

How to get the model to do what we want:

Structured outputs and their applications

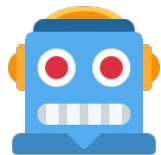
Zdeněk Kasner



Deploying LLMs be like...



Generate a weather forecast from the data. **Output only.** No talking around, no formatting!



Sure! Here is a required output with no extra details:
The weather in Prague this morning will be cold...

What can we do about it?

1. Be **extremely** precise *(mostly works for the state-of-the-art models)*

You will output a single-paragraph response. The output will be delimited by triple backticks ``` from the top and from the bottom like this: {example}. Do not output anything else: once you generate the final set of backticks, end your response.

2. Prime the LLM by **prefixing its answer**

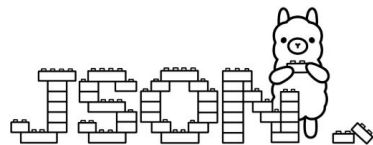
```
{ 'user' : prompt with instructions
```

```
  'assistant': 'Sure, here is my response: \n “ ” }
```

(note the opening quote)

What can we do about it?

3. Use **structured outputs**



JSON mode

basic, supported by most
current frameworks / APIs



GBNF grammar

advanced, needs to use a
specific library

Defining a **JSON scheme**: fields and their types.

- See the [Pydantic](#) library.

Generally supported: [ollama](#), [OpenAI](#), [Anthropic](#), etc.

- Careful: currently only limited support for reasoning models.

```
from ollama import chat
from pydantic import BaseModel

class Country(BaseModel):
    name: str
    capital: str
    languages: list[str]

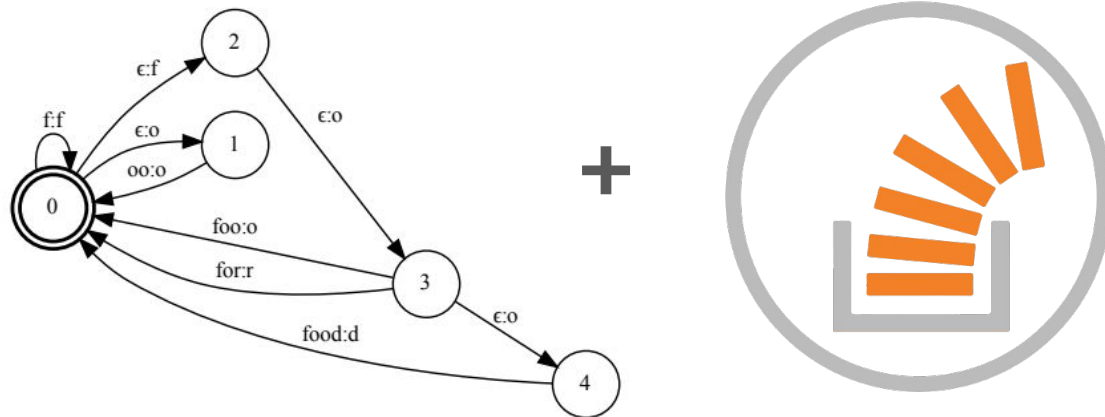
response = chat(
    messages=[ { 'role': 'user', 'content': 'Tell me
about Canada.' } ],
    model='llama3.1',
    format=Country.model_json_schema(),
)

country =
Country.model_validate_json(response.message.content)
print(country)
```

GBNF grammar

LLM output needs to follow a specific grammar → wider range of use-cases.

Implemented in the [outlines library](#).



“GBNF, a rebranding of Backus-Naur Form is a kind of Regex if you somehow made Regex more obtuse and clunky and also way less powerful. (...) HOWEVER, if you tame this vile beast of a language you can make AI respond to you in pretty much any way you like.”

[Reddit user in r/LocalLLaMa](#)

Issue #1: Structured output can be unnatural

For some tasks, structured output is an unnatural format → can lead to degraded performance ([Tam et al., 2024](#)).

Standard Prompting

Instruct : Provide your output in the following text format:

Step by step reasoning: ...

