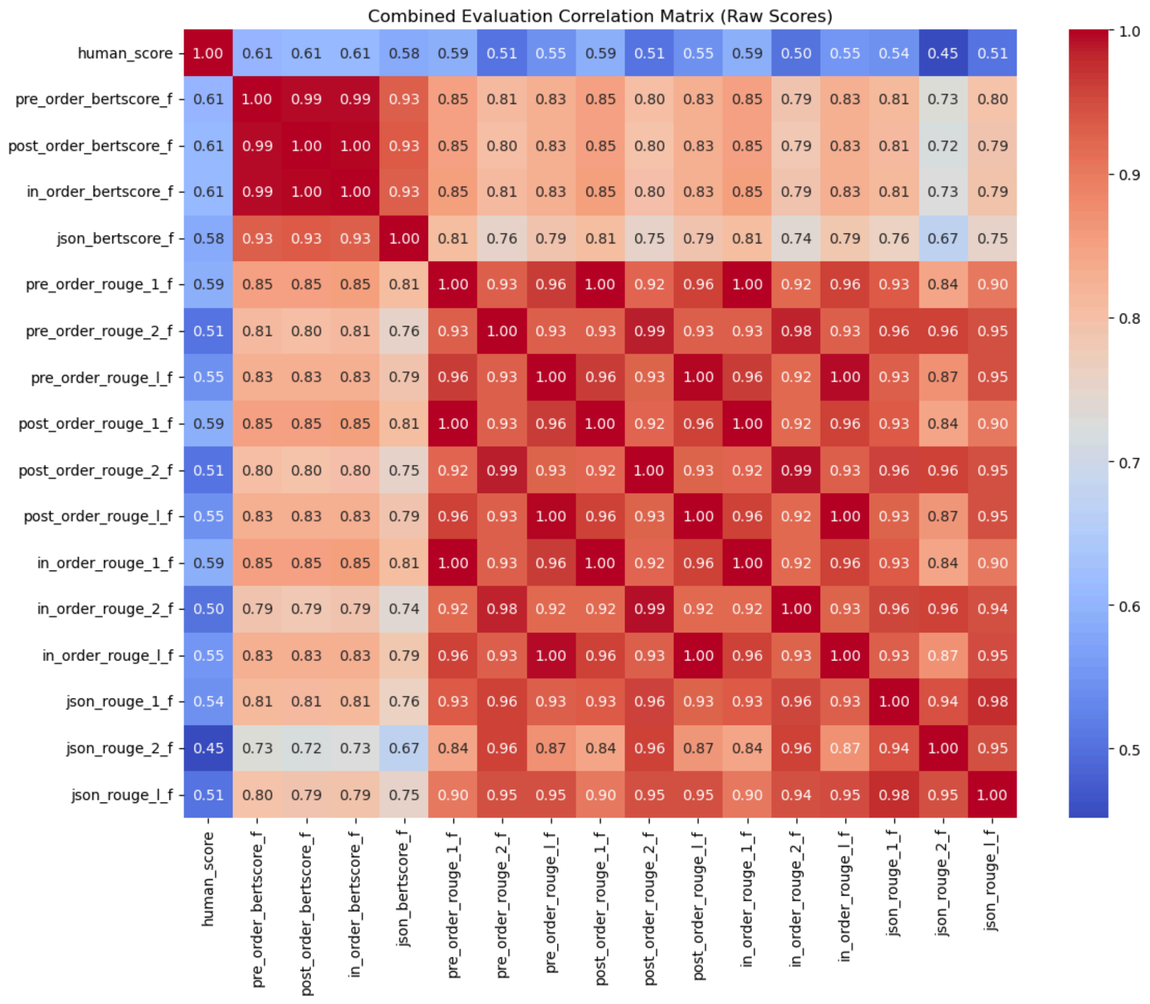
8. Results and Analysis

Below, we present all the metrics as reported in our scaled and raw evaluation documents, and we discuss their correlation values. We do not correlate node-node similarity due to the unoptimized

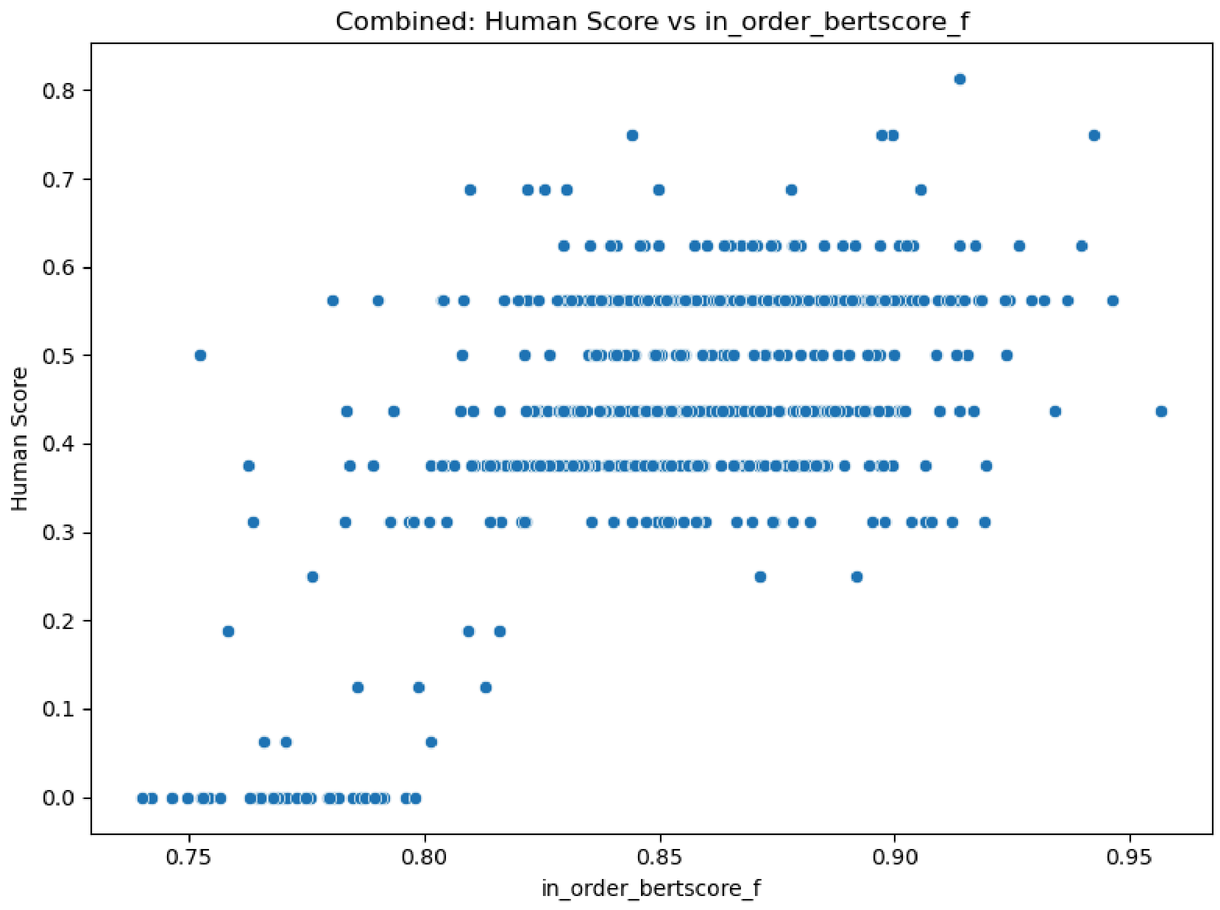
---

8.2. Raw Scores

A. Combined Evaluation (Raw Scores)



	human_score	pre_order_bertscore_f	post_order_bertscore_f	in_order_bertscore_f	json_bertscore_f	pre_order_rouge_1_f	pre_order_rouge_2_f	pre_order_rouge_l_f	post_order_rouge_1_f	post_order_rouge_2_f	post_order_rouge_l_f	in_order_rouge_1_f	in_order_rouge_2_f	in_order_rouge_l_f	json_rouge_1_f	json_rouge_2_f	json_rouge_l_f
human_score	1.00	0.61	0.61	0.61	0.58	0.59	0.51	0.55	0.59	0.51	0.55	0.59	0.50	0.55	0.54	0.45	0.51
pre_order_bertscore_f	0.61	1.00	0.99	0.99	0.93	0.85	0.81	0.83	0.85	0.80	0.83	0.85	0.79	0.83	0.81	0.73	0.80
post_order_bertscore_f	0.61	0.99	1.00	1.00	0.93	0.85	0.80	0.83	0.85	0.80	0.83	0.85	0.79	0.83	0.81	0.72	0.79
in_order_bertscore_f	0.61	0.99	1.00	1.00	0.93	0.85	0.81	0.83	0.85	0.80	0.83	0.85	0.79	0.83	0.81	0.73	0.79
json_bertscore_f	0.58	0.93	0.93	0.93	1.00	0.81	0.76	0.79	0.81	0.75	0.79	0.81	0.74	0.79	0.76	0.67	0.75
pre_order_rouge_1_f	0.59	0.85	0.85	0.85	0.81	1.00	0.93	0.96	1.00	0.92	0.96	1.00	0.92	0.96	0.93	0.84	0.90
pre_order_rouge_2_f	0.51	0.81	0.80	0.81	0.76	0.93	1.00	0.93	0.93	0.99	0.93	0.93	0.98	0.93	0.96	0.96	0.95
pre_order_rouge_l_f	0.55	0.83	0.83	0.83	0.79	0.96	0.93	1.00	0.96	0.93	1.00	0.96	0.92	1.00	0.93	0.87	0.95
post_order_rouge_1_f	0.59	0.85	0.85	0.85	0.81	1.00	0.93	0.96	1.00	0.92	0.96	1.00	0.92	0.96	0.93	0.84	0.90
post_order_rouge_2_f	0.51	0.80	0.80	0.80	0.75	0.92	0.99	0.93	0.92	1.00	0.93	0.92	0.99	0.93	0.96	0.96	0.95
post_order_rouge_l_f	0.55	0.83	0.83	0.83	0.79	0.96	0.93	1.00	0.96	0.93	1.00	0.96	0.92	1.00	0.93	0.87	0.95
in_order_rouge_1_f	0.59	0.85	0.85	0.85	0.81	1.00	0.93	0.96	1.00	0.92	0.96	1.00	0.92	0.96	0.93	0.84	0.90
in_order_rouge_2_f	0.50	0.79	0.79	0.79	0.74	0.92	0.98	0.92	0.92	0.99	0.92	0.92	1.00	0.93	0.96	0.96	0.94
in_order_rouge_l_f	0.55	0.83	0.83	0.83	0.79	0.96	0.93	1.00	0.96	0.93	1.00	0.96	0.93	1.00	0.93	0.87	0.95
json_rouge_1_f	0.54	0.81	0.81	0.81	0.76	0.93	0.96	0.93	0.93	0.96	0.93	0.93	0.96	0.93	1.00	0.94	0.98
json_rouge_2_f	0.45	0.73	0.72	0.73	0.67	0.84	0.96	0.87	0.84	0.96	0.87	0.84	0.96	0.87	0.94	1.00	0.95
json_rouge_l_f	0.51	0.80	0.79	0.79	0.75	0.90	0.95	0.95	0.90	0.95	0.95	0.90	0.94	0.95	0.98	0.95	1.00



- **BERTScore Metrics:**
  - Pre-order: **0.6058**
  - Post-order: **0.6080**
  - In-order: **0.6073**
  - JSON: **0.5816**
- **ROUGE Metrics:**
  - Pre-order ROUGE-1: **0.5901**
  - Pre-order ROUGE-2: **0.5070**
  - Pre-order ROUGE-L: **0.5454**
  - Post-order ROUGE-1: **0.5901**

- Post-order ROUGE-2: **0.5069**
- Post-order ROUGE-L: **0.5464**
- In-order ROUGE-1: **0.5901**
- In-order ROUGE-2: **0.5022**
- In-order ROUGE-L: **0.5500**
- JSON ROUGE-1: **0.5428**
- JSON ROUGE-2: **0.4520**
- JSON ROUGE-L: **0.5061**

B. Qwen Evaluation (Raw Scores)

- **BERTScore Metrics:**
  - Pre-order: **0.1794**
  - Post-order: **0.1773**
  - In-order: **0.1834**
  - JSON: **0.1258**
- **ROUGE Metrics:**
  - Pre-order ROUGE-1: **0.2753**
  - Pre-order ROUGE-2: **0.2956**
  - Pre-order ROUGE-L: **0.2880**
  - Post-order ROUGE-1: **0.2753**
  - Post-order ROUGE-2: **0.2988**
  - Post-order ROUGE-L: **0.2869**
  - In-order ROUGE-1: **0.2753**
  - In-order ROUGE-2: **0.2972**
  - In-order ROUGE-L: **0.2964**
  - JSON ROUGE-1: **0.2994**

- JSON ROUGE-2: **0.2907**
- JSON ROUGE-L: **0.2921**

C. Phi Evaluation (Raw Scores)

- **BERTScore Metrics:**
  - Pre-order: **0.7472**
  - Post-order: **0.7529**
  - In-order: **0.7497**
  - JSON: **0.7366**
- **ROUGE Metrics:**
  - Pre-order ROUGE-1: **0.7083**
  - Pre-order ROUGE-2: **0.6189**
  - Pre-order ROUGE-L: **0.6563**
  - Post-order ROUGE-1: **0.7083**
  - Post-order ROUGE-2: **0.6183**
  - Post-order ROUGE-L: **0.6585**
  - In-order ROUGE-1: **0.7083**
  - In-order ROUGE-2: **0.6151**
  - In-order ROUGE-L: **0.6594**
  - JSON ROUGE-1: **0.6513**
  - JSON ROUGE-2: **0.5532**
  - JSON ROUGE-L: **0.6103**

8.2. Min-Max Scaled Scores

We also min max scale al scores as different metrics have different ranges.

A. Phi Evaluation (Scaled)

- **BERTScore Metrics:**
  - Pre-order: **0.7472**
  - Post-order: **0.7529**
  - In-order: **0.7497**
  - JSON: **0.7366**
- **ROUGE Metrics:**
  - Pre-order ROUGE-1: **0.7083**
  - Pre-order ROUGE-2: **0.6189**
  - Pre-order ROUGE-L: **0.6563**
  - Post-order ROUGE-1: **0.7083**
  - Post-order ROUGE-2: **0.6183**
  - Post-order ROUGE-L: **0.6585**
  - In-order ROUGE-1: **0.7083**
  - In-order ROUGE-2: **0.6151**
  - In-order ROUGE-L: **0.6594**
  - JSON ROUGE-1: **0.6513**
  - JSON ROUGE-2: **0.5532**
  - JSON ROUGE-L: **0.6103**

B. Qwen Evaluation (Scaled)

- **BERTScore Metrics:**
  - Pre-order: **0.1794**
  - Post-order: **0.1773**
  - In-order: **0.1834**
  - JSON: **0.1258**
- **ROUGE Metrics:**

    - Pre-order ROUGE-1: **0.2753**
    - Pre-order ROUGE-2: **0.2956**
    - Pre-order ROUGE-L: **0.2880**
    - Post-order ROUGE-1: **0.2753**
    - Post-order ROUGE-2: **0.2988**
    - Post-order ROUGE-L: **0.2869**
    - In-order ROUGE-1: **0.2753**
    - In-order ROUGE-2: **0.2972**
    - In-order ROUGE-L: **0.2964**
    - JSON ROUGE-1: **0.2994**
    - JSON ROUGE-2: **0.2907**
    - JSON ROUGE-L: **0.2921**

C. Combined Evaluation (Scaled)

- **BERTScore Metrics:**
    - Pre-order: **0.5207**
    - Post-order: **0.5292**
    - In-order: **0.5313**
    - JSON: **0.4570**
- **ROUGE Metrics:**
    - Pre-order ROUGE-1: **0.5259**
    - Pre-order ROUGE-2: **0.4895**
    - Pre-order ROUGE-L: **0.4994**
    - Post-order ROUGE-1: **0.5259**
    - Post-order ROUGE-2: **0.4920**
    - Post-order ROUGE-L: **0.4997**

- In-order ROUGE-1: **0.5259**
- In-order ROUGE-2: **0.4898**
- In-order ROUGE-L: **0.5049**
- JSON ROUGE-1: **0.4966**
- JSON ROUGE-2: **0.4446**
- JSON ROUGE-L: **0.4661**

8.3. Analysis: Which Metric Correlates Better?

- **BERTScore Metrics:**
  - Across both scaled and raw evaluations, the pre-, post-, and in-order BERTScore values consistently show high correlation with LLM judgments. In the raw combined evaluation, these metrics are around **0.606–0.608**, which is slightly higher than the JSON-based variant (**0.5816**).
  - In the scaled Phi evaluation, the BERTScore correlations are exceptionally high (around **0.74–0.75**), although note that the Qwen scaled values are much lower. The combined scaled metrics moderate these differences, with pre-/post-/in-order BERTScore correlations averaging around **0.52–0.53**.
- **ROUGE Metrics:**
  - ROUGE-based correlations are also robust but generally slightly lower than BERTScore. In the combined raw evaluation, ROUGE-1 is about **0.5901** while ROUGE-2 and ROUGE-L are lower (around **0.5070–0.5500**). For the combined scaled evaluation, ROUGE-1 values are around **0.5259**, with ROUGE-2 and ROUGE-L dropping to around **0.4895–0.5049**.
  - Among the ROUGE metrics, the JSON variants tend to have the lowest correlations.
- **Comparison:**
  - **Traversal-based BERTScore metrics (pre-, post-, in-order)** consistently yield the highest correlations with LLM judgments—especially in the raw

combined evaluation (≈0.606–0.608) compared to ROUGE metrics (≈0.507–0.590).

  - **ROUGE metrics** still provide useful signals, particularly ROUGE-1, but overall, the semantic sensitivity of BERTScore makes it a slightly better indicator of quality as perceived by LLM judges.

8.4. Summary

- **Best Correlating Metrics:**

  Traversal-based BERTScore (pre-, post-, and in-order) achieves the highest correlation with LLM-as-a-judge scores, with raw evaluation values around **0.606–0.608** and scaled values in the high **0.52–0.53** range in the combined reports.

- **ROUGE Metrics Performance:**

  ROUGE-1 scores are competitive (raw: ≈0.5901; scaled: ≈0.5259) but ROUGE-2 and ROUGE-L tend to be lower, particularly in the JSON-based variants.

- **Traversal Order Impact:**

  Differences among pre-, post-, and in-order traversals are minimal, indicating that the evaluation framework is robust across various traversal orders. However, metrics computed on JSON representations (without specific traversal order) generally correlate lower than those computed on pre-, post-, or in-order sequences.

In conclusion, the evidence suggests that **traversal-based BERTScore metrics**—which leverage tree linearizations of structured outputs—correlate better with human-like (LLM) judgments compared to ROUGE-based metrics and comparing raw JSONs. This finding supports the adoption of semantic evaluation methods for structured output evaluation in NLP applications.

8.5 Traversal Case Study: Real Output Comparison

We examined outputs of both models on a Wikipedia-derived task (e.g., "Explain NLP concepts") using pre-order traversal.

Traversal Sample Output (Prediction vs Reference):

- **Prediction (Phi-3-mini)**:

  `{}` `document` `{}` `title` `"Natural Language Processing"` `sections` `[]` `{}` `header` `"Introduction"` `content` `"NLP is a field of AI."` ...

- **Reference**:

  `{}` `title` `"Natural Language Processing"` `content` `{}` `"Introduction"` `"NLP is a field of AI."` `"Applications"` `"NLP is used in chatbots..."`

ROUGE Scores:

Traversal	ROUGE-1	ROUGE-2	ROUGE-L
Pre-order	0.905	0.609	0.905
Post-order	0.905	0.609	0.857
In-order	0.905	**0.667**	0.857

Even with different nesting strategies, traversal-based evaluation captures semantic overlap well.

🔍 Summary of Model Performance

Metric Type	Best Model
ROUGE (Structure)	Qwen2.5-3B
BERTScore (Semantics)	Phi-3-mini
Node Similarity (Consistency)	Qwen2.5-3B

💡 Takeaway:

- **Qwen2.5-3B** excels at structural fidelity and consistent coverage.
- **Phi-3-mini** excels at isolated high-quality outputs with strong semantic alignment.
- **Traversal order has minor effects**, with **pre-order** generally best for structural tasks.

9. Challenges Faced

While developing a robust evaluation framework for structured outputs in NLP, we encountered several challenges—ranging from dataset construction to the design of reliable metrics. These difficulties helped shape our methodology and highlight important gaps in current research infrastructure.

9.1 Lack of Existing JSON Datasets

A major obstacle was the **absence of standardized datasets** designed for evaluating JSON generation. Unlike text classification or summarization tasks, there are very few public benchmarks where both:

- The **input** is unstructured (e.g., Wikipedia pages), and
- The **output** is structured (e.g., JSON documents)

This forced us to adopt a **self-supervised data generation strategy**, using a large model (Qwen2.5-14B-Instruct) to create reference outputs.

9.2 Noisy Model Outputs from Smaller Models

When generating predictions using smaller models (Phi-3-mini and Qwen2.5-3B), we found:

- A significant portion of outputs were **malformed or invalid JSON**
- Many outputs were **structurally misaligned** with the references
- Common issues included:
  - Missing keys
  - Incorrect nesting
  - Arrays vs objects mismatches

This necessitated aggressive **filtering** of samples, reducing our dataset from hundreds to just **340 high-quality examples**.

9.3 Evaluation Metric Misalignment

Standard metrics (ROUGE, BLEU, even BERTScore) are not designed for structured data:

- They are **overly sensitive to formatting** and surface form
- They **ignore key order** or **semantic nesting**
- They fail to capture **partial correctness** in deeply nested trees

This inspired our **tree traversal** and **node-level similarity** approaches, but these too brought trade-offs (e.g., traversal-induced ambiguity).

9.4 Complexity of Tree Processing

While the JSON-to-tree conversion sounds straightforward, it introduced subtleties:

- Differentiating between **leaf vs internal** nodes consistently
- Deciding whether to treat **arrays as sequences** or semantic groupings
- Constructing **uniform traversal outputs** across models with slightly different key orders or data formatting

Designing these representations to be **metric-friendly yet structure-preserving** required multiple iterations.

9.5 Semantic Ambiguity in Human Evaluation

Evaluating structured data manually (or via GPT-4o-mini) revealed further challenges:

- Two outputs could differ **vastly in structure**, yet express **identical meaning**
- Conversely, two outputs could look **lexically similar**, yet **omit key details**
- Even with a rubric, assigning consistent **numeric scores** was occasionally ambiguous

This highlights the importance of **multi-metric evaluation** and **LLM-based consensus scoring** to reduce bias.

9.6 Variability Across Traversal Strategies

We found that:

- **Pre-order** often worked best for ROUGE
- **Post-order** occasionally helped BERTScore
- **In-order** was less consistent, especially with non-binary trees

Choosing the "best" traversal was non-trivial, and we opted to **evaluate all three** for completeness.

⚠️ Summary

Challenge	Impact	Solution
No reference JSON datasets	Limited evaluation baseline	Used Qwen-14B as reference generator
Model output inconsistency	Reduced usable dataset size	Filtered outputs, enforced JSON constraints
Metric inadequacy	Poor alignment with human intuition	Designed traversal + node-based methods
Tree transformation complexity	Hard to normalize outputs	Developed consistent traversal schemes
Ambiguity in judgment	Inconsistent human/model scoring	Created a structured rubric for GPT-4o-mini

---

10. Conclusion

As large language models increasingly transition from open-ended text generation to structured data generation, the **need for robust evaluation of structured outputs**—especially in formats like JSON—has become both critical and underexplored.

In this project, we tackled the challenge of evaluating JSON outputs produced by LLMs, where correctness depends not just on surface similarity, but on:

- **Structural alignment** (hierarchical nesting, ordering, grouping),

- **Semantic equivalence** (keys and values with identical or near-identical meaning),
- **Completeness and data fidelity** (no missing or extraneous information),
- And **conformity to schema**, where applicable.

🔍 Summary of Contributions

1. **Problem Formalization**:

   We categorized evaluation into schema-based and schema-free tasks, identifying key metrics for structure, semantics, and completeness.

2. **Traversal-Based Evaluation Framework**:

   We introduced a novel tree-to-string linearization approach using **pre-, post-, and in-order traversals** to apply standard metrics like ROUGE and BERTScore to structured data in a meaningful way.

3. **Node-to-Node Semantic Similarity**:

   We proposed a **fine-grained evaluation method** using BERT embeddings to measure the semantic alignment of every predicted field against its best-matching reference node.

4. **Empirical Analysis Across Models**:

   Using outputs from **Phi-3-mini-128k-instruct** and **Qwen2.5-3B-Instruct**, we applied these metrics and uncovered strengths and trade-offs in their structural vs semantic fidelity.

5. **LLM-as-a-Judge Framework** *(in progress)*:

   We designed a rubric-based scoring system for GPT-4o-mini to act as a human-like evaluator, allowing us to assess how well our metrics align with practical human judgment.

🧠 Key Insights

- No single metric is sufficient—**structure, semantics, and completeness must all be considered.**

- **Pre-order traversal** offers a strong balance between hierarchy preservation and lexical match.
- **Phi-3-mini** showed strong semantic alignment but inconsistency in structure.
- **Qwen2.5-3B** showed more consistent formatting and better structural fidelity.
- Our **node-to-node similarity approach** captures partial credit and nuanced alignment better than ROUGE or BERTScore alone.

This work demonstrates that **evaluating structured outputs is not a solved problem**, and that **task-aware, structure-aware evaluation frameworks** are crucial for deploying LLMs in real-world systems where correctness is non-negotiable.

---

11. Future Work

While our proposed framework significantly improves the evaluation of structured outputs in NLP, several avenues remain open for further development. These range from advancing evaluation metrics to generalizing across data formats and improving human-model alignment.

🔁 11.1 Reference-Free Evaluation

Our current methodology requires a high-quality **reference JSON** to compare against. In the future, it would be useful to develop **reference-free evaluation strategies**, such as:

- **Self-consistency checks**: Compare multiple model generations for agreement.
- **Schema-conformance + confidence estimation**: Evaluate whether the output conforms to a schema and contains high-entropy content.
- **Language Model Critics**: Use an LLM not only to score predictions, but also to **explain errors or missing elements**—potentially as part of a training loop.

🧮 11.2 Handling Quantitative Fields & Lists

JSON outputs often contain **lists of numbers**, dates, or numeric attributes. Future work could explore:

- **Statistical similarity metrics**: Mean, variance, range comparisons over arrays.
- **Tolerant numerical matching**: Check if values are within acceptable margins (e.g., ±5%).

This would be particularly useful in scientific, financial, or educational settings where models extract structured data from numerical sources.

🧠 11.3 Graph-Based Representations

JSON structures are trees—but in many cases (e.g., knowledge graphs, citations, cross-references), the structure is closer to a **graph**.

Future directions include:

- **Graph edit distance** for looser, reference-free comparisons
- **Graph neural network embeddings** for comparing structural "shapes" semantically
- **Path2Vec / Tree2Vec**: Generating embeddings for nested paths and comparing via cosine similarity

---

🌍 11.4 Generalization Beyond JSON

Our methods are designed with JSON in mind, but can be extended to other structured languages:

- **HTML & XML**: Evaluate web generation, scraping, and parsing tasks
- **LaTeX**: Evaluate document formatting (e.g., equation extraction or paper generation)
- **YAML**: Common in configuration tasks and DevOps/NLP pipelines
- **Code / Abstract Syntax Trees (ASTs)**: For program synthesis and documentation generation

These domains would benefit from **structure-aware semantic metrics** that reward both syntactic correctness and functional fidelity.

🧪 11.5 Human-Centric Evaluation Design

Even with GPT-4o-mini as a surrogate judge, **human preferences** vary:

- Do users prefer compact or descriptive structures?
- Are nested outputs harder or easier to interpret?
- Can visualizations (like trees or diagrams) help evaluators spot differences?

Future work could involve **human studies** exploring:

- Usability of various output formats
- Cognitive load during structured data evaluation
- Agreement rates between LLMs and human evaluators

With the increasing deployment of LLMs in structured data generation tasks—from chatbots to APIs to data processing pipelines—advancing evaluation techniques is not just a research curiosity but a **practical necessity**. This project lays the foundation for structure-aware evaluation, and future work can carry it across modalities, domains, and languages.

---

12. References

1. Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). *BLEU: a method for automatic evaluation of machine translation*. In ACL.
2. Lin, C. Y. (2004). *ROUGE: A package for automatic evaluation of summaries*. In ACL Workshop.
3. OpenAI. (2023). *Function calling in GPT models*. Retrieved from https://platform.openai.com/docs/guides/function-calling
4. Confident AI. (2024). *DeepEval - LLM Evaluation Framework*. https://github.com/confident-ai/deepeval

5. Geng, R. et al. (2025). *JSONSchemaBench: Evaluating Constrained Decoding of Structured Outputs*. [Preprint]
6. Lin, B. Y., & Ng, V. (2020). *Tree-based Evaluation Metrics for Program Synthesis*. In ACL.
7. Sun, Y. (2023). *JYCM: JSON Your Configuration Manager*. https://github.com/cubicdaiya/jycm
8. Distancia Library. (2022). *JSON Tree Edit Distance Tools*. https://github.com/gabsens/Distancia
9. Li, H. et al. (2024). *Evaluating Structured Data Extraction with JSON Similarity Metrics*.
10. Zhang, S. et al. (2023). *StructuredRAG: Structured Output Evaluation for Retrieval-Augmented Generation*.
11. Vilar, D., Xu, J., & Ney, H. (2006). *Error Analysis of Statistical Machine Translation Output*. In LREC.
12. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In NAACL.!