Text Analysis with Large Language Models

An Introduction and Best-Practice Guide

Violeta Haas
violeta.haas@iast.fr

29th January 2025

Postdoctoral Research Fellow
Institute for Advanced Study in Toulouse
Toulouse School of Economics
University of Toulouse Capitole

Introduction

Hi 👋

About myself:

- Violeta Haas
- Research Fellow at IAST and TSE
- BA, MA & PhD at HU and DYNAMICS

Research Interests:

Comparative politics and political behavior: *how institutions and elites shape public perceptions and inter-group relations*, with a special focus on sexuality and gender. I also study electoral behavior, party competition, social norms and movements.

Objectives

What we'll cover today:

- Short introduction to Large Language Models (LLMs).
- Step-by-Step Guide for text analysis with LLMs.
    - Model Selection
    - Prompt Engeneering and Parameter Tuning
    - Prompt Validation
    - Model Evaluation
- Applied example using GPT API.
- (How to avoid the next Replication Crisis)

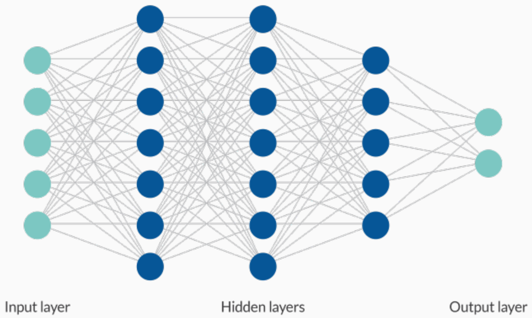
Goal: Introduction and best-practice guide for text analysis with LLMs

Interactive: Feel free to ask questions throughout!

Two Disclaimers

1. Introductory Session:
   - This is an *introductory session*—we will only cover the basics.
   - Advanced topics like fine-tuning and detailed parameter adjustments are beyond today's scope (but see ”Appendix”).
2. Rapidly Evolving Field:
   - LLMs are a *rapidly evolving field.*
   - Best-practice advice is often a work in progress.
   - What is true today may change tomorrow as new techniques and models emerge.

Why Use LLMs for Text Analysis?



What are LLMs?

- Deep learning models trained on massive text corpora to understand and perform human-like behavior.
- Built using transformer architectures consisting of neural networks with many layers (hence ”deep” learning). *Examples:* GPT, BERT, LLaMa and their variants.

Why Use LLMs for Text Analysis?

Applications in Social Sciences:

- Label topics and frames of documents (e.g., Gilardi et al., 2023)
- Analyzing open-ended responses (e.g., Mellon et al., 2024)
- Act as treatments in experiments (e.g., Argyle et al., 2023)
- Generate data by simulating respondents (e.g., Argyle et al., 2023)
  - → Not uncontested (Bisbee et al., 2024, Dominguez-Olmedo et al., 2023, von der Heyde et al., 2024)

LLMs have outperformed humans across a variety of classification and annotation tasks for a fraction of the cost (Bang et al., 2023, Qin et al., 2023, Goyal et al., 2022, Chiang and Lee, 2023, Grossmann et al., 2023, Ziems et al., 2024).

Why Use LLMs for Text Analysis

Ease-of-use, high accuracy and relatively low costs!

Step-By-Step Guide: Text Analysis with LLMs

1. Model Selection
2. Prompt Engineering and Parameter Tuning
3. Prompt Validation
4. Model Evaluation

Model Selection: Proprietary vs. Open-Source Models

Two Catgories of LLMs:

- *Propriatary:* LLMs developed and maintained by private entities with exclusive rights to the model's code, data, and usage.
- *Open-Source:* LLMs whose code, architecture, and sometimes training data are publicly accessible.

	Proprietary LLMs	Open-Source LLMs
Ownership	Retained by company	Community or developers
Access	Paid or restricted	Free or low-cost
Transparency	Limited	Full
Customization	Limited	High
Examples	GPT	LLaMA

Model Selection: Proprietary Models

Challenges with Proprietary Models:

- *Privacy concerns:* Commercial APIs require sending data to service providers, who may use it for training.
- *Lack of transparency:* Training data and methodologies are unknown, potentially leading to bias.
- *Unpredictable updates:* Models can change over time without notice or depreciate, altering results and making replication impossible.