Answer: The final answer is ...

.....
Step by step reasoning:

1. Eliza's regular rate per hour is \$10 for the first 40 hours.
2. For the 5 hours of overtime, her rate is 1.2 times her regular rate, which is $1.2 * \$10 = \12 per hour.
3. Eliza worked 40 regular hours and 5 overtime hours, totaling 45 hours.
4. Earnings for regular hours = 40 hours * \$10/hour = \$400
5. Earnings for overtime hours = 5 hours * \$12/hour = \$60
6. Total earnings for the week = \$400 (regular) + \$60 (overtime) = \$460

Answer: The final answer is 460.



Format-Restricting Prompting

Instruct : Provide your output in the following valid JSON

format:

```
```json
```

```
{
 "step_by_step_reasoning": ...,
 "answer": ...
}
```

```
```
```

.....

```
```json
```

```
{
```

```
 "step_by_step_reasoning": "Calculate the earnings for \\
the first 40 hours at $10 per hour. Then calculate the \\
earnings for the additional 5 hours at $10 * 1.2 per \\
hour. Add both amounts to find the total earnings for \\
the week.",
 "answer": 490
```

```
}
```

```
```
```



Issue #2: Token-terminal mismatch

LLM's token boundaries may not match the terminals in the grammar.

S → FUNCTION "(" NUMBER ")"
FUNCTION → "foo" | "bar"
NUMBER → "123" | "456"

| Score | Tokenization |
|-------|--------------|
| 0.4 | fo o(1 2 3) |
| 0.3 | bar (456) |
| 0.2 | foo (123) |
| 0.1 | ba r(4 5 6) |

Naive approach: allow only tokens that match the terminals
→ can lead to suboptimal performance

Better approaches: subgrammars ([Beurer-Kellner et al., 2024](#)), FSA composition ([Koo et al., 2024](#))

How we used structured outputs in our experiments

Our use-case: producing a structured list of text annotations.

gemma2 gold-final-zeroshot-claude-3-7-sonnet-ann0

The Ituano team won 4-3 against Vila Nova in a match played on November 17th, 2024. Ituano scored four goals throughout the game, with Leozinho and Salatiel contributing to their victory. Vila Nova fought back with three goals from Luciano Naninho, Alesson, and Jemmes, but ultimately fell short.

```
[{
    "reason": "The match date is incorrect. The data shows the match was on November 17th, 2024, but the actual date format in the data is '2024-11-17', which is November 17th, 2024.",
    "text": "November 17th, 2024",
    "type": 0
},
{
    "reason": "The text mentions Leozinho scoring for Ituano without mentioning José Aldo and Vinícius Paiva who also scored according to the events data.",
    "text": "Leozinho and Salatiel contributing to their victory",
    "type": 2
}]
```

Other examples of applications

- **Multiple-choice question answering.**

```
generate.choice(model, ["Pizza", "Pasta", "Salad"])
```

- **Calling tools with specific interface.**

```
click(x={number}, y={number}); fill({text})
```

- **Parsing unstructured documents.**

```
pets=[ Pet(name='Luna', animal='cat', age=5, color='grey',  
favorite_toy='yarn'), Pet(name='Loki', animal='cat', age=2,  
color='black', favorite_toy='tennis balls')]
```

Resources

- [outlines library](#)
- JSON mode: [ollama](#), [OpenAI](#), [Google](#), [Anthropic](#)
- papers
 - [Efficient Guided Generation for Large Language Models \(Willard and Louf, 2023\)](#)
 - [Automata-based constraints for language model decoding \(Koo et al., 2024\)](#)
 - [Let Me Speak Freely? A Study on the Impact of Format Restrictions on Performance of Large Language Models \(Tam et al., 2024\)](#)
 - [Guiding LLMs The Right Way: Fast, Non-Invasive Constrained Generation \(Beurer-Kellner et al., 2024\)](#)
- [video: Structured Output from LLMs: Grammars, Regex, and State Machines](#)