

GOR 2025 Workshop: Structured Information Extraction with LLMs

Paul Simmering
Q Agentur für Forschung

1 Introduction

1.1 About

- I'm a Senior Data Scientist at Q Agentur für Forschung
- Market research agency from Mannheim
- Classic market research + data science
- Using language models since 2019 for social media and review analysis

This workshop builds on the GOR24 presentation “Where do LLMs fit in NLP pipelines?” by my colleague Dr. Paavo Huoviala and me.

1.2 Goals for this workshop

1. Get hands-on experience with structured information extraction
2. Get an overview of available models and tools
3. Learn about evaluation, efficiency and limitations
4. Share experiences and use cases

1.3 Agenda 10:00 - 12:30

Start time	Topic
10:00	Introduction & overview
10:15	Setting up the environment
10:30	Notebook and exercises
11:10	Micro-Hackathon
11:40	Discussion: classic NLP vs LLM
12:00	Models, tools and best practices
12:15	Q&A and wrap-up

Feel free to take a break during the exercises.

1.4 Workshop organization

- Ask questions at any time
- Notebook is on [Google Colab](#)
- All workshop materials are available at github.com/qagentur/structured-information-extraction-with-llms

1.5 Quick survey

- Have you used a chat assistant, like ChatGPT?
- Do you know Python?
- Do you know R?
- Have you used an LLM API?
- Have you used a local LLM or set up your own server?
- Have you used tool calling or structured output?

2 LLMs and structured output

2.1 Types of language model outputs

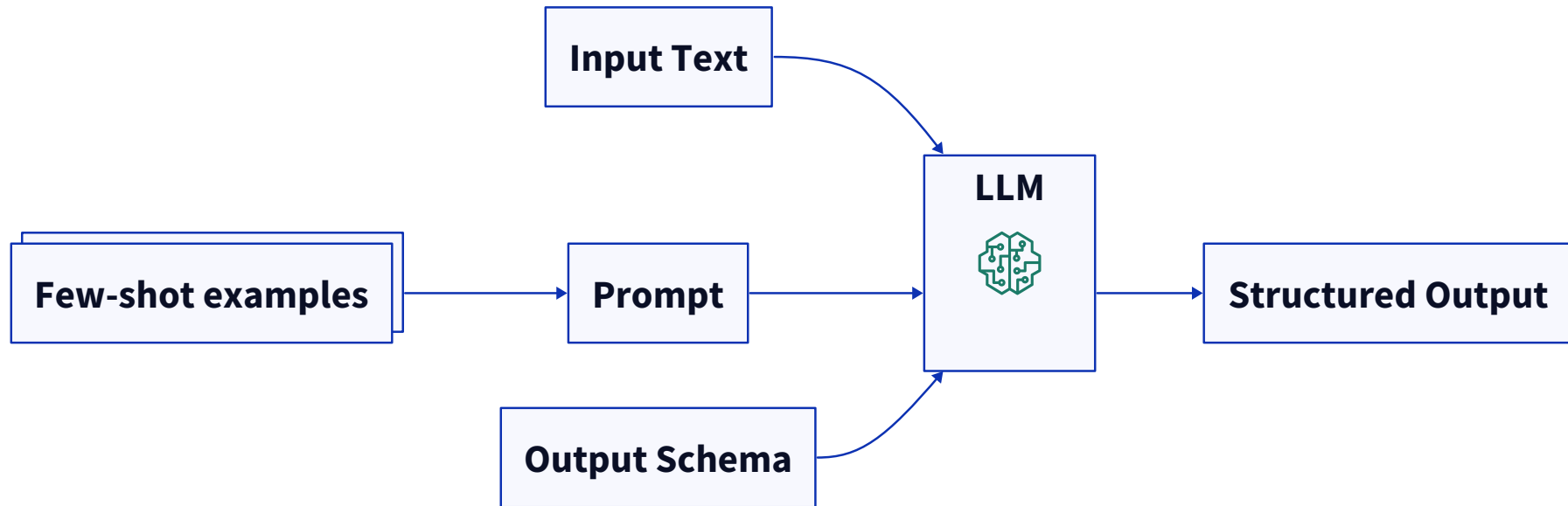
Unstructured output

- Chat
- Creative writing
- Code generation
- Summarization
- Translation
- Reasoning

Structured output

- Text classification
- Named entity recognition
- Attribute & relation extraction
- Sentiment analysis
- Entity linking
- Part of speech tagging
- Claim extraction

2.2 Information extraction with LLMs



Basic information extraction pipeline



2.3 LLM architecture is inefficient

Encoder-decoder (BERT)

$P(\text{Class 1})$

$P(\text{Class 2})$

$P(\text{Class 3})$

-  Directly read probabilities
-  Clear loss function (cross entropy)



Decoder-only (GPT)

$P(\text{Token 1})$

$P(\text{Token 2})$

⋮

$P(\text{Token } k)$

-  Need to teach output format
-  Training loss is not the classification loss

2.4 Why use an LLM for structured information extraction?

- Use built-in knowledge
- Training examples are optional
- Single model for multiple tasks
- Multiple tasks at once
- Analyze text and images together
- Easy to use via an API
- Use in agentic workflows

3 Notebook and exercises

3.1 Jupyter notebook setup

- Use Colab (easier, requires Google account) or your own computer
- Colab is a free cloud environment to run Python notebooks
- Use a T4 GPU on Colab
- Run the first few cells to install the necessary packages and download the model

Link to the notebook on Colab:

<https://colab.research.google.com/drive/1lpQDRxTx3tnbChFqGUOusp=sharing>

3.2 Notebook

- Let's run the notebook together
- I'll add more background information and alternatives later
- Use the AG News dataset at
https://huggingface.co/datasets/fancyzhx/ag_news
- AG News is a dataset of news article snippets with 4 possible categories: World, Sports, Sci/Tech, and Business

3.3 Exercise 1: Prompt the model

Run the notebook cells and see if you receive an answer from the model.

- Can you make it answer with just the name of the news category?
- What could go wrong?

3.4 Exercise 2: Structured output

Modify the examples and tinker with them. Some ideas:

- Add a new news category
- Add a new type of entity
- Extract the intensity of the sentiment
- Find a case where the model fails to give the right answer
- Implement chain-of-thought:
 - Use a free form text field as the first field in structured output
 - Alternatively, use a two-step process with an unstructured and a structured output

3.5 Exercise 3: Multi task prompt

Modify the NewsAnalysis class to extract more information from the news text.

Find other long news texts on the web and see if the model can extract the information.

3.6 Exercise 4: Few shot learning

Add another news category and an example for it. Test your classifier with some examples.

3.7 Exercise 5: Eval & optimization

Let's have a little **competition** and **see how high we can get the accuracy**. Everything is allowed, except for changing the model or using information from the test set. Collaboration is encouraged.

Competiting is optional. If you'd rather just try out things at a slower pace, that's fine too 😊

3.8 Boosting accuracy 🚀

- **More examples:** Add few-shot examples
- **Better definitions:** Detailed definitions of each category
- **Chain-of-thought:** Add an intermediate step where the model thinks out loud
- **Self-consistency:** Ask the model to check its own answer
- **Binary classification:** Ask “is it this category?” separately for each category

3.9 Exercise 5: Winners

- Did a team reach **100%** accuracy?
- Did a team reach **95%** accuracy?
- Did a team reach **90%** accuracy?
- Did a team reach **85%** accuracy?
- Did a team reach **80%** accuracy?
- Did a team reach **75%** accuracy?
- Did a team reach **70%** accuracy?

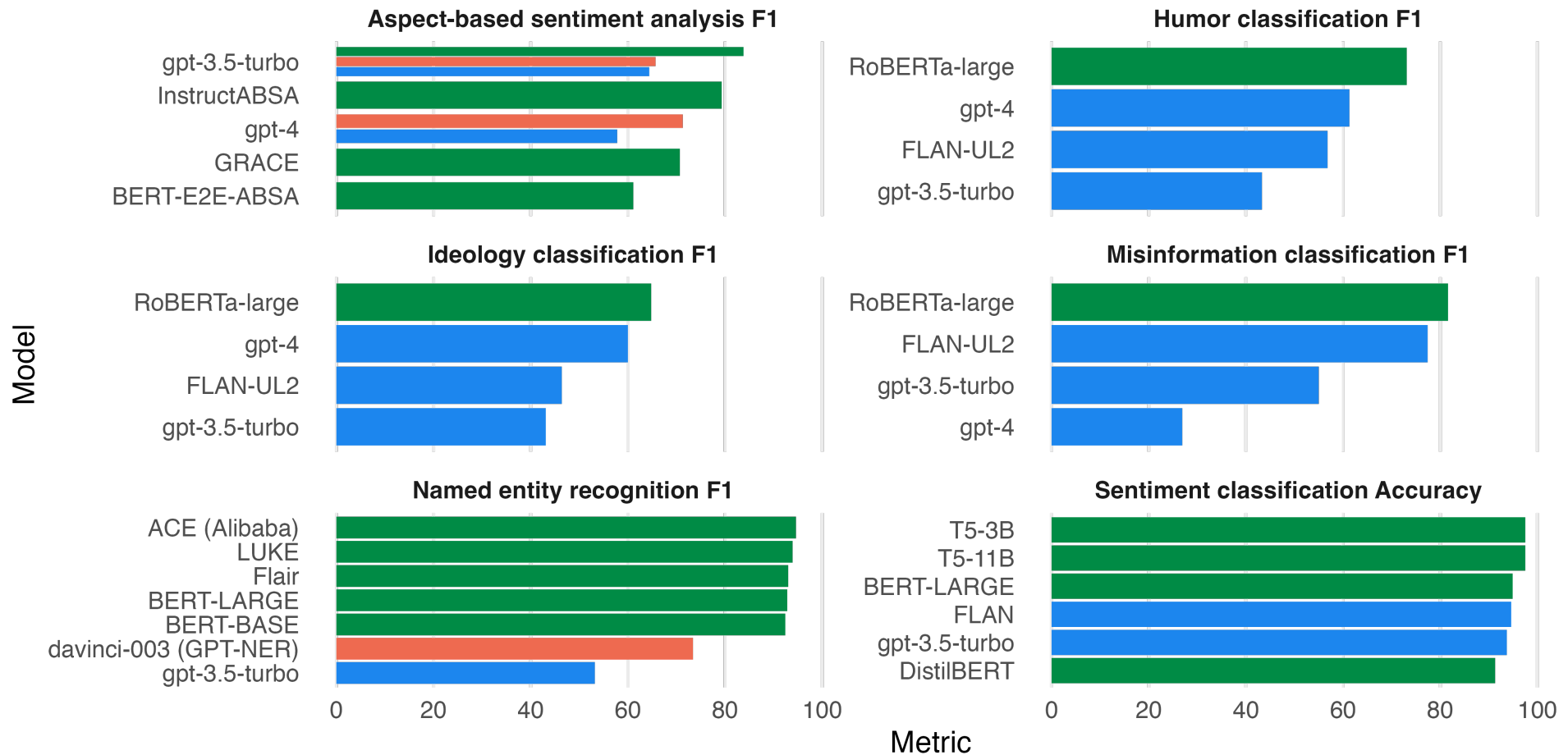
3.10 Recap of the competition and exercises

- What did you try?
- What worked well, what didn't?

4 Classic NLP vs LLMs

4.1 Fine-tuning still beats prompting

Setting ■ fine-tuned ■ few-shot ■ zero-shot



Benchmarks

4.2 Benchmark data sources

Task	Dataset	Sources
Aspect-based sentiment analysis: aspect extraction and polarity classification	SemEval 2014 (Pontiki et al. 2014) Task 4 1+2	Simmering, Paul F., and Paavo Huoviala. "Large language models for aspect-based sentiment analysis." <i>arXiv preprint arXiv:2310.18025</i> (2023).
Named entity recognition: extraction of entities and classification by type	ConLL 2003 (Sang et al. 2003)	Wang, Shuhe et al. "GPT-NER: Named entity recognition via large language models". <i>arXiv preprint arXiv:2304.10428</i> (2023).
Sentiment analysis: Classification of texts as positive, neutral, negative	SST 2 (Socher et al. 2013)	Qin, Chengwei, et al. "Is ChatGPT a general-purpose natural language processing task solver?." <i>arXiv preprint arXiv:2302.06476</i> (2023).
Ideology: Classification of political leaning as liberal, conservative or neutral	Ideological Books Corpus (Gross et al. 2013)	Ziems, Caleb, et al. "Can large language models transform computational social science?." <i>Computational Linguistics</i> (2023): 1-53. https://arxiv.org/pdf/2305.03514.pdf
Misinformation: Binary classification on news article headlines	Misinfo Reaction Frames (Gabriel et al. 2022)	
Humor: Binary classification of internet texts	Humor classification (Davies 2017)	

Benchmark data sources

4.3 Discussion: When to use which method?

- Manual prompting
- Automated prompt engineering
- Fine-tuning SLM, e.g. ModernBERT or specialized models like GliNER
- Fine-tuning LLM

5 Models, tools and best practices

5.1 Best open weights LLMs for structured information extraction

Version	Parameters	Tools	Vision
Llama 3.1	8B, 70B, 405B	✓	✗
Llama 3.2	1B, 3B, 11B, 90B	✓	✓ (11B, 90B)
Llama 3.3	70B	✓	✗
Gemma 3	27B	✓	✓

Also good: Phi (Microsoft) and Qwen (Alibaba). Close models tend to be ~6 months ahead of open models. [Zhou et al. 2024](#) noted diminishing returns for larger models past 8B parameters.

5.2 Reasoning models

- Built-in chain of thought
- Top benchmark scores on logical tasks (math, science, coding)

→ high potential for correct information extraction

But they are even slower than LLMs and have higher costs due to thinking token generation. Not the right choice for high-volume applications.

5.3 Alternatives inference engines

We've used Llama.cpp for the demo, which let us run the models without needing an external API.

- **vLLM** for large scale inference with your own GPUs
- **API providers** like OpenAI, Anthropic, Google, Fireworks.ai, Groq...

Huge differences in cost and performance between models and providers. Use artificialanalysis.ai or llm-stats.com to compare models and providers.

Considerations: available models, cost, throughput, latency, privacy, security

5.4 Ways to get structured output

By default LLMs can generate any token at any position. To use it in a pipeline, we need to constrain the output.

- **Tool calling:** Provide a JSON schema that the model's answer must conform to. Can be used to let the model call a function. Widely supported.
- **Constrained generation:** Limit which tokens can be generated so that the output conforms to an arbitrary schema. Most efficient, but not as widely supported.
- **JSON mode:** Request that the output is always a valid JSON object. Not recommended.

5.5 Alternative client libraries

In Python:

- [mirascope](#): tool calling
- [marvin](#): tool calling
- [outlines](#): constrained generation
- [sglang](#): constrained generation

In R:

- [ellmer](#)

Any programming language: directly specifying JSON schemas for your API calls

5.6 Monitoring systems

Track everything:

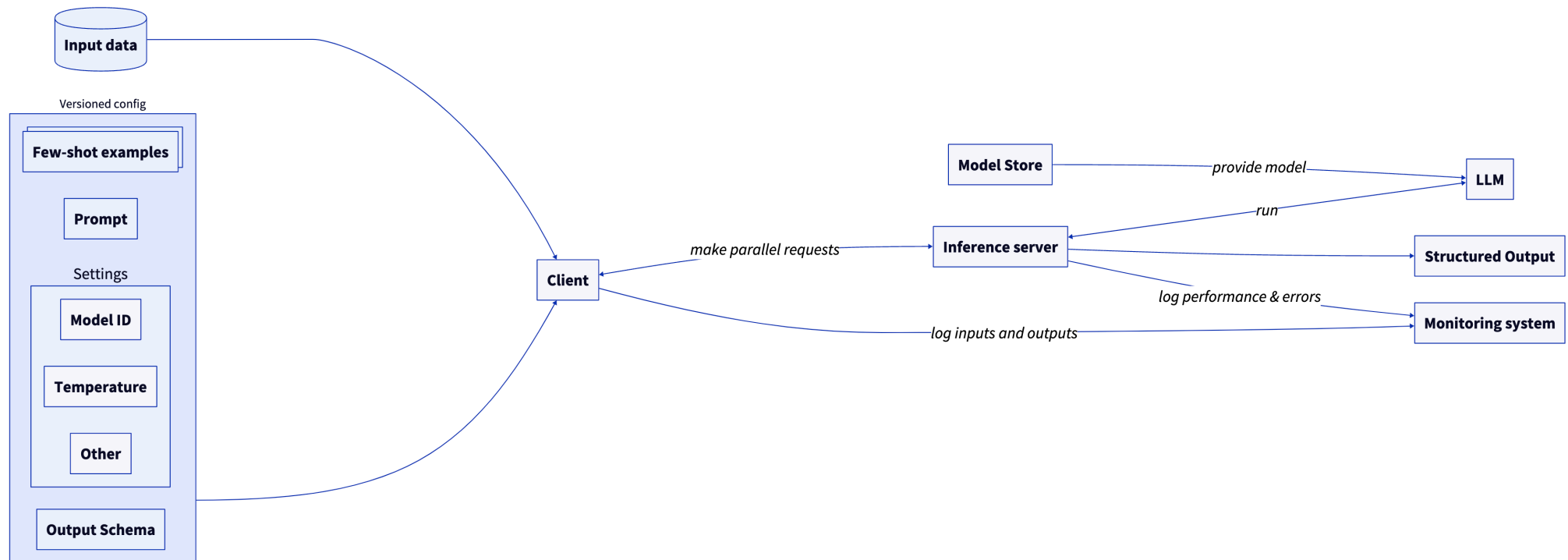
- Inputs: prompts, schemas, settings
- Outputs: responses, latency, token usage, cost, errors
- Model versions
- Use tags to group traces

Stay provider agnostic to be able to switch providers later.

Many options, typically open core with commercial addons:

[Arize AI](#), [Helicone](#), [Inspect](#), [Langsmith](#), [openllmetry](#), [W&B Weave](#) and dozens more.

5.7 Production system overview



6 Wrap-up

6.1 Summary

- LLMs are powerful tools for structured output
- There are many open source models and libraries available
- Classic models are still strong and are more efficient
- Fine-tuning is required to get the highest accuracy

6.2 Contact

Website: teamq.de

Workshop materials: github.com/qagentur/structured-information-extraction-with-llms

Email: paul.simmering@teamq.de

We offer market research services and AI consulting.

I would enjoy hearing about what you build with LLMs!