Model Selection: Open-Source Models

Challenges of Open-Source Models:

- *Performance Gaps:* May not match proprietary models' performance, especially for complex tasks.
- *Scalability Challenges:* Often not optimized for large-scale deployment.
- *Setup Costs:* May require access to powerful hardware and more expertise in machine learning and model optimization.

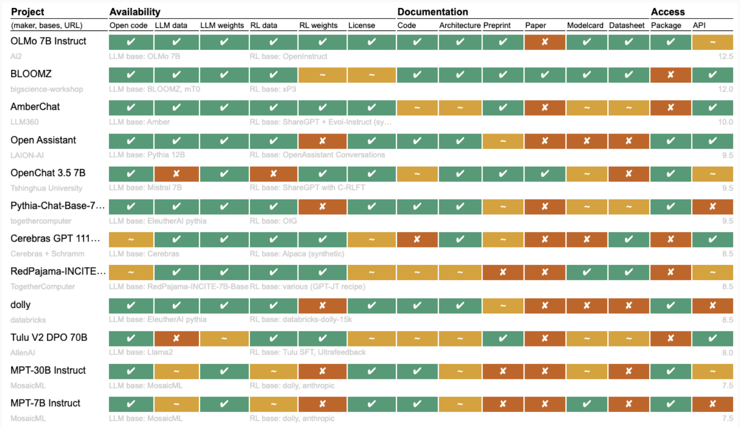
Model Selection: Self-Hosting

Why host models on your own infrastructure?

- Control over model version and updates without relying on external services (improves reproducibility).
- Ensures sensitive or confidential information is handled securely without exposing it to third-party APIs.

Model Selection: ”Open-Washing”

Not all downloadable models are open-source models!



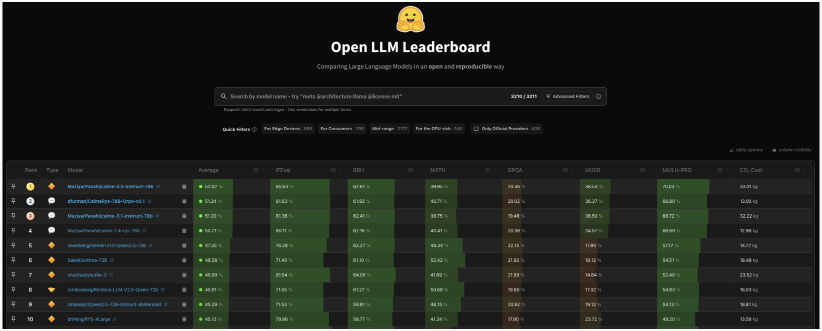
Project	Availability						Documentation						Access	
(maker, bases, URL)	Open code	LLM data	LLM weights	RL data	RL weights	License	Code	Architecture	Preprint	Paper	Modelcard	Datasheet	Package	API
OLMo 7B Instruct	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	~
Ai2	LLM base: OLMo 7B			RL base: OpenInstruct										12.5
BLOOMZ	✔	✔	✔	✔	~	~	✔	✔	✔	✔	✔	✔	✘	✔
bigscience-workshop	LLM base: BLOOMZ, mT0			RL base: xP3										12.0
AmberChat	✔	✔	✔	✔	✔	✔	~	~	✔	✘	~	~	✘	✔
LLM360	LLM base: Amber			RL base: ShareGPT + Evol-Instruct (sy…										10.0
Open Assistant	✔	✔	✔	✔	✘	✔	✔	✔	~	✘	✘	✘	✔	✔
LAION-AI	LLM base: Pythia 12B			RL base: OpenAssistant Conversations										9.5
OpenChat 3.5 7B	✔	✘	✔	✘	✔	✔	~	✔	✔	✔	~	✘	✔	~
Tshinghua University	LