Two Shots Are Enough: Reliable Constrained Generation with LLMs

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

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

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

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

***Abstract*—Despite exhibiting impressive performance across many complex tasks, Large Language Models (LLMs) often struggle to consistently produce structured responses, a key element in many applications such as agent orchestration. While template-based methods or in-context learning can improve structure compliance, constrained decoding and additional context have been shown to skew the distribution of the responses and may result in suboptimal performance. In this work, we show that the number of in-context examples is a key factor: in many settings, one to two shots are sufficient to reliably produce the predefined structures, while additional examples may degrade the response quality. Our analysis shows that this phenomenon is consistent across a wide range of datasets, models, and structures and indicates that providing this small number of shots achieves the best overall trade-off between structure compliance and response correctness while improving the results of constrained generation methods such as template-based approaches.**

***Index Terms*—Large Language Models, In-context Learning, Constrained Generation, Structured Output**

I. Introduction

In recent years, Large Language Models (LLMs) have yielded impressive results across many natural language processing (NLP) tasks [1]–[3], including knowledge retrieval [4], [5], reasoning [6], and code generation [7]. Consequently, many automated systems use LLMs as part of a multi-agent framework to leverage their NLP capabilities [8]–[10]. In many cases, the integration of LLMs into automated systems requires that their output adheres to a predefined structure, such as JSON objects, CSV files, or XML documents [11]–[13]. For instance, using external tools with function calling [14], [15], which are necessary to solve complex problems (e.g., interpreting code, web search), require specific arguments as input (e.g., a bash script). Similarly, extracting the answer of an LLM from the generated natural language text is key for automated evaluation (e.g., benchmarking of LLM abilities) [4], [6]. Indeed, when the models are not constrained to generate a specific format, the measures of the base task ability can get intertwined with their structure compliance ability (i.e., the aptitude to produce the answer in a specific format), resulting in inaccurate evaluations [16].

Two main approaches have been proposed to generate structured responses with LLMs: format-restricting instructions and constrained decoding. With the former, the model is prompted with instructions, formulated in natural language, describing the structure to follow (e.g., "format your response as a JSON object, with keys X and Y"). This approach can be placed into the broader category of Instruction-following, a widely studied topic [6], [17]–[19], and one of the core aspects of fine-tuning dialogue systems [20]. One of the most powerful instruction-following techniques is arguably the few-shot method, where the model is provided with in-context examples of the task to achieve [21]–[24]. Due to its efficacy, this approach is commonly used in many benchmarks when evaluating LLM performance on various tasks, see e.g., [4], [25]. However, instructing the model to follow a specific data structure does not guarantee that the model's output will adhere to this structure. For various reasons (e.g., ambiguity in the instruction, model inability, or large deviation from the training distribution), the model may apply a different formatting (e.g., double quotes instead of single quotes in a JSON object, renamed XML tags) or refuse to comply.

Constrained decoding, on the other hand, intervenes at an LLM's token sampling stage: when the model computes the probability distribution over its vocabulary, an external verifier prevents it from sampling a token that would induce a prohibited structure (e.g., an invalid JSON object) [26]. Many forms of external verifier approaches have been proposed, including templates, regular expressions, grammars, or automata-based techniques [27], [28]. The external verifier is said to intervene whenever the token sampled by the language model is rejected due to breaching one of the syntax constraints. While this approach almost guarantees the structure compliance of the answer, previous works have shown that these interventions may impair the LLMs' performance. The core issue with this approach stems from the invasiveness of the constrained decoder: when the external verifier rejects the model's preferred token, it may have to sample an alternative further down the tail of the token probability distribution, thereby distorting the model distribution [27], [29].

In this work, we investigate the combination of both approaches (instruction-following and constrained decoding) and, in particular, how in-context learning can alleviate the distortion problem. Therein, the task to be solved (e.g., producing a well-formatted JSON object) is presented to the model in one or more examples, which are included in the prompt. More precisely, we study the influence of the number of examples on structure compliance and its impact on the distortion effect to assess the optimal number of examples. Many benchmarks

using few-shot prompting techniques include three to ten examples by default [4], [5], [25], [30]. We argue that this choice may be suboptimal for multiple reasons. First, recent studies have shown that increased context size (an unavoidable consequence of additional examples) can negatively impact the performance of LLMs [31]–[33]. Additionally, by the nature of the transformer-based architecture, increasing the context size incurs additional computational cost [34] and latency [35]. Thus, a careful analysis of this trade-off may significantly improve the LLMs' performance when generating structured content. Our contributions are as follows:

- We propose an evaluation method to measure the impact of few-shot prompting on structure compliance. To achieve this, we divide the response tokens into structure (i.e., tokens that are necessary to the correct format of the answer) and content (i.e., tokens pertaining to the content of the answer), and by assessing their joint probability separately (Section III-A). Notably, this approach allows us to accurately measure the distortion effect of constrained generation under different conditions.
- We introduce several new datasets tailored toward assessing the aptitude of LLMs to produce structured content under various constraints (Section III-B). These datasets cover multiple base tasks (including reasoning, entity recognition) and structures (e.g., JSON, CSV, XML, HTML), and allow for automated evaluation of the LLM answers without the use of LLMs-as-judges.
- We study the impact of the number of in-context examples on both structure compliance and answer correctness, and show that one to two shots are enough to obtain the expected structure with very high probability (>95%) on many datasets and for many models. Moreover, we show that additional examples do not improve this probability, while having in some settings a noticeable negative impact on the overall model performance (Section IV). Finally, we show that the combination of few-shot prompting and constrained generation achieves the best of both worlds, guaranteeing structure compliance while minimizing probability distortion.

Our experimental results support these claims with LLMs of various families (e.g., Llama3.1, Qwen2.5, Mistral) and sizes (7 to 141 billion parameters) across a variety of base tasks and data structures.

II. Related work

a) Constrained decoding and Distortion: Previous works have investigated the impact of constrained decoding on the quality of the responses. Recently, the authors in [29] showed that constrained decoding, when used for generating JSON objects, significantly deteriorates the performance of language models on both reasoning and knowledge-based tasks. Similar observations were made by [27], where the authors showed that existing constrained decoding approaches commonly induce a distortion of the output distribution. If such a distortion is strong enough, it may skew the distribution of the model's answer, resulting in token misalignment and thus deteriorating the model's performance. They introduce DOMINO, which leverages automata and a speculative decoding strategy [36] to alleviate the distortion problem by minimizing the number of rejected tokens. However, the flexibility of the proposed approach may produce significantly longer answers from LLMs, resulting in considerable computational overhead. Furthermore, DOMINO is not guaranteed to reach perfect structure compliance in practice, as the model can still exceed its context window without generating the desired structure.

Compared to these works, our approach differs in multiple ways. First, we focus on the impact of few-shot prompting on structured content generation. We show that using two examples is often enough to minimize the distortion of constrained generation approaches, such as templates, to a negligible level, solving the token misalignment problems. Second, our evaluations do not rely on LLMs-as-judges, compared to e.g., [29]. Indeed, this approach has been shown to suffer from multiple biases, see [37]–[39]. Instead, we analyze the joint distributions of the tokens, carefully distinguishing between structure and content tokens. This evaluation led to several key observations, such as the saturation phenomenon associated with multiple examples (see Section IV).

b) Instruction-following for structured generation: Several benchmarks have been developed to investigate the structured generation capabilities of LLMs through instruction-following. While IFEval [17] and its extension in LiveBench [6] only contain a limited number of format-following instructions (e.g., "Entire output should be wrapped in JSON format."), more recent benchmarks have focused on format-restrictions [40], [41]. Our evaluation differs in multiple ways. First, our experiments cover a broader range of structures, and each format is associated with multiple base tasks to better capture the influence of each component. In addition, existing benchmarks often evaluate the output document's formatting using LLMs-as-judges (with e.g., GPT-4 acting as an external assessor of output quality). As aforementioned, multiple studies [37]–[39] have, however, raised concerns about the reliability of such an approach.

c) In-context learning with few-shot prompting: Few-shot learning is a well-established approach in the field of LLMs [23], [24], and is commonly used to improve model performance on various tasks, such as data augmentation [42], model augmentation [43], or knowledge updating [44]. This approach is also used in constrained generation, such as five-shot prompting in [27]. However, recent studies have shown that model performance can deteriorate with increasing context size [31], [45] or when including irrelevant information in the context window [32], [33]. Additionally, by the nature of the transformer-based autoregressive architecture, increasing the context size incurs additional computational costs. Our work investigates the impact of the number of examples on structure compliance and on the quality of the answers, and highlights the resulting trade-off.

USER. *Please list the first two Catalan numbers, by outputting a dictionary with keys "number i" for i between 1 and 2. Write your answer between [[and]].*

AGENT. Of course! Here's the requested dictionary output: [[{"number 1": 2 , "number 2": 1 }]].

ANSWER EVALUATION.

$$\begin{cases} P(\mathcal{C}) = P_{tok}(\ 2\ ) \times P_{tok}(\ 1\ ) \\ P(\mathcal{S}) = P_{tok}(\ \{\ ) \times P_{tok}(\ \text{“}\ ) \times P_{tok}(\ \text{number}\ ) \times \dots \end{cases}$$

Fig. 1: Illustration of token disentanglement with a toy prompt. The reasoning tokens are highlighted in blue, the structure tokens in red, and the content tokens in yellow. Only the probabilities of the contents tokens $\mathcal{C}$ and the structure tokens $\mathcal{S}$ are computed. P_{tok} denotes the conditional probability of each token, given the context prefix.

III. METHODS

A. Evaluating LLMs' structured outputs

The automated evaluation of LLM outputs is particularly challenging, as their answers are expressed in natural language. Outside LLMs-as-judges, which suffer from many biases, the most common approach to assess structured output, such as a JSON document, is (A) parsing the answer to identify the presence of well-formed JSON, followed by (B) the evaluation of said JSON and its comparison to the ground truth. However, this method, while principled, conflates structure compliance $\mathcal{S}$ (i.e., is the structure well-formed?) and content accuracy $\mathcal{C}$ (i.e., is the information contained within the structure correct?). For instance, a parsing error (e.g., due to an extra closing "}") may hide the fact that the content of the JSON (i.e., the key–value pairs) was correct. As a result, this approach conflates the model's ability to solve a base task from its structure-compliance ability, thus limiting the possibility to accurately pinpoint the source of the errors. Instead, we propose an alternative method that carefully partitions the answer's tokens depending on whether they pertain to structure or content, and computes their joint probability independently. Our approach measures for a more fine-grained analysis of LLM performance, allowing to pinpoint specific failures cases.

a) Reasoning tokens: In our evaluation, models are allowed to generate intermediate reasoning tokens that precede their final answer. This intermediate generation phase has been shown to be important to the model's performance [46], [47] and may include Chain-of-Thought (CoT), restating parts of the problem statement, or generating other common verbal interjections (e.g., "Sure, let us look at this problem."). These tokens are not taken into account in our proposed evaluation scheme and do not impact the structure compliance and content quality scores.

TABLE I: Available combinations of datasets and data structures. *N* denotes the number of questions in each dataset.

Base task	JSON	CSV	XML	HTML	STARS	N
Reasoning	✓	✓	✓	✓		50
Typos	✓	✓	✓	✓		100
Sequence	✓	✓	✓	✓		46
Entity		✓				99
MMLU	✓				✓	581
MMLUPro	✓				✓	119

b) Disentangling Structure and Content: Once the first token of the final answer (which is generally a delimitation token "[[" in our experiments– see Section III-C below) is generated, we compute the joint probability of the tokens of the structure and content, respectively. To achieve this, we proceed as follows. First, the entire expected answer (ground truth) is tokenized, and the tokens are partitioned into structure and content tokens. Content tokens are all the tokens pertaining to the answer of the base task (e.g., the response to a multiple-choice question, a corrected typo). Conversely, we consider as structure tokens all tokens necessary to satisfy the formatting requirements, which do not pertain to the content of the base task. This includes commas, colons, spaces, new-line characters, curly brackets, angle brackets, and quotation marks.

As the tokenization scheme depends on the model[1], the expected ground truth, i.e., the list of structure tokens and the list of content tokens, is computed for each model's tokenizer separately when evaluating it. For some datasets (e.g., Reasoning in JSON), the keys (e.g., "person 1") are also considered structure tokens, as these are pre-determined by the expected template. When the dictionary keys are not predetermined (e.g., for finding typos or entities in a text), they are considered content tokens, as finding them is part of the base task. Details regarding the structure tokens of each dataset and task can be found below. Second, the individual probability distribution of each token is computed, conditioned on the current context prefix (i.e., the concatenation of the reasoning tokens, followed by the prefix tokens of the expected answers), similar to [48], [49]. Finally, the joint probabilities of the structure tokens and the content tokens are calculated separately. Figure 1 illustrates this method.

B. Datasets for structure generation

In order to evaluate the LLMs' ability to produce structured content, we introduce a new collection of datasets. It includes several base tasks commonly used to evaluate Large Language Models, such as reasoning, knowledge, and entity recognition abilities. Each base task is available in various formats, including JSON, CSV, XML, HTML, and STARS (which leverages asterisk symbols, see below for details). Table I provides an

[1]Example of variations include: the presence of leading spaces before numbers, the splitting of numbers into individual tokens, the grouping of some special characters into a single token, etc.

overview of all base task datasets and their expected formats, while additional details on each dataset can be found below.

a) Knowledge retrieval: For knowledge base tasks, we use three subsets (high-school statistics, sociology, and virology) of the MMLU benchmark [4], considered among the hardest subtopics for LLM performance. This dataset consists of multiple-choice questions, where one of four provided answers is correct. Furthermore, we use a subset of the MMLU-Pro benchmark [5], which consists of a more difficult set of multiple-choice questions with 10 possible answers per task. Two possible structures are available for these datasets: STARS and JSON. For the STARS structure, the answer (a single capital letter) must be repeated five times, separated by an asterisk symbol (for instance, "`A*A*A*A*A`"). This structure is arbitrary and intentionally unlikely to be part of any LLM training set to assess out-of-distribution behavior. Consequently, the models' performance on this structure is expected to hint at their potential to generate unseen structures. For the JSON format, the LLM is asked to produce a JSON with a single "answer" key, whose value is the capital letter of the chosen answer, similar to [29].

b) Reasoning: The reasoning-based tasks were built from the Web of Lies, originally proposed by Big-Bench Hard [25] and LiveBench [6]. Each task consists of evaluating the truth value of a random Boolean function expressed in natural language. A set of statements about fictional people is proposed, and the binary truthfulness of three people needs to be determined. We evaluate models on four structures for this base task: JSON, CSV, XML, and HTML. For the JSON format, we require a dictionary with the keys "person 1|2|3" and the values: "yes|no". The CSV expects a header row ("person,candid"), followed by rows of person IDs and their truthfulness, separated by a comma. To format the response in valid XML, a tag for each person, with the attribute "person id" and their truthfulness as the text content, must be wrapped into an "`<answer>`" tag. For HTML, we expect an ordered list ("`<ol>`"). Each list item ("`<li>`") stands for a person (in order) and contains as its text value their truthfulness.

c) Named entity recognition: The entity recognition base task is based on the traditional CoNLL-2003 dataset [50]. It consists of segments of news stories and aims at recognizing named entities (e.g., places, people). The model's response is expected in a two-column CSV format, where each entity and its assigned category is to be written as a comma-separated row under an "entity,category" header.

d) Mathematical sequences: We introduce a new Mathematical sequence dataset, where the models are requested to generate the first elements of various mathematical sequences. These sequences include prime numbers, multiples of an integer, and named sequences (e.g., Catalan numbers). Four formats are available for this dataset. For CSV, the model must generate one row per answer value (with columns "index" and "number"). The XML variant must be formatted as a list of number tags, each with an "id" attribute and the value as its text content, all wrapped into an `<answer>` root tag. Similarly, for HTML, the answer must be an ordered list containing a list item tag for each value. Finally, in JSON, each key-value pair maps the position of the number in the sequence to its value (e.g., "`{..., "number 7": 17}`").

e) Typos: We also introduce a new Typos dataset, which contains a sample of texts with multiple typographical errors. The text snippets are based on news article excerpts from the Guardian, collected using the provided API[2], where multiple words were randomly selected and then altered to exhibit typographic errors. The model's objective is to find and correct the misspelled words. Four formats are available for this dataset, including a JSON dictionary and a two-column CSV, mapping each erroneous word to its correction. For XML, each misspelled word is contained as the "word" attribute value of a `<misspelled>` tag and the corrected version as its text content. For the HTML format, each misspelled word is contained within an item of an ordered list and is followed by a hyphen and its corrected version.

C. Experiments

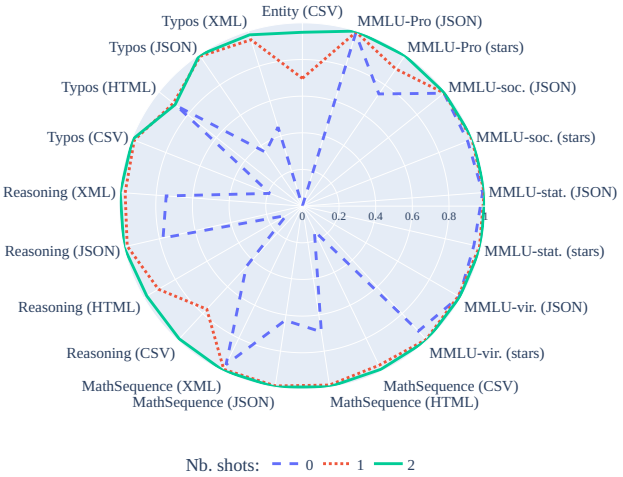
In the following experiments, we investigate the aptitude of various LLMs to produce structured content under different conditions, by varying datasets, structures, number of in-context examples, and presence of constrained generation.

a) Instructions and few-shot prompting: The general structure of the prompt was organized as follows. In the first part, the models are instructed to follow a specific structure when generating their response to the base task. The structure includes specific dictionary keys, column headers, tags, attributes, etc., depending on the dataset (see Section III-B). This instruction is followed by examples, each consisting of a question and the corresponding well-structured answer. To measure the impact of few-shot in-context learning, we vary the number of examples shown to the model in the prompt, up to four shots. The zero-shot setting corresponds to pure format-restricting instruction-following: the prompt contains only the description of the expected structure, followed by the base task to be solved. The few-shot examples are randomly selected from the remaining dataset (while ensuring no overlap with the evaluated task).

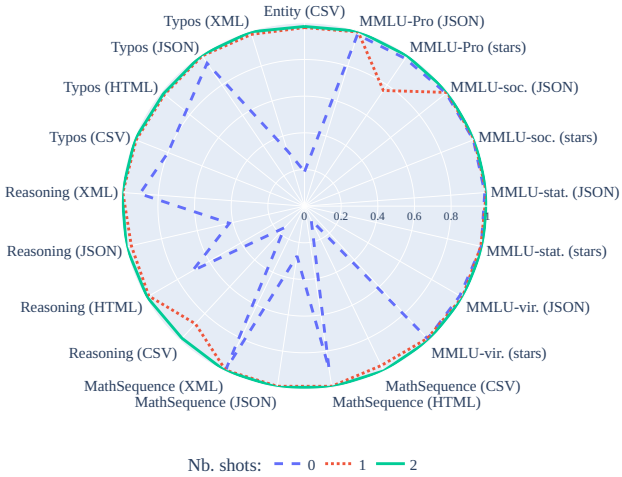
b) Reasoning tokens: As discussed in Section III-A, we allow each model to generate intermediate reasoning tokens prior to their final answer, which starts with delimitation tokens (here, two square brackets "`[[`"). If the model exceeds the limit of 10,000 intermediary tokens without generating the delimitation token, its reasoning phase is considered failed. Failure during the reasoning phase is extremely rare and was observed to occur less than 1% of the time in our few-shots experiments. Most failures occurred with smaller models (<10B parameters) and were generally caused by either a generation loop or a refusal to answer the base task.

c) Constrained decoding: To investigate the impact of constrained decoding and the distortion effect in particular, we used both pure instruction-following and instruction-following combined with a template-based approach. More specifically,

[2]open-platform.theguardian.com/access.



(a) Median joint probability for Llama3.1-8B.



(b) Median joint probability for Llama3.1-70B.

Fig. 2: Structure compliance scores (median of the joint probability assigned to the structure tokens) for Llama3.1 models at two scales (8|70)B on all datasets with zero, one, and two-shot prompting.

after the reasoning tokens (see Section III-A), all the structure tokens are imposed to match the ground truth, and only the content tokens are sampled by the LLMs. While this approach is guaranteed to produce the expected structure, it may distort the distribution of the content tokens, and thus the evaluation of the answers (see Section IV).

d) Models: In our experiments, we compare various models of sizes and families[3]. The models were selected based on their instruction-following scores on the IFEval benchmark [51], as reported on the Hugging Face Open LLM Leaderboard[4]. We exclusively selected models with available open weights (as we require access to the full logits) and omitted third-party variants (e.g., fine-tuned and merged models). This process resulted in models from the Llama3.1 [52], Qwen2.5 [53], and Mistral families. For Llama3.1, we use the 8 and 70 billion parameter variants. For Qwen2.5, we use the 72 billion variant for the main evaluation and compare the effect across model scales additionally for the 7, 14, and 32 billion parameter variants. From the Mistral family, we use the Mistral-Large-2407 (123 billion) and the Ministral-8B variants. To diversify the set of evaluated models, we also include a model using a different architecture (Mixture of Experts) Mixtral-8x22B [54], with 141 billion parameters.

e) Metrics: To assess the performance of the models, we use the Structure Compliance Error (SCE) and the Content Quality (CQ). The SCE is defined as $1 - P(\mathcal{S})$, i.e., the probability that the model outputs a structure that differs from the ground truth (see Section III-A). It is important to note that this error is an upper bound of the model's probability of producing an invalid structure, as the ground truth is more constraining (e.g., the order of the keys of a dictionary is fixed in the ground truth). The CQ is defined as the proportion of samples with a geometric average of $\mathcal{C}$ above a threshold of 95%. Using the geometric average allows us to take into account the compounding effect of joint probabilities.

$$\text{CQ} \propto \#\left\{\text{Content } \mathcal{C},\ \text{s.t.}\ \sqrt[\text{len}(\mathcal{C})]{\prod_{\text{tok}\in\mathcal{C}} P(\text{tok})} > 0.95\right\}$$

Thus, CQ represents the proportion of contexts that had a correct answer (with probability > 0.95). Importantly, our experiments showed that CQ is robust to threshold variations and that our analysis holds for various thresholds.

IV. Results

We present here the main results of our experiments, organized into three main findings.

A. Finding 1: Two shots are enough for structure compliance

Figure 3 reports the SCE for models of two families at two scales, for up to four shots, on five representative datasets. We observe that most models reach near-maximal structure compliance (i.e., minimal SCE) with two in-context examples. Additional examples generally do not appear to yield significant improvement in the structure compliance scores for most models. We can observe that in particular instances, the error reduces slightly further when provided with a third shot in one of the datasets (e.g., the Ministral-8B model for Reasoning in XML). However, in other situations (e.g. Llama3.1-70B for the same dataset), the error increases with more than two shots. It is important to note that in both of these cases, the SCE for $N = 2$ is already below 0.001 – i.e., less than 0.1% probability for the model to produce an invalid structure. Therefore, we hypothesize that such variations may be due to the inherent randomness induced by the reasoning tokens, such as the reasoning steps of Chain-of-Thought.

[3]For all models, we use their instruction fine-tuned variant.

[4]huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.

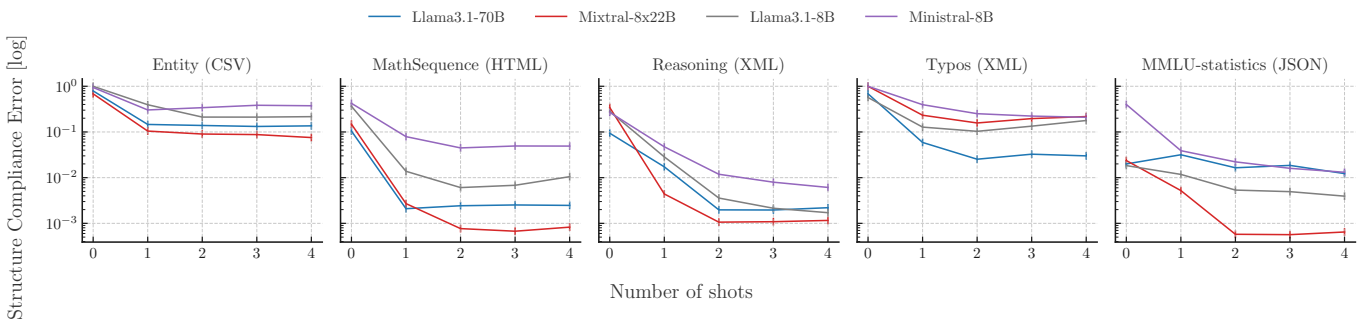


Fig. 3: Structure Compliance Error (SCE) on five representative combinations of datasets and structure, for a number of examples ranging between zero and four.

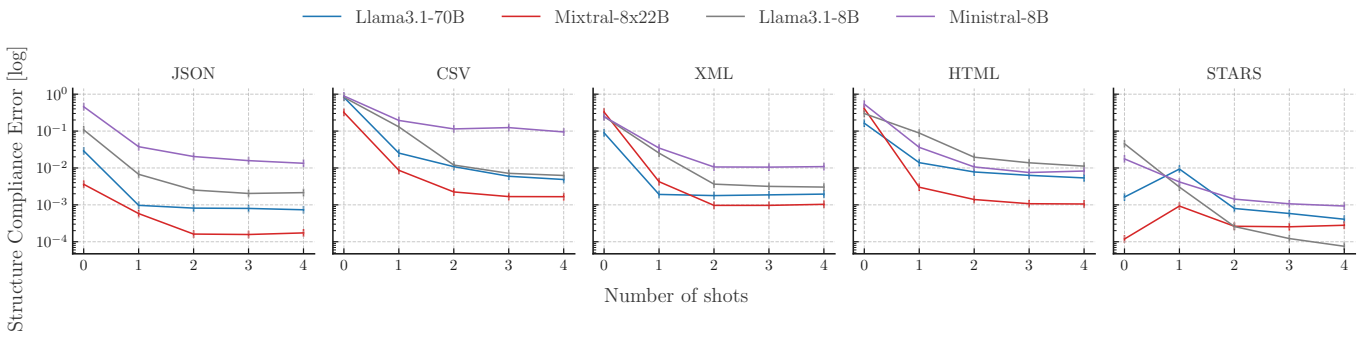


Fig. 4: Aggregated SCE for the different data structures (JSON, HTML, XML, CSV, STAR), for a number of examples ranging between zero and four.

a) Impact of Structure: Figure 4 reports the average SCE aggregated over each of the available data structures (JSON, HTML, XML, CSV, STAR). First, we observe that the different formatting structures pose various levels of difficulty to the models. For instance, generating CSV or HTML without examples ($N = 0$) leads to very high SCEs for all models, while the use of at least one in-context example leads to a reduction of SCE by an order of magnitude. On the other hand, simple structures such as STARS, where the models are asked to alternate between the response token and a delimitation asterisk, appear to be easily produced by LLMs, even without examples. Note that some models (e.g, Mixtral-8x22B), show an SCE for $N = 1$ larger than for $N = 0$. However, it is important to note that in those cases, the SCE remains low (<0.01). We hypothesize that such behavior stems from the fact that STARS is an arbitrary structure, which the models have never met during training. Regardless of the structure, the models' performance saturates around two few-shot examples.

b) Influence of models' size: Figure 5 reports the SCE of models of the same family (Qwen2.5) across the different model sizes (7B to 72B parameters) on the same datasets. Interestingly, a similar phenomenon can be observed at every model scale. Indeed, we also note that models generally achieve near-optimal performance when provided with two in-context examples, with smaller models generally saturating at higher structure compliance error rates.

c) Llama3.1 Case Study: Finally, Figure 2 reports the detailed structure compliance of Llama3.1-8B and Llama3.1-70B on all combinations of datasets and structure for $N = 0, 1$, and 2 examples. We observe that pure instruction-following with $N = 0$ (the zero-shot setting) yields unreliable results for both models. This can be observed across many base tasks and for most data structures, with structure compliance rates far below 1 in the majority of cases. For instance, both Llama3.1 models achieve particularly poor structure compliance for data formatted as CSV, e.g., Entity (8B: .0; 70B: .21), HTML, e.g., Reasoning (8B: .20, 70B: .70), and JSON, e.g., Sequence (8B: .61, 70B: .21).

In contrast, adding a single in-context example ($N = 1$) significantly improves results across all datasets. We observe that both models reach near maximum structure compliance performance on several datasets. Multiple exceptions remain, however, including for the Reasoning (CSV) dataset for both Llama3.1 models, as well as Entity (CSV), Reasoning (HTML), and MMLU-Pro (STARS) for the smaller Llama3.1-8B model.

Finally, we observe that providing a second in-context example ($N = 2$) is generally sufficient to saturate the performance of structured output generation. Indeed, for most base tasks and data formats, the models comply with the given data format almost perfectly at both parameter scales, with limited exceptions.

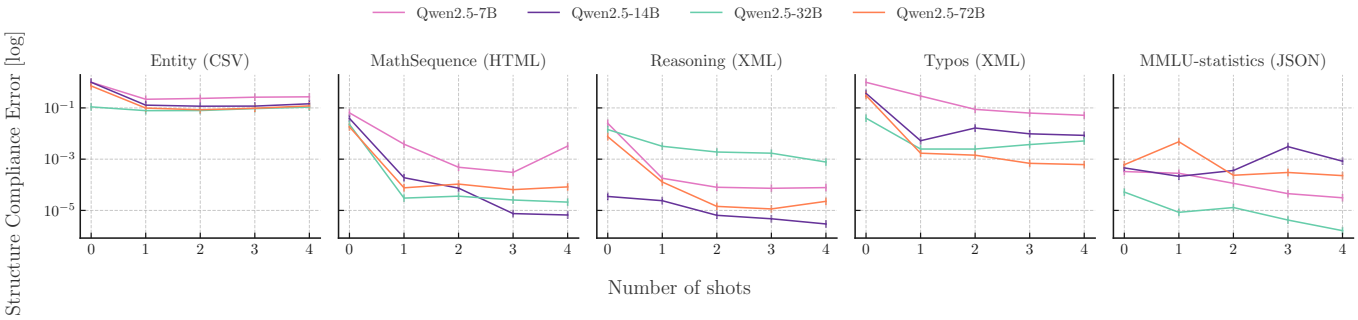


Fig. 5: SCE comparison across Qwen2.5 model sizes.

B. Finding 2: Additional shots may impair content

Table II reports the CQ of both Llama3.1-70B and Qwen2.5-72B with constrained generation for the reasoning datasets and for all four formats with $N = 0, 1, 2, 3, 4$ in-context examples. While the content of the answer is not directly linked to its structure, and the examples of few-shot prompting only contain information regarding the structure of the answer, we observe a significant influence of the number of examples on the content quality.

First, the CQ is significantly lower for $N = 0$ than for $N \geq 1$, for both models and all four formats. This phenomenon might be caused by the distortion effect. Indeed, both Llama3.1 and Qwen2.5 achieve high SCE with zero examples, as shown in Table IV. As the models are unlikely to spontaneously produce the expected structure, the intervention of the external verifier may significantly skew the model's next token distribution, resulting in subpar performance. The distortion effect is further investigated below.

Second, we observe that the CQ decreases for $N = 3$ and $N = 4$ compared to $N = 2$ for some data structures. As the SCE does not decrease for $N = 3, 4$, we hypothesize that this slight deterioration in content quality might be explained by the additional context provided to the model (see Section V).

TABLE II: Content Quality (CQ) of Llama3.1-70B and Qwen2.5-72B for the reasoning datasets with all four formats (JSON, CSV, HTML, XML) with up to four-shot examples. Models are instructed to generate Chain-of-Thought tokens.

Model	Format	Examples				
		$N=0$	$N=1$	$N=2$	$N=3$	$N=4$
Llama3.1	JSON	0.26	0.52	0.76	0.74	0.62
	CSV	0.04	0.44	0.68	0.54	0.60
	HTML	0.26	0.68	0.70	0.70	0.58
	XML	0.24	0.60	0.68	0.68	0.62
Qwen2.5	JSON	0.48	0.60	0.50	0.68	0.60
	CSV	0.06	0.50	0.50	0.44	0.50
	HTML	0.32	0.56	0.66	0.68	0.50
	XML	0.50	0.62	0.48	0.50	0.60

TABLE III: Content Quality scores (CQ) and Distortion impact of the format STARS for the MMLU-statistics dataset with $N = \{0, 1\}$ few-shot examples, for both Llama3.1 variants.

Model	Size	Structure	Examples	
			$N=0$	$N=1$
Llama3.1	70B	STARS	.780	.883
		NONE	.840	.876
Distortion			**.060**	**.007**
Llama3.1	8B	STARS	.520	.667
		NONE	.657	.664
Distortion			**.137**	**.003**

C. Finding 3: In context-learning can alleviate distortion

To further investigate the distortion impact of structures, we compare the CQ of the same models with a format constraint (STARS) and without (normal answer). Table III reports the CQ of models at two scales (of the Llama 3.1 family) for the $\text{MMLU}_{\text{statistics}}$ dataset and $N = 0, 1$. The results corroborate the previous observation: a significant decrease in performance can be observed for $N = 0$ when a structure is imposed ($-.137$ CQ for Llama3.1-8B and $-.06$ CQ for Llama3.1-70B). This difference is reduced when one example is provided ($N = 1$): the unconstrained CQ remains the same, while the constrained CQ increases. As a result, the difference is significantly lower (-0.003 CQ for Llama3.1-8B and -0.007 CQ for Llama3.1-70B). Since the provided example only contains information about the expected structure, we hypothesize that this phenomenon reflects the distortion effect.

D. Structure Compliance Error types

In Table IV we summarize the results of the largest models over all datasets. We show for each experimental configuration (dataset $\times$ model $\times$ number of in-context examples) the proportion of samples for which the model has an SCE $< .05$, i.e., a probability over 95% to generate the correct structure. As measured in this work, when presented with too few in-context examples, models often fail to comply with a given structure. In the following, we survey the most common Structure Compliance Error types observed across model families, scales, and output formats when no or too few

TABLE IV: Proportion with joint structure probability $> 95\%$ for the largest models. Zero to three-shot settings.

	Mistral-Large				**Mixtral-8x22B**				**Llama3.1-70B**				**Qwen2.5-72B**			
↓ **Dataset N-shots** →	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
Entity (CSV)	0.0	95.0	94.1	93.1	0.0	79.2	86.1	84.2	0.0	75.2	69.3	74.3	9.9	88.1	88.1	87.1
MMLU-Pro (JSON)	100	99.3	100	99.3	90.4	90.4	97.1	100	80.1	86.0	88.2	90.4	98.5	100	100	100
MMLU-Pro (stars)	100	92.6	99.3	99.3	91.9	87.5	96.3	98.5	61.8	33.1	69.9	83.8	100	97.8	99.3	100
$\text{MMLU}_{\text{Soc.}}$ (JSON)	100	100	100	100	95.5	100	100	100	91.0	99.0	98.5	99.5	100	100	100	100
$\text{MMLU}_{\text{Soc.}}$ (stars)	99.0	100	100	100	100	100	100	100	90.0	92.0	96.5	98.0	100	100	100	100
$\text{MMLU}_{\text{Stat.}}$ (JSON)	100	100	100	100	93.5	99.1	100	100	90.3	93.1	96.3	95.4	99.5	98.1	100	100
$\text{MMLU}_{\text{Stat.}}$ (stars)	99.1	98.1	99.5	99.5	100	96.3	99.5	100	88.9	64.4	92.1	94.9	100	93.5	99.5	99.5
$\text{MMLU}_{\text{Vir.}}$ (JSON)	98.2	100	100	100	85.5	98.8	99.4	99.4	69.3	74.7	73.5	74.7	96.4	97.6	99.4	99.4
$\text{MMLU}_{\text{Vir.}}$ (stars)	85.5	86.1	92.8	97.0	97.0	98.2	98.2	97.6	59.6	59.6	69.9	69.9	100	98.8	99.4	99.4
Sequence (CSV)	90.6	99.1	99.1	100	17.0	100	100	100	0.0	96.2	99.1	98.1	90.6	90.6	97.2	99.1
Sequence (HTML)	50.0	99.1	100	100	66.0	99.1	99.1	100	2.8	99.1	99.1	99.1	90.6	100	100	100
Sequence (JSON)	0.0	99.1	100	100	0.0	100	100	100	0.0	100	98.1	99.1	98.1	72.6	100	100
Sequence (XML)	92.5	99.1	100	100	62.3	100	100	100	100	100	100	99.1	98.1	99.1	100	100
Reasoning (CSV)	72.0	92.0	100	100	0.0	98.0	100	100	0.0	14.0	96.0	98.0	76.0	18.0	90.0	100
Reasoning (HTML)	2.0	74.0	100	100	0.0	100	100	100	0.0	82.0	100	100	66.0	100	100	100
Reasoning (JSON)	0.0	86.0	98.0	100	0.0	100	100	100	0.0	78.0	100	100	100	98.0	100	100
Reasoning (XML)	2.0	96.0	100	100	0.0	100	100	100	14.0	96.0	100	100	100	100	100	100
Typos (CSV)	100	100	100	100	5.0	95.0	98.0	97.0	0.0	96.0	91.0	92.0	100	100	100	100
Typos (HTML)	98.0	98.0	97.0	98.0	0.0	57.0	61.0	61.0	0.0	84.0	83.0	76.0	85.0	89.0	100	100
Typos (JSON)	0.0	93.0	98.0	98.0	91.0	70.0	62.0	58.0	55.0	96.0	94.0	90.0	100	100	100	100
Typos (XML)	0.0	0.0	70.0	97.0	0.0	6.0	38.0	35.0	0.0	86.0	94.0	91.0	0.0	99.0	100	100

examples are provided in-context. The fine-grained evaluation method proposed in this work lets us detect these token-level failures, which often go unnoticed when the evaluation relies strictly on binary parsing success. We identify two types of SCE: errors within syntactically valid outputs and strict schema violations. As can be observed from our empirical evaluations, both of these error types are addressed by using sufficient in-context examples.

Extraneous content and type/value mismatches When presented with no or a single example, models often insert additional content within their final response structure. For instance, instead of generating the requested JSON structure {`"person 1": ...`}, models frequently attempt to specify the person's name: {`"person 1 (Ash)": ...`}. Similar phenomena with superfluous information (e.g., a person's name, when only the person's ID was requested) occur in multiple choice answers (e.g., {`"answer": "E is the correct solution"`}), in CSV structures within otherwise compliant columns as well as with additional unrequested columns, and in HTML/XML structures with extra children tags or XML comments.

Furthermore, models may fail to comply with the expected formatting of the content. A typical situation is the generation of an unevaluated arithmetic expression, e.g., writing "10^0" instead of the expected value "1". When producing this type of SCE, the models generate structures that are syntactically valid and would thereby be missed by a purely parsing-based evaluation.

However, the subtle deviations from the expected formatting can cause a downstream parser in a broader automated system to fail. Furthermore, if instead of increasing the number of few-shot examples, a strict template-based constrained decoder were used, it would need to intervene in the sampling process at this stage, with the risk of distorting the model's predicted probability distribution.

Schema violations: These types of errors appear when models do not comply with the specified conventions of the target schema. A common case is omitting quotation marks enclosing string values in a JSON structure, especially when the value can also be interpreted as a number. This behavior is indicative of the impact of structured data in the training data: numerical values in JSON structure are often not enclosed within quotes, and models will have learned this pattern. When a requested structure deviates from a common template, the bias towards reproducing patterns seen during training must be overcome by providing sufficient in-context examples.

Furthermore, models sometimes attempt to end their response before the structure is complete. In the STARS structure, for example, models occasionally output fewer than five repetitions of the selected answer or omit the structure's closing symbols. This failure mode is particularly frequent when the models produce a particularly long Chain-of-Thought sequence to solve the base task.

V. Discussion

In this work, we studied the impact of few-shot prompting on constrained generation. Our experimental analysis shows that two shots are sufficient to obtain nearly optimal structure compliance across evaluated models and datasets. Indeed, when using only format-restricting instructions with a description of the output format (i.e., the zero-shot setting), we observe that models often do not produce the required data structures on many base tasks. Conversely, when provided with at least one example (i.e., an example pair of question and answer), the models' performance – their structure compliance error, or SCE – significantly improves. We believe the low scores without few-shot prompts are caused by the divergence of the constrained generation task from the usual natural language generation task for which LLMs are trained. Indeed, structured data represents a small share of the global training set, and the token probability distribution will hence be biased towards natural language. Prompting the model with well-structured examples in the context may guide the model toward the structured part of the embedding space. This interpretation may also explain why LLMs' SCE do not change significantly with more than two examples, as the embedding space may have already shifted to said structured part.

Additionally, we observed in our experiments that more than two examples may degrade Content Quality (CQ). We argue that these results are in line with previous works such as [31], where the authors observed that additional context may negatively impact the LLM's performance. Moreover, while examples contain information regarding the structure of the output, they are not relevant to the content of the answer. Therefore, it may be argued that these additional examples can have the same detrimental impact as irrelevant context on the quality of the answer – similarly to what was observed in [32]. Additionally, the lengthened context also increases the computational costs of the task. Consequently, we argue that two shots appear to achieve the optimal trade-off between context length and structure compliance.

Finally, we observed that the presence of examples in the prompt alleviates the distortion problem that may be induced by the constrained generation of some structured output. We hypothesize that this phenomenon may result from the near-optimal structure compliance achieved by the models with $N \geq 1$, which, in turn, minimizes the skewing of the next-token distribution. Conversely, this is not the case for $N = 0$, where SCE was substantially larger in our experiments, resulting in the significant distortion observed in our results.

Importantly, while the previous observations hold for many datasets, models, and data structures, these results may not apply to every existing LLM and structure. For instance, it is possible that LLMs fine-tuned to the production of structured output may not suffer from poor performance with $N = 0$, or from the distortion effect. However, fine-tuning an LLM comes with different trade-offs, such as additional training cost or unintended security issues [55], and as a consequence, few-shot prompting may be a better approach in some settings. It is also likely that particularly intricate structures may require more than two examples to achieve optimal SCE. Nevertheless, we argue that our results are general enough to warrant choosing $N = 2$ shots for structured output in most cases, and only adding additional examples when necessary.

References

[1] B. Zhang, B. Haddow, and A. Birch, "Prompting large language model for machine translation: A case study," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 41 092–41 110.

[2] S. Wang, X. Sun, X. Li, R. Ouyang, F. Wu, T. Zhang, J. Li, and G. Wang, "Gpt-ner: Named entity recognition via large language models," *arXiv preprint arXiv:2304.10428*, 2023.

[3] X. Sun, X. Li, J. Li, F. Wu, S. Guo, T. Zhang, and G. Wang, "Text classification via large language models," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 8990–9005.

[4] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *International Conference on Learning Representations*, 2021.

[5] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang *et al.*, "Mmlu-pro: A more robust and challenging multi-task language understanding benchmark," *arXiv preprint arXiv:2406.01574*, 2024.

[6] C. White, S. Dooley, M. Roberts, A. Pal, B. Feuer, S. Jain, R. Shwartz-Ziv, N. Jain, K. Saifullah, S. Naidu, C. Hegde, Y. LeCun, T. Goldstein, W. Neiswanger, and M. Goldblum, "Livebench: A challenging, contamination-free llm benchmark," 2024.

[7] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[8] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges." *33rd International Joint Conference on Artificial Intelligence (IJCAI 2024)*, 2024.

[9] J. C. M. Tan, P. Saroj, B. Runwal, H. Maheshwari, B. L. Y. Sheng, R. Cottrill, A. Chona, A. Kumar, and M. Motani, "Taskgen: A task-based, memory-infused agentic framework using strictjson," *arXiv preprint arXiv:2407.15734*, 2024.

[10] Q. Wang, T. Wang, Q. Li, J. Liang, and B. He, "Megaagent: A practical framework for autonomous cooperation in large-scale llm agent systems," *arXiv preprint arXiv:2408.09955*, 2024.

[11] S. Arora, B. Yang, S. Eyuboglu, A. Narayan, A. Hojel, I. Trummer, and C. Ré, "Language models enable simple systems for generating structured views of heterogeneous data lakes," *Proc. VLDB Endow.*, vol. 17, no. 2, p. 92–105, Oct. 2023.

[12] S. Shankar, H. Li, P. Asawa, M. Hulsebos, Y. Lin, J. Zamfirescu-Pereira, H. Chase, W. Fu-Hinthorn, A. G. Parameswaran, and E. Wu, "spade: Synthesizing data quality assertions for large language model pipelines," *Proceedings of the VLDB Endowment*, vol. 17, no. 12, pp. 4173–4186, 2024.

[13] M. Mondal, J. Audiffren, L. Dolamic, G. Bovet, and P. Cudré-Mauroux, "Cleaning semi-structured errors in open data using large language models," in *2024 11th IEEE Swiss Conference on Data Science (SDS)*. IEEE, 2024, pp. 258–261.

[14] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," *Advances in Neural Information Processing Systems*, vol. 36, pp. 68 539–68 551, 2023.

[15] Y. Qin, S. Hu, Y. Lin, W. Chen, N. Ding, G. Cui, Z. Zeng, X. Zhou, Y. Huang, C. Xiao *et al.*, "Tool learning with foundation models," *ACM Computing Surveys*, vol. 57, no. 4, pp. 1–40, 2024.

[16] C. Fourrier, N. Habib, J. Launay, and T. Wolf, "What's going on with the open llm leaderboard," *Hugging Face Blog (June 2023). URL: https://huggingface. co/blog/evaluatingmmlu-leaderboard*, 2023.

[17] J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou, "Instruction-Following Evaluation for Large Language Models," Nov. 2023, arXiv:2311.07911 [cs].

[18] Y. Qin, K. Song, Y. Hu, W. Yao, S. Cho, X. Wang, X. Wu, F. Liu, P. Liu, and D. Yu, "Infobench: Evaluating instruction following ability in large language models," *arXiv preprint arXiv:2401.03601*, 2024.

[19] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, "Self-instruct: Aligning language models with self-generated instructions," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 13 484–13 508.

[20] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu *et al.*, "Instruction tuning for large language models: A survey," *arXiv preprint arXiv:2308.10792*, 2023.

[21] R. Agarwal, A. Singh, L. M. Zhang, B. Bohnet, L. Rosias, S. C. Chan, B. Zhang, A. Faust, and H. Larochelle, "Many-shot in-context learning," in *ICML 2024 Workshop on In-Context Learning*, 2024.

[22] S. Zhao, T. Nguyen, and A. Grover, "Probing the decision boundaries of in-context learning in large language models," *arXiv preprint arXiv:2406.11233*, 2024.

[23] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, B. Chang, X. Sun, L. Li, and Z. Sui, "A Survey on In-context Learning," Jun. 2024, arXiv:2301.00234 [cs].

[24] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and A. Askell, "Language models are few-shot learners," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, pp. 1877–1901.

[25] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. Le, E. Chi, D. Zhou, and J. Wei, "Challenging BIG-bench tasks and whether chain-of-thought can solve them," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 13 003–13 051.

[26] B. T. Willard and R. Louf, "Efficient guided generation for llms," *arXiv preprint arXiv:2307.09702*, 2023.

[27] L. Beurer-Kellner, M. Fischer, and M. Vechev, "Guiding LLMs the right way: Fast, non-invasive constrained generation," in *Forty-first International Conference on Machine Learning*, 2024.

[28] S. Geng, M. Josifoski, M. Peyrard, and R. West, "Grammar-constrained decoding for structured NLP tasks without finetuning," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Dec. 2023, pp. 10 932–10 952.

[29] Z. R. Tam, C.-K. Wu, Y.-L. Tsai, C.-Y. Lin, H.-y. Lee, and Y.-N. Chen, "Let me speak freely? a study on the impact of format restrictions on performance of large language models," *arXiv preprint arXiv:2408.02442*, 2024.

[30] M. Abdin, J. Aneja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, H. Behl *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024.

[31] M. Levy, A. Jacoby, and Y. Goldberg, "Same Task, More Tokens: the Impact of Input Length on the Reasoning Performance of Large Language Models," 2024, publisher: arXiv Version Number: 2.

[32] F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. H. Chi, N. Schärli, and D. Zhou, "Large language models can be easily distracted by irrelevant context," in *International Conference on Machine Learning*. PMLR, 2023, pp. 31 210–31 227.

[33] O. Yoran, T. Wolfson, O. Ram, and J. Berant, "Making retrieval-augmented language models robust to irrelevant context," in *The Twelfth International Conference on Learning Representations*, 2024.

[34] F. D. Keles, P. M. Wijewardena, and C. Hegde, "On the computational complexity of self-attention," in *International conference on algorithmic learning theory*. PMLR, 2023, pp. 597–619.

[35] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–28, 2022.

[36] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19 274–19 286.

[37] P. Wang, L. Li, L. Chen, Z. Cai, D. Zhu, B. Lin, Y. Cao, Q. Liu, T. Liu, and Z. Sui, "Large language models are not fair evaluators," *arXiv preprint arXiv:2305.17926*, 2023.

[38] C. Shen, L. Cheng, X.-P. Nguyen, Y. You, and L. Bing, "Large language models are not yet human-level evaluators for abstractive summarization," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 4215–4233.

[39] J. Ye, Y. Wang, Y. Huang, D. Chen, Q. Zhang, N. Moniz, T. Gao, W. Geyer, C. Huang, P.-Y. Chen *et al.*, "Justice or prejudice? quantifying biases in llm-as-a-judge," *arXiv preprint arXiv:2410.02736*, 2024.

[40] C. Xia, C. Xing, J. Du, X. Yang, Y. Feng, R. Xu, W. Yin, and C. Xiong, "Fofo: A benchmark to evaluate llms' format-following capability," *arXiv preprint arXiv:2402.18667*, 2024.

[41] B. Wen, P. Ke, X. Gu, L. Wu, H. Huang, J. Zhou, W. Li, B. Hu, W. Gao, J. Xu, Y. Liu, J. Tang, H. Wang, and M. Huang, "Benchmarking Complex Instruction-Following with Multiple Constraints Composition," Jul. 2024, arXiv:2407.03978 [cs].

[42] H. J. Kim, H. Cho, J. Kim, T. Kim, K. M. Yoo, and S.-g. Lee, "Self-generated in-context learning: Leveraging auto-regressive language models as a demonstration generator," *arXiv preprint arXiv:2206.08082*, 2022.

[43] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, and Y. Shoham, "In-context retrieval-augmented language models," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1316–1331, 2023.

[44] N. De Cao, W. Aziz, and I. Titov, "Editing factual knowledge in language models," in *EMNLP 2021-2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2021, pp. 6491–6506.

[45] K. Wu, E. Wu, and J. Y. Zou, "Clasheval: Quantifying the tug-of-war between an llm's internal prior and external evidence," *Advances in Neural Information Processing Systems*, vol. 37, pp. 33 402–33 422, 2024.

[46] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[47] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[48] J. Hu and R. Levy, "Prompting is not a substitute for probability measurements in large language models," *arXiv preprint arXiv:2305.13264*, 2023.

[49] M. Mondal, L. Dolamic, G. Bovet, P. Cudré-Mauroux, and J. Audiffren, "Do large language models exhibit cognitive dissonance? studying the difference between revealed beliefs and stated answers," *arXiv preprint arXiv:2406.14986*, 2024.

[50] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 2003, pp. 142–147.

[51] J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou, "Instruction-following evaluation for large language models," *arXiv preprint arXiv:2311.07911*, 2023.

[52] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[53] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, and Z. Fan, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.

[54] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[55] X. Qi, Y. Zeng, T. Xie, P.-Y. Chen, R. Jia, P. Mittal, and P. Henderson, "Fine-tuning aligned language models compromises safety, even when users do not intend to!" *arXiv preprint arXiv:2310.03693*, 2023.