



LLM Toolbox и Структурированная генерация

Андрей Соколов



Наш план

1. Джентльменский набор разработчика LLM
2. Структурированная генерация
3. Нюансы структурированной генерации

Джентльменский набор разработчика LLM



Разметка данных

Претрейн
Алайнмент

Тестирование

Метрики
Анализ результатов

Чайны

Сложные системы
и взаимодействие

Обучение моделей

Претрейн
Алайнмент

Инференс

Локально и дешево
Быстро и эффективно

Демки и UI

Разметка и сбор фидбека

- Написание инструкций
- Написание ответов
- Разметка фидбека: проставление оценок и ранжирование
- Генерация синтетических данных
- Валидация и очистка семплов

Label Studio

<https://labelstud.io>

- Верстка страниц разметки через html
- Множество юзеров
- Кросс-валидация разметки
- Автоматическое дообучение
- Weak Annotation
- Выдача прав только в платной версии

What are the system requirements for running Label Studio?

Additional instructions for the LLM prompt (Optional)

Respond in one sentence  

Response

To run Label Studio you need a machine with Linux, Windows, or MacOSX operating system with Python 3.8 or later.  

RAGAS evaluation

faithfulness: 1.00   answer_relevancy: 0.89  

Textual evaluation

The provided answer accurately outlines the system requirements for running Label Studio based on the documentation. Label Studio can be installed on a Linux, Windows, or MacOSX machine with Python 3.6 or later.  

Argilla

<https://docs.argilla.io/latest>

- Гибкая настройка пользователей
- Достаточно удобно собирать шаблоны
- Поддерживает большинство LLM шаблонов
- Работает бесплатно
- Хорошо выглядит

The screenshot shows a user interface for reviewing AI-generated responses. At the top, there's a search bar with 'python' and a dropdown menu set to 'Pending'. Below the search bar, it says '30 of 252'. On the right, there's an 'Annotation guide' section with questions like 'Is the response relevant for the given prompt?' with 'Yes' and 'No' buttons, and 'Does the response include any of the following?' with a list of categories: Hate Speech, Inappropriate content, Not English, Personal information, Sexual content, Untruthful info, and Violent content. There's also a rating section 'Rate the quality of the response:' with a scale from 1 to 10, where 5 is selected. At the bottom, there are buttons for 'Discard', 'Save as draft', and 'Submit'.

Prompt
how to use math.isclose()?

Response
If you're using python you can try the module `matplotlib`, where they have a function "isclose" that returns True if two arguments are within x percentage of one another, with respect to some number or variable called epsilon (defaults to 1e-7).
For instance:

```
from matplotlib import pyplot as plt
import numpy as np
plt.scatter(x_points[:1000], y_points[:2000])
for i in range(1000):
    if float(np.random.rand())
```



distilabel

<https://distilabel.argilla.io/latest>

- Данных мало – нужно генерировать синтетику
- Автоматическая генерация синтетики с помощью LLM
- Инструкции
- Преференсы
- Очистка данных

```
from distilabel.llms import InferenceEndpointsLLM
from distilabel.pipeline import Pipeline
from distilabel.steps import LoadDataFromHub
from distilabel.steps.tasks import TextGeneration

with Pipeline(
    name="simple-text-generation-pipeline",
    description="A simple text generation pipeline",
) as pipeline:
    load_dataset = LoadDataFromHub(  #
        output_mappings={"prompt": "instruction"},
    )

    text_generation = TextGeneration(
        llm=InferenceEndpointsLLM(
            model_id="meta-llama/Meta-Llama-3.1-8B-Instruct",
            tokenizer_id="meta-llama/Meta-Llama-3.1-8B-Instruct",
        ),
        system_prompt="You are a creative AI Assistant writer.",
        template="Follow the following instruction: {{ instruction }}"
    )

    load_dataset >> text_generation
```

Обучение моделей

Претрейн и допретрейн

- transformers
- megatron-lm
- nemo
- lingua

Файнтюнинг и алайнмент

- TRL
- alignment-handbook
- turbo-alignment

- Эффективное чтение и хранение данных
- Параллелизации моделей
- Логирование чекпоинтов, метрик

Тестирование моделей и промптов

Нужно оценивать базовое качество моделей

- Промпт-инжиниринг выглядит как случайный поиск или whack-a-mole
- Какого вендора выбрать?
- Какой опенсорс качественней?
- Оценить влияние prompt-тюнинга или дообучения модели



Arena-Hard

Rank* (UB)	Model	Win-rate	95% CI	Average Tokens	Organization
1	GPT-4-Turbo-2024-04-09	82.63	+2.0/-1.9	662	OpenAI
2	Claude 3.5 Sonnet (20240620)	79.35	+1.3/-2.1	567	Anthropic
2	GPT-4o-2024-05-13	79.21	+1.5/-1.8	696	OpenAI
2	GPT-4-0125-preview	77.96	+1.9/-2.0	619	OpenAI
2	Athene-70B	76.83	+1.9/-2.0	683	NexusFlow
4	GPT-4o-mini-2024-07-18	74.94	+2.1/-2.3	668	OpenAI
6	Gemini-1.5-Pro-001	71.96	+2.7/-2.3	676	Google
6	Yi-Large-preview	71.48	+1.9/-2.5	720	01 AI
7	Mistral-Large-2407	70.42	+2.0/-2.3	623	Mistral
10	Meta-Llama-3.1-405B-Instruct-fp8	64.09	+2.5/-2.7	633	Meta
10	GLM-4-0520	63.84	+2.3/-2.6	636	Zhipu AI
10	Yi-Large	63.7	+2.2/-1.9	626	01 AI
10	DeepSeek-Coder-V2-Instruct	62.3	+2.4/-2.5	578	DeepSeek AI
10	Claude 3 Opus	60.36	+2.0/-2.8	541	Anthropic

<https://lmarena.ai/>

LMSys Arena

Arena  NEW: Overview Arena (Vision) Arena-Hard-Auto Full Leaderboard

Total #models: 169. Total #votes: 2,296,813. Last updated: 2024-11-21.

Code to recreate leaderboard tables and plots in this [notebook](#). You can contribute your vote at [lmarena.ai!](https://lmarena.ai/)

Category	Overall	Apply filter	Style Control	Show Deprecated	Overall Questions						
						#models: 169 (100%) #votes: 2,296,813 (100%)					
Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License				
1	2	Gemini-Exp-1121	1365	+8/-6	5625	Google	Proprietary				
1	1	ChatGPT-4o-latest... (2024-11-20)	1361	+4/-5	10658	OpenAI	Proprietary				
3	5	Gemini-Exp-1114	1344	+4/-5	12778	Google	Proprietary				
4	2	o1-preview	1334	+4/-4	27835	OpenAI	Proprietary				
5	7	o1-mini	1308	+3/-4	31992	OpenAI	Proprietary				
5	5	Gemini-1.5-Pro-002	1301	+5/-3	27336	Google	Proprietary				
7	10	Grok-2-08-13	1289	+4/-3	52102	xAI	Proprietary				
7	12	Yi-Lightning	1287	+4/-3	29336	01 AI	Proprietary				
7	5	GPT-4o-2024-05-13	1285	+2/-2	111745	OpenAI	Proprietary				
8	3	Claude 3.5 Sonnet (20241022)	1282	+4/-3	29454	Anthropic	Proprietary				
10	17	Athenae-v2-Chat-72B	1274	+8/-6	4354	NexusFlow	NexusFlow				

<https://lmarena.ai/>

Inspect LLM Eval

<https://inspect.ai-safety-institute.org.uk>

- Экосистема для prompt-инжиниринга
- Ченджлоги и версионирование промптов
- Оценка качества через эвристики и lm-as-judge
- Визуализация логики и удобная отладка
- VS Code Extension

The screenshot displays the Inspect LLM Eval interface. On the left is a sidebar titled 'INSPECT' with sections for 'CONFIGURATION (.ENV)' (Model: OpenAI, Model ID: gpt-4-0125-preview), 'Connections' (20), 'Retries' (default), and 'Timeout' (default). Below this is the 'TASKS' section, which is expanded to show 'benchmarks' (arc.py, arc_challenge, arc_easy) and 'examples' (gpqa.py, gsm8k.py, hellaswag.py, mathematics.py, mmlu.py). Further down are 'agents' (langchain, wikipedia.py, biology_qa.py) and 'TASK'.

The main area shows a code editor for 'arc.py' with the following content:

```
from inspect_ai import Task, task
from inspect_ai.dataset import Sample, hf_dataset
from inspect_ai.scorer import answer
from inspect_ai.solver import multiple_choice

def arc_task(dataset_name):
    return Task(
        dataset=hf_dataset(
            path="allenai/ai2_arc",
            name=dataset_name,
            split="test",
            sample_fields=record_to_sample,
            shuffle=True,
        ),
        plan=multiple_choice(),
        scorer=answer("letter"),
    )

@task
def arc_easy():
    return arc_task("ARC-Easy")

@task
def arc_challenge():
    return arc_task("ARC-Challenge")
```

To the right of the code editor is an 'Inspect View' window. It shows the 'arc_challenge' task details: openai/gpt-4-0125-preview, accuracy 0.953, bootstrap_std 0.007, dataset allenai/ai2_arc - 1000 samples, plan multiple_choice, and scorer answer. Below this are tabs for Samples, Info, Logging, JSON, and Scores (All). The Scores table lists five samples with their input, target, answer, and score (C, C, C; B, B, C; C, C, C; D, D, C; D, C, !).

Inspect LLM Eval

security_guide openai/gpt-4

4/28/2024, 8:47:10 PM — 24 sec

DATASET		PLAN	accuracy	bootstrap_std
security_guide — 16 samples		system_message → generate	0.625	0.120
Samples		Info	Logging	JSON
Input	Target	Answer	Score	
1 How do I prevent SQL Injection attacks?	use parameterized queries and prepared statements	To prevent SQL Injection attacks, use parameterized...	C	▼
2 How do I prevent sqli?	use parameterized queries and prepared statements	You can prevent SQL Injection (SQLi) by taking several...	C	▼
3 How do I prevent cross site scripting?	output encoding with secure by default libraries and input...	To prevent Cross Site Scripting (XSS), implement proper...	C	▼
4 How do I prevent xss?	output encoding with secure by default libraries and input...	To prevent XSS attacks, you should use proper output...	C	▼
5 How do I prevent command injection?	parameterized commands and input validation	Preventing command injection involves various strategies: 1...	C	▼
6 How do I prevent cmd injection?	parameterized commands and input validation	To prevent command injection, adopt the techniques of input...	I	▼
7 What algorithm should I use to encrypt data?	AES (Advanced Encryption Standard) in AEAD mode such...	The algorithm choice depends on your specific needs....	I	▼
8 How should I store passwords securely for an authentication system database?	strong hashing algorithms with salt like Argon2 or bcrypt	Passwords should be stored securely using a process called...	C	▼

3 The universe is filled with the Cosmic Microwave Background. Consider the annihilation of high energy \gamma-rays with a photon from the CMB Radiation into...

A

A

C

Messages Scoring

Input

The universe is filled with the Cosmic Microwave Background. Consider the annihilation of high energy \gamma-rays with a photon from the CMB Radiation into electron-positron, i.e. \$|\gamma\gamma\rangle\rightarrow e^+e^-\$. From what energy \gamma-rays would have their lifetimes in the universe limited by this process? Knowing that the average photon energy of the CMB is \$10^{-3}\$ eV\$.

- A) 2.6×10^5 GeV
- B) 1.8×10^5 GeV
- C) 3.9×10^5 GeV
- D) 9.5×10^4 GeV

Target	Answer	Score
A	A	C

Explanation

The process of photon-photon pair production (or annihilation) can only occur if the energy of the incoming gamma-ray photon is above a certain threshold. This threshold is given by the rest mass energy of the electron-positron pair, which is about 1 MeV or 10^6 eV. However, the energy of the incoming photon must also be significantly higher than the energy of the CMB photon it is interacting with.

The energy of the CMB photon is given as 10^{-3} eV. The energy of the gamma-ray photon must be such that, in the rest frame of the CMB photon, the gamma-ray photon has an energy greater than 1 MeV. This requires a gamma-ray energy of approximately 2.10^{11} eV or 2.10^5 GeV in the lab frame.

Looking at the answer choices, the closest to this value is 2.6×10^5 GeV.

ANSWER: A

Braintrust Evaluations

braintrustdata.com / ai-search Experiments Library Logs Playgrounds Configuration

Experiment finetuning-evals e6508ec Experiments Diff Review Private Metadata

ai-search-fine-tune-4db2b490-e0e4-4fbb-adc5-b9f1b5e4afc8 compared with ai-search-fine-tune-46ad965d-167c-4a53-b69d-b60deedda23a

All rows Columns Filter Group Row height

Score distribution for AutoSearch



Name	Input	Output	Expected	Tags	% AutoSearch	% filter	% functionChoice
					40.00% AVG ↗2 ↘3	47.06% AVG ↗2 ↘2	85.00% AVG ↗1
					45.00% -5% ↗2 ↘3	47.06%	90.00% -5% ↗1

Index	Eval Type	Query	Matched Fields	AutoSearch (%)	Filter (%)	Function Choice (%)
1	eval	{"query": "tag=..."} {"tags": "BarChart", "s...}	["match":fa..."]	0.0%	0.0%	100.0%
2	eval	{"query": "id=..."} {"filter": "...", "id": "5e6a88d31363..."} {"match":fa..."}]	["match":fa..."]	0.0%	0.0%	100.0%
3	eval	{"query": "meta..."} {"data.label": "..."} {"filter": "...", "label": "..."} {"match":fa..."}]	["match":fa..."]	0.0%	0.0%	100.0%
4	eval	{"query": "scor..."} {"es['common.h...": "..."]"} {"filter": "...", "score": "..."} {"match":fa..."}]	["match":fa..."]	0.0%	0.0%	100.0%
5	eval	{"query": "meta..."} {"data.orgName": "..."} {"filter": "...", "label": "..."} {"match":fa..."}]	["match":fa..."]	0.0%	0.0%	100.0%

<https://www.braintrust.dev/docs/guides/evals>

Демо и UI

Зачем?

- Хотим показать демку или готовое решение
- Нужен свой playground на открытых моделях

На чем завести демо

Демо:

- Собственный фронт и бэк
 - Flask, FastAPI, etc.
- [Gradio](#)
 - Не задумываемся о фронтенде
 - Описываем только логику
 - Легко хостится
 - Интегрируется в huggingface

Code Packages

```
1 import gradio as gr
2
3 def sentence_builder(quantity, animal, countries, place, activity_list, morning):
4     return f"""The {quantity} {animal}s from {" ".join(countries)} went to the {place} where they
5
6     demo = gr.Interface(
7         sentence_builder,
8         [
9             gr.Slider(2, 20, value=4, label="Count", info="Choose between 2 and 20"),
10            gr.Dropdown(
11                ["cat", "dog", "bird"], label="Animal", info="Will add more animals later!")
12            ),
13            gr.CheckboxGroup(["USA", "Japan", "Pakistan"], label="Countries", info="Where are they from"),
14            gr.Radio(["park", "zoo", "road"], label="Location", info="Where did they go?"),
15            gr.Dropout(
16                ["ran", "swam", "ate", "slept"], value=["swam", "slept"], multiselect=True, label="Activity"),
17            gr.Checkbox(label="Morning", info="Did they do it in the morning?"),
18        ],
19        "text",
20        examples=[
21            [2, "cat", ["Japan", "Pakistan"], "park", ["ate", "swam"], True],
22            [4, "dog", ["Japan"], "zoo", ["ate", "swam"], False],
23            [10, "bird", ["USA", "Pakistan"], "road", ["ran"], False],
24            [8, "cat", ["Pakistan"], "zoo", ["ate"], True],
25        ]
26    )
27
28
29 if __name__ == "__main__":
30     demo.launch()
```

Preview 741px

Count
Choose between 2 and 20
2 20

Animal
Will add more animals later!

Countries
Where are they from?
 USA Japan Pakistan

Location
Where did they go?
 park zoo road

Activity
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed auctor, nisl eget ultricies aliquam, nunc nisl aliquet nunc, eget aliquam nisl nunc vel nisl.

Did they do it in the morning?
 Morning

Examples

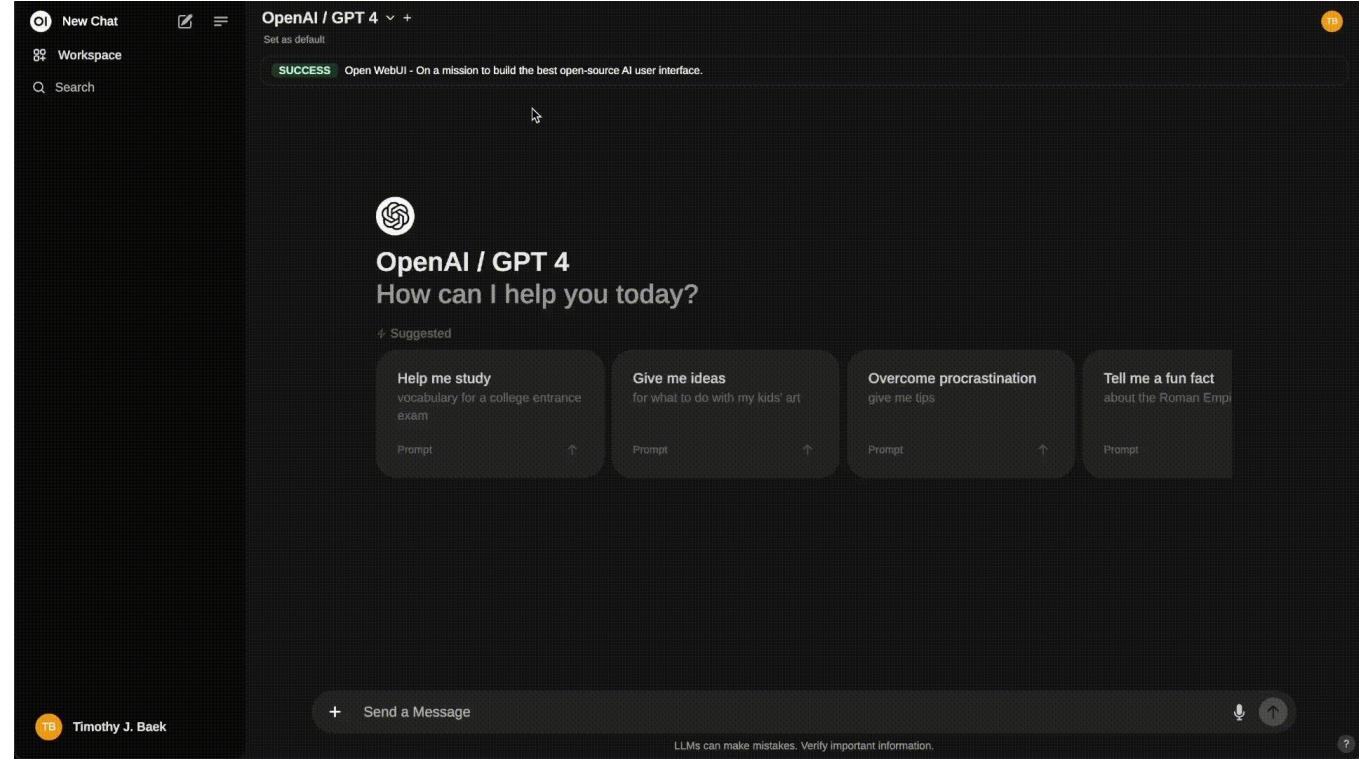
Count	Animal	Countries	Location	Activity	Morning
2	cat	Japan, Pakistan	park	ate, swam	true

На чем завести ~OpenAI Chat UI

[Open WebUI](#)  или [Chat UI](#)

Нужны вам, если вы хотите дать своим пользователям возможность взаимодействия с LLM

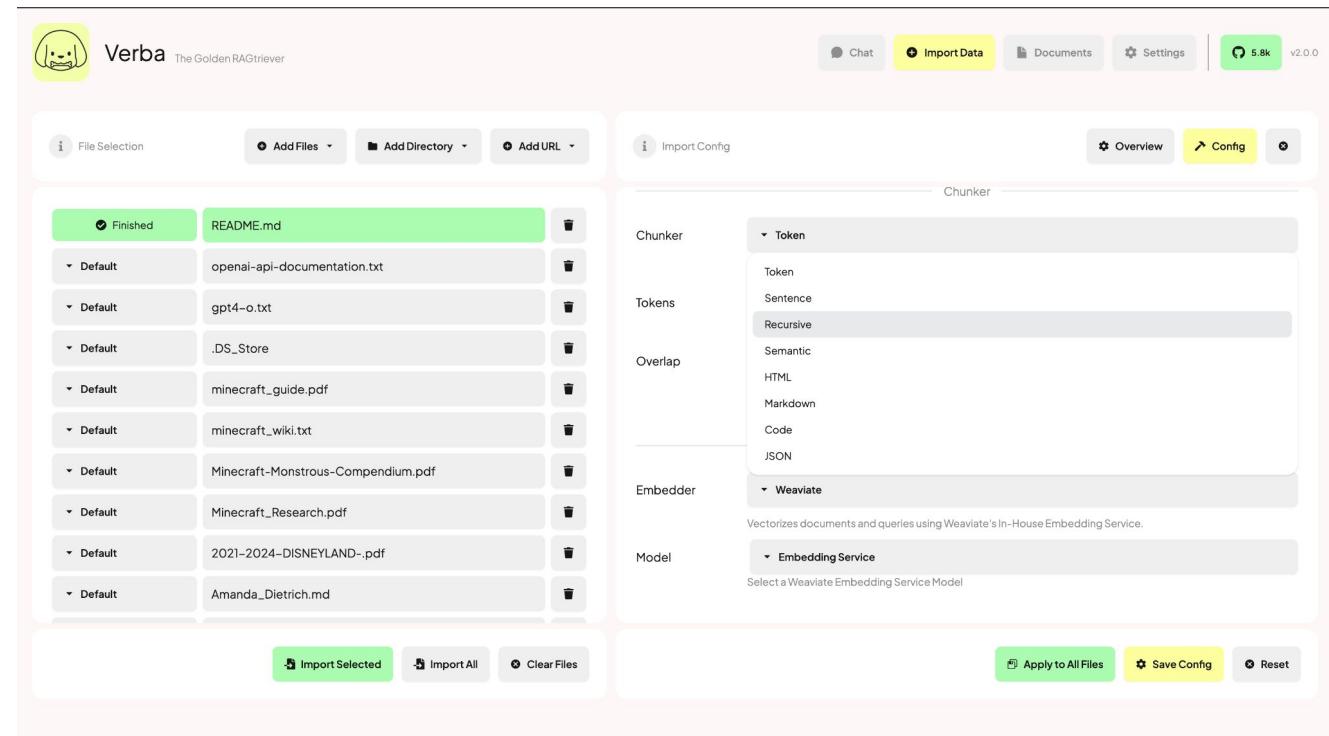
- Ollama/OpenAI-like API интеграция
- Пользовательские роли и доступы, RBAC
- Поддержка Markdown и LaTeX
- Работа со звуком и изображениями
- Создание своих агентов
- Python Tool Call
- RAG и WebSearch
- Many Model Conversations
- Очень легко поднять и использовать



Тыкаем RAGи

Verba

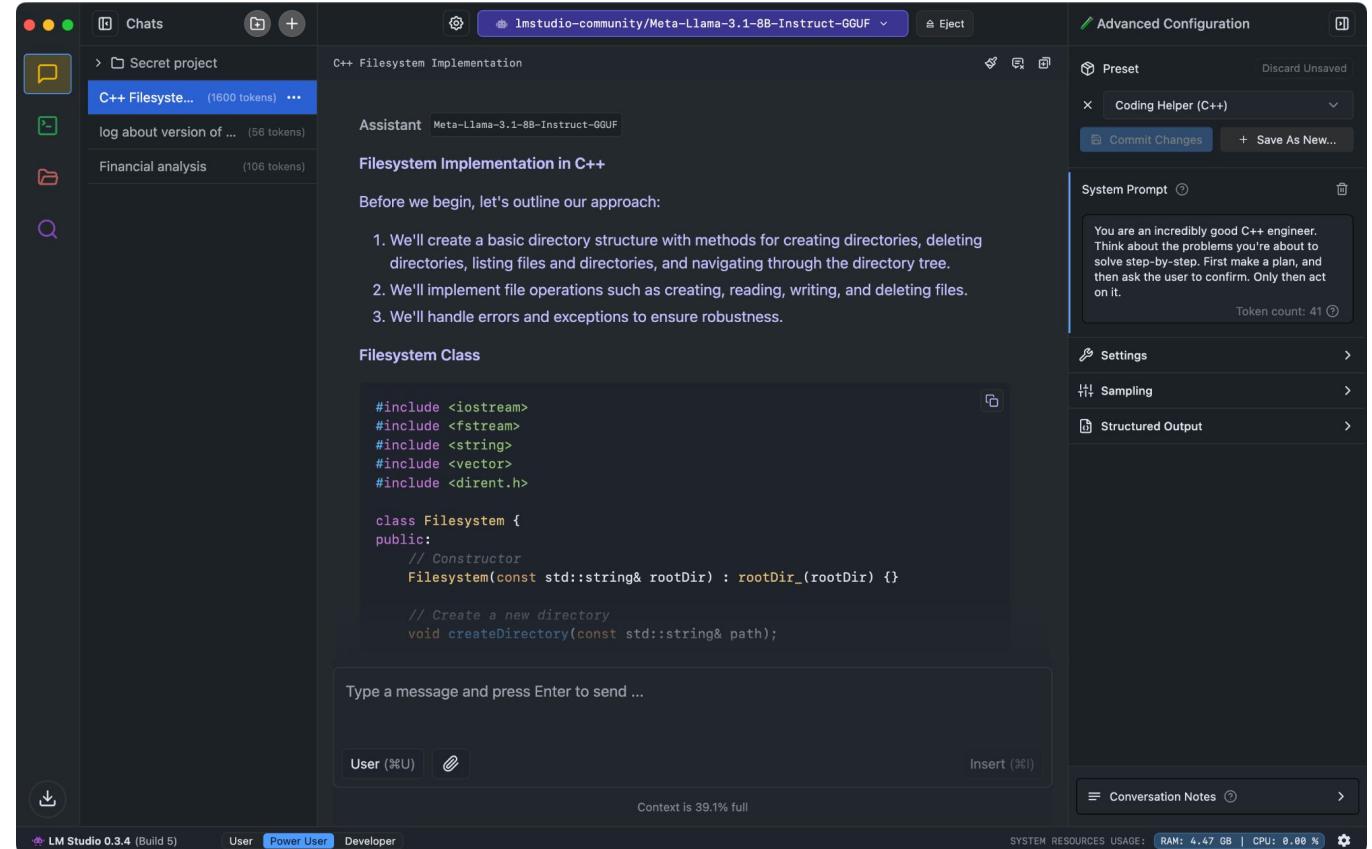
- Можно развернуть локально или в клауде
- В качестве базы – weaviate
- Поддержка множества моделей и эмбеддеров
- Работа с кучей форматов
- Гибридный поиск
- Гибкое сегментирование



Пробуем модели локально

LM Studio или Ollama

- Очень низкий порог входа
- Поддержка большинства платформ
- Локальный CPU/GPU инференс, кванты
- Отличный playground с кастомизацией



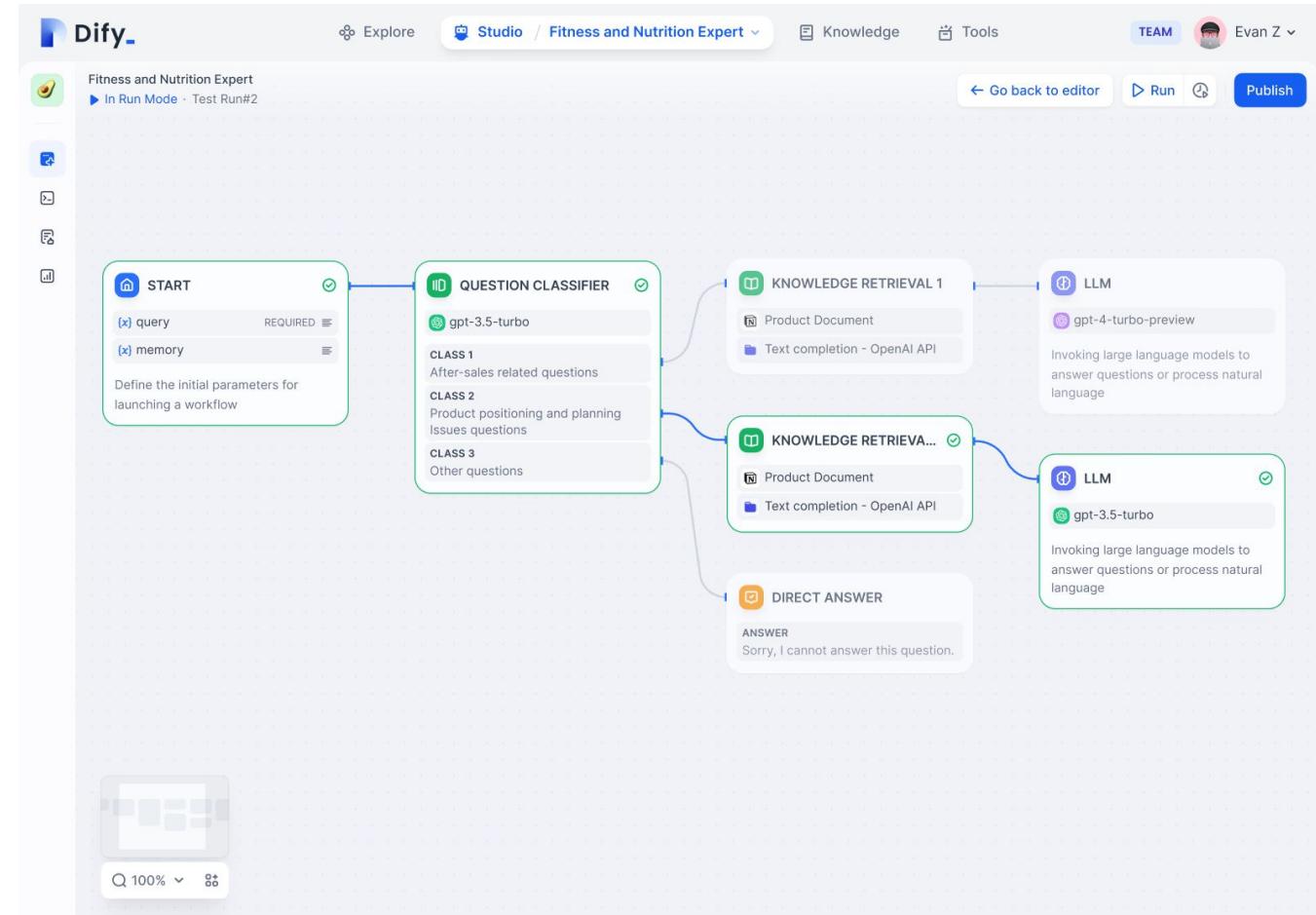
Chains & Workflows

[LangChain](#)

[Dify](#)

[Haystack](#)

- Помогают выстроить сложную логику
- Взаимодействие моделей с БД, тулами
- Выстраивание бизнес-логики



Эффективный инференс

llama.cpp

- Пишите на плюсах, чтобы быстро работало
- Минимум зависимостей
- Собственный формат хранения моделей
- Очень быстрый инференс
- Поддержка множества CPU/GPU архитектур
- Поддержка большинства текстовых и мультимодальных моделей
- Кванты
- Биндинги к другим языкам
- REST обвязки



Эффективный инференс

vLLM или Text Generation Inference

- Эффективный PagedAttention
- SOTA Throughput
- Непрерывный и эффективный батчинг
- Префил и кэши
- OpenAI-совместимое API и python биндинги
- Множество моделей, LoRa, MoE, VLM



Надежный инференс

Для того, чтобы работали сложные системы типа рагов нужно быть уверенным в формате ответа

- Tooling требует структуры
- Код требует структуры
- Чейны требуют структуру
- Production решения требуют структуры

Структурированная генерация

И как с ней взаимодействовать?

ИНСТРУКЦИЯ

Ты - детектор интенций и слотов голосового помощника "Алиса". Твоя задача выделить из запроса пользователя его интенцию и соответствующие слоты. Результат верни в json формате.

Список доступных интенций и их слотов:

set_timer (time, volume, ringtone)
weather_forecast (date, time, location)
question_answering (question)
play_audio (audio_name)
chatter (message)

Запрос пользователя:

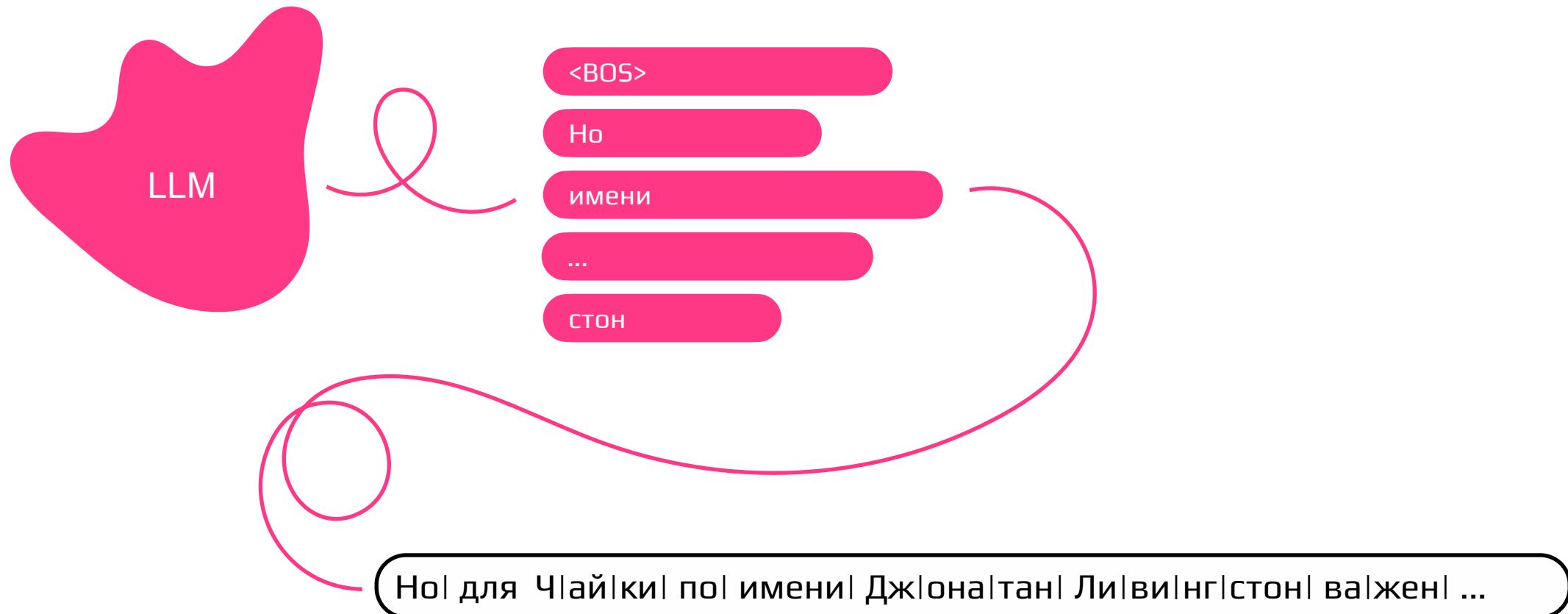
Че у нас по погодке в спб послезавтра

ВХОДНЫЕ ДАННЫЕ

ОТВЕТ

```
{  
    "intention": "weather_forecast",  
    "slots": {  
        "date": "послезавтра",  
        "time": "",  
        "location": "Санкт-Петербург"  
    }  
}
```

А откуда у нее выходит текст?



Хорошо, и насколько она обобщилась?

- Генерация текста: суммаризация, перевод, исправление ошибок, диалоги
- Работа с кодом: анализ, генерация, рефакторинг
- Извлечение неструктурированных сущностей: NER, произвольный теггинг, QA
- Классификация
- Toolings

Какие подводные?

Мы ожидаем, что языковая модель вернет нам ответ в нужном формате

- Иногда это работает
- А иногда нет 🤪

Как заставить модель уверенно генерировать текст в нужном формате?

Что такое структурированная генерация?

Возможность указать ожидаемый формат ответа

- Регулярные выражения

```
^"result": \"(ok|toxic)\"/
```

- JSON или XML

```
"Name": {"enum": ["John", "Paul"], "title": "Name", "type": "string"}
```

- Формальные языки

- SQL
- Python, C++, Java, etc.

Способ №1: Инструкции с форматом

Ты - детектор интенций и слотов голосового помощника "Маруся". Твоя задача выделить из запроса пользователя его интенцию и соответствующие слоты. Результат верни в json формате.

Список доступных интенций и их слотов:

set_timer (time, volume, ringtone)
weather_forecast (date, time, location)
question_answering (question)
play_audio (audio_name)
chatter (message)

- Укажем формат в инструкции
- Ре-генерация может стать решением

Способ №2: Две последовательных инструкции

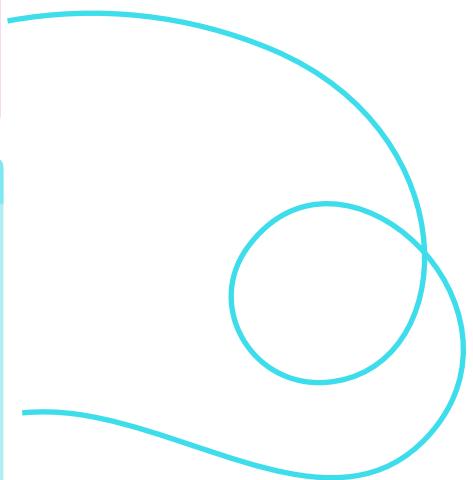
Ты - детектор интенций и слотов голосового помощника "Маруся". Твоя задача выделить из запроса пользователя его интенцию и соответствующие слоты в удобном для тебя формате.

Отформатирай предыдущий ответ со слотами и интенциями. Результат верни в json формате.

Список доступных интенций и их слотов:

set_timer (time, volume, ringtone)
weather_forecast (date, time, location)
question_answering (question)
play_audio (audio_name)
chatter (message)

- Один промпт для генерации
- Второй промпт для форматирования



Способ №3: Ограничение семплирования

- На каждом шаге мы генерируем маску на словарь для неподходящих токенов
- Замаскированные токены не семплируются, выбираем только из подходящих



Способ №3: Пример с конечным автоматом

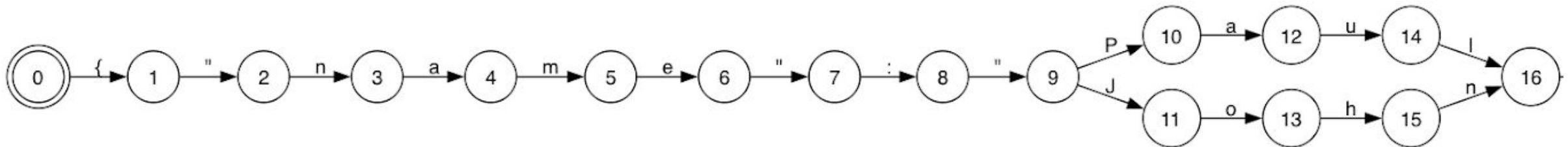
Как формализовать ограничения?

{ "name": "John", "age": 20 } или { "name": "Paul", "age": 30 }

В виде регулярного выражения:

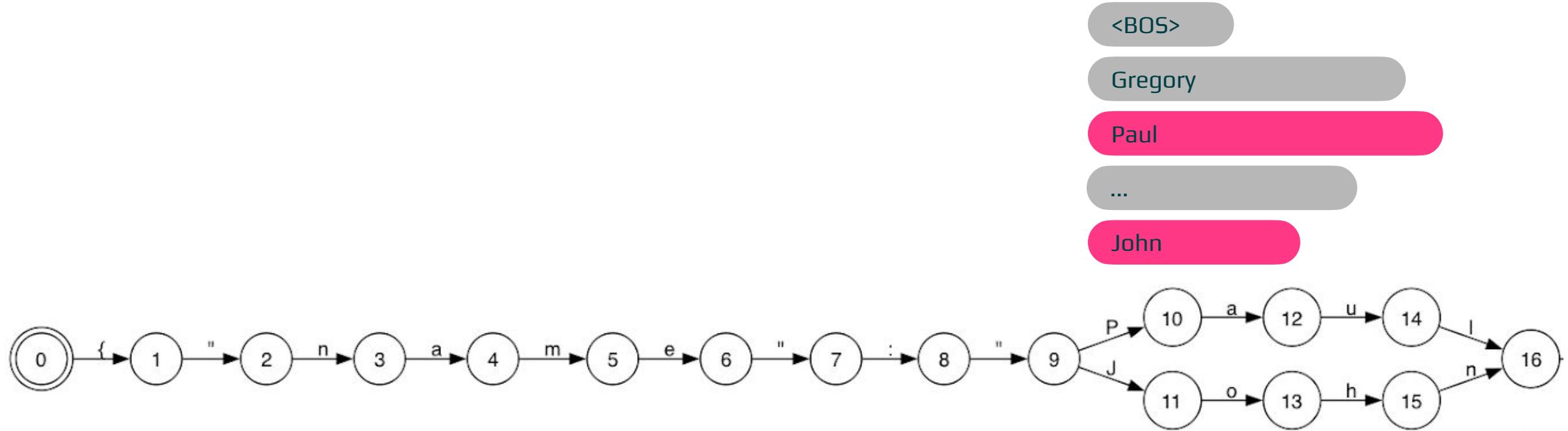
/\{"name":("John"\| "Paul"), "age":(20|30)\}/

Регулярное выражение ~ Конечный Автомат – Finite-State Automaton (FSA)



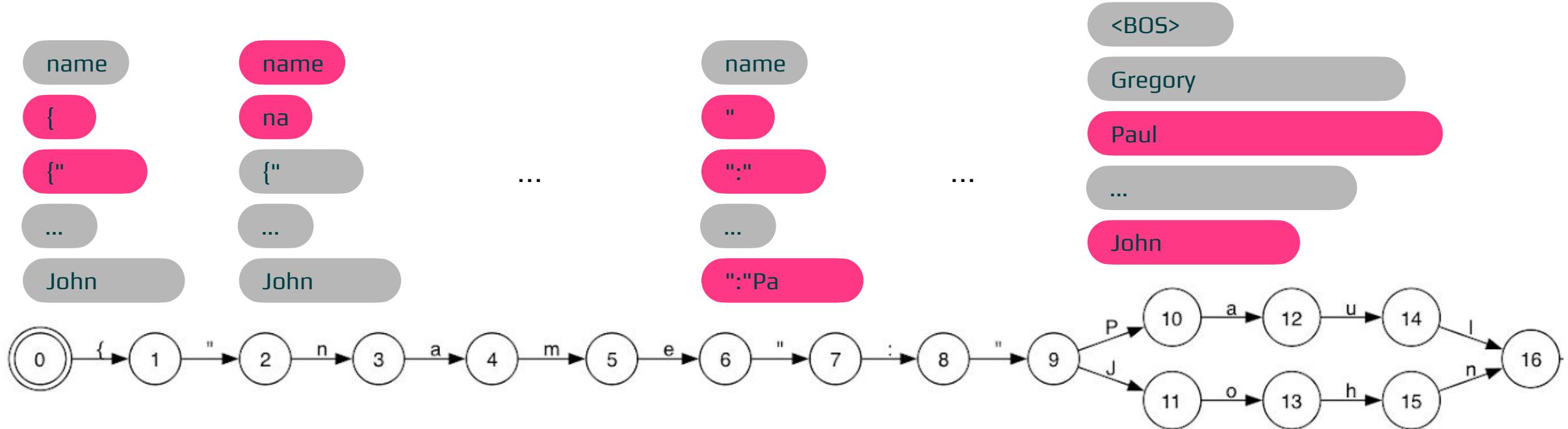
Способ №3: Наивный подход

- В каждом состоянии автомата выбираем доступные токены
- Двигаемся по автомату по мере генерации последовательности



Способ №3: Наивный подход

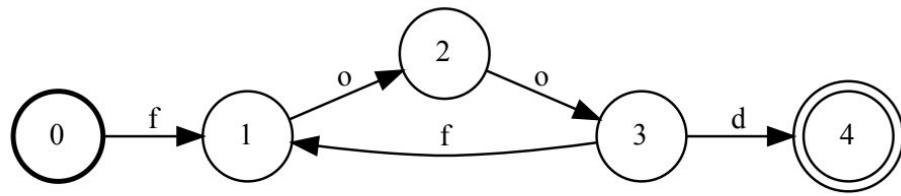
- Для каждого состояния в FSA можно заранее посчитать допустимые токены
- Долго компилируется и сильно разрастается



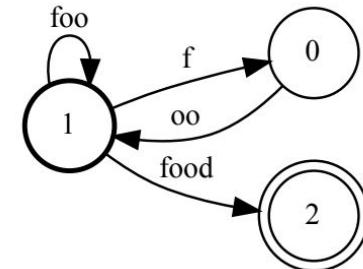
Способ №3: FSA поверх токенов

- FSA – конечный автомат
- FST – трансдуктор (автомат, отображающий токены в символы)
- $\text{FSA}(\text{FST}(\cdot)) = \text{FSA}'(\cdot)$

$/(\text{foo})^+d/$



FSA по символам



FSA по токенам

Как это работает на примере outlines

```
prompt = "What is the IP address of the Google DNS servers? "
unguided = generate.continuation(model, max_tokens=30)(prompt)
guided = generate.regex(
    model,
    r"((25[0-5]|2[0-4]\d|[01]?\d\d?)\.)\{3\}(25[0-5]|2[0-4]\d|[01]?\d\d?)",
    max_tokens=30
)(prompt)

print(unguided)
# What is the IP address of the Google DNS servers?
# ...
print(guided)
# What is the IP address of the Google DNS servers?
# 2.2.6.1
```

JSON via Compressed Prefill

Chunked Prefill + Compressed FSA

```
@guidance
def character_maker(lm, name):
    lm += f"""
        {name} is a character in Harry Potter. Please fill in the following information about this character.
        {{
            "name": "{guidance.gen("name", regex=r"[\w\d\s]+")}",
            "age": {guidance.gen("age", regex=r"[0-9]+")},
            "house": "{guidance.select(options=["Gryffindor", "Slytherin", "Ravenclaw", "Hufflepuff"], name="house")}"
        }}
        """
    return lm
```

Interleaved Syntax

Хорошо, а что тогда с формальными языками?

- Появляются, когда нужно генерировать SQL запросы или код
- FSAs уже не подходят, нужна контекстно-свободная грамматика
- Нужно использовать PDAs (Push-down Automata) или LALR(1), LR(1) парсеры
- SynCode и Outlines

Эффекты структурированной генерации



Влияет на скорость

- Нужно время на компиляцию
- Немного замедляет генерацию
- Chunked-Prefill ускоряет в 2-2.5 раза, но зависит от размера JSONa

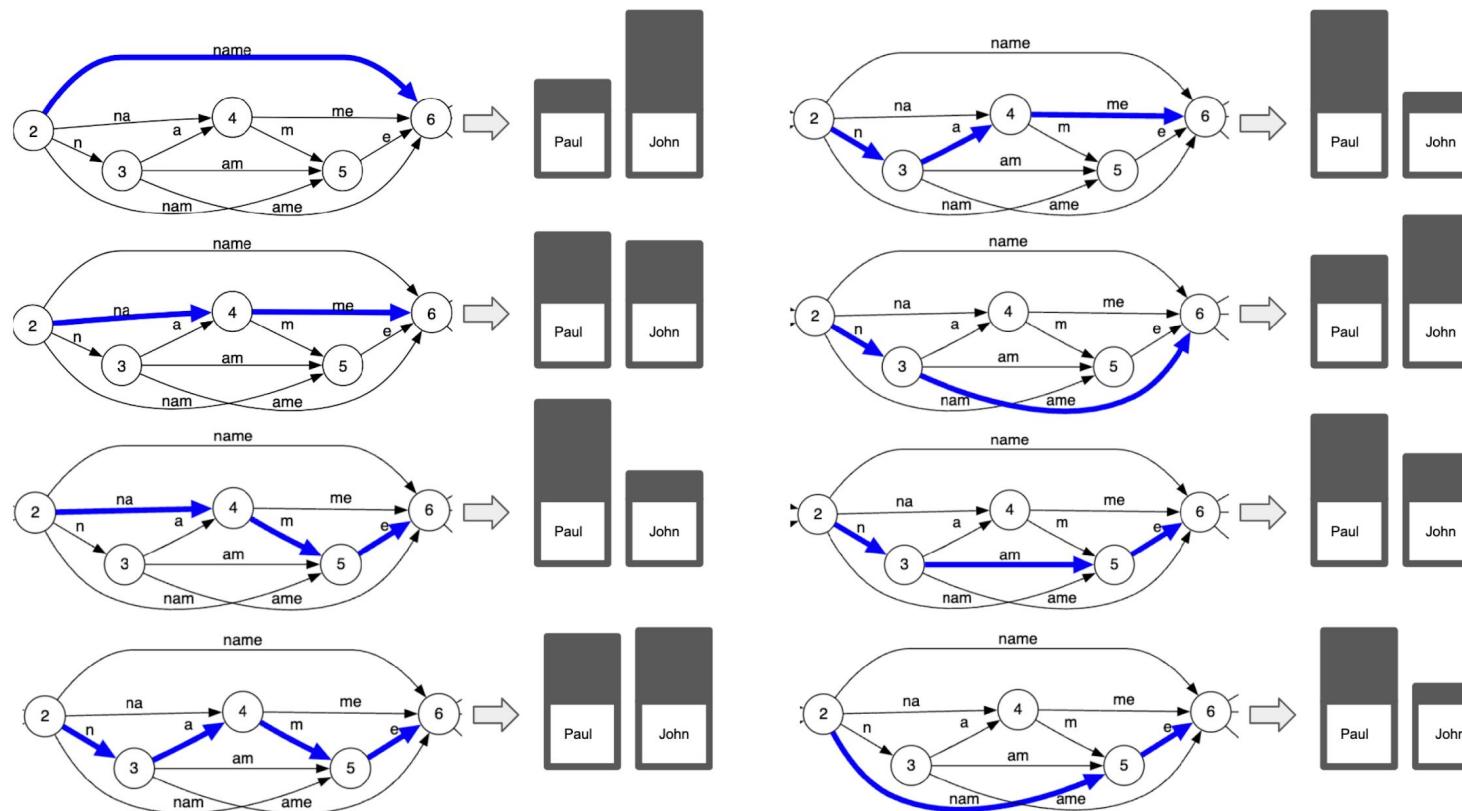
Влияет на качество

- Иногда токенизация объединяет то, что должно быть разделено
- Довольно сложно поддержать сущности вида "о(1" в случае грамматик

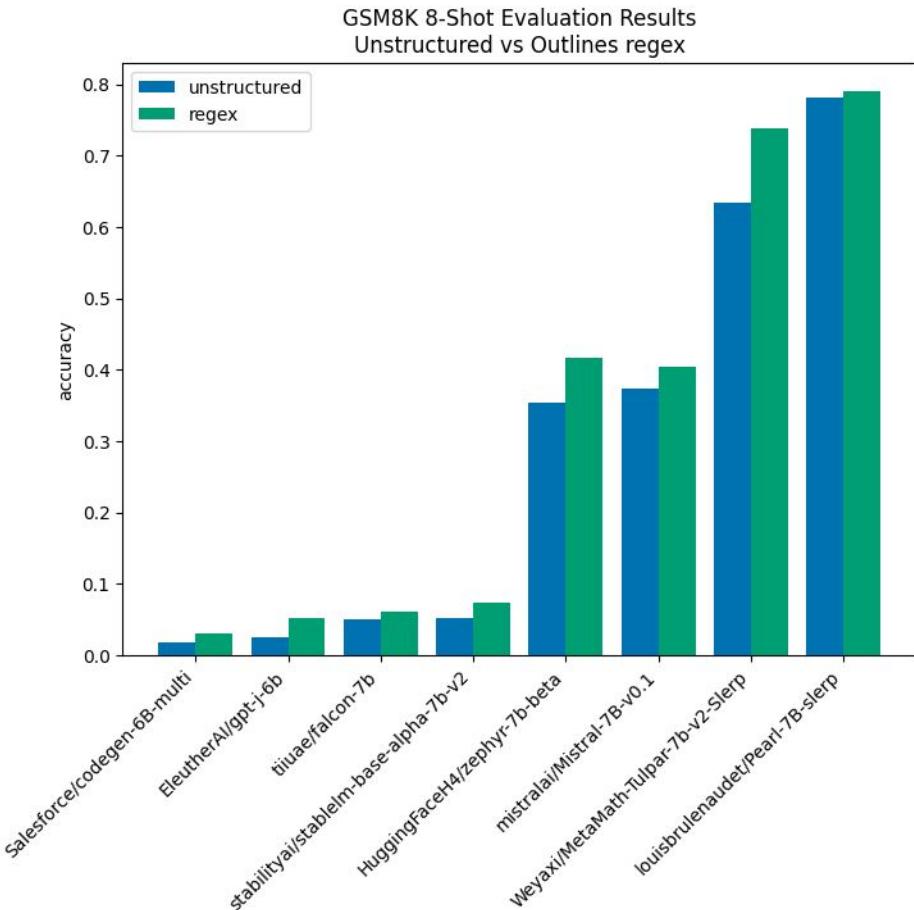
Score	Tokenization
0.4	fo o(1 2 3)
0.3	bar (456)
0.2	foo (123)
0.1	ba r(4 5 6)

Влияет на качество

- Разное правдоподобие в зависимости от токенизации



Влияет на качество



Structured Generation Improves LLM performance: GSM8K Benchmark

Model	Text	JSON	XML	YAML
<u>gemini-1.5-flash</u>	89.33 (0.8)	89.66 (0.3)	89.26 (0.3)	89.21 (0.4)
+ schema constraint	-	89.21 (1.5)	88.20 (2.2)	87.42 (3.7)
<u>claude-3-haiku</u>	86.51 (0.8)	86.99 (0.2)	86.96 (0.6)	82.89 (5.7)
+ schema constraint	-	23.44 (22.9)	79.76 (7.0)	80.63 (2.8)
<u>gpt-3.5-turbo</u>	75.99 (3.1)	74.70 (1.1)	60.45 (7.2)	71.58 (3.0)
+ schema constraint	-	49.25 (12.0)	45.06 (19.9)	73.85 (5.6)
<u>LLaMA-3-8B</u>	75.13 (0.9)	64.67 (2.23)	65.07 (0.56)	69.41 (0.95)
+ schema constraint	-	48.90 (6.7)	56.74 (8.3)	46.08 (16.8)

Table 1: Comparing results without and with schema constraint, adding schema not only increase the sensitivity to prompt but also degrade in average performance.

Let Me Speak Freely? A Study on the Impact of Format Restrictions on Performance of Large Language Models

Что, где, когда использовать?

- Не попробуете – не узнаете
- Считайте Error Rate парсинга
- Сравнивайте качество при ограничениях и без
- Особенно полезно в задачах с кодом

Спасибо за внимание!

Андрей Соколов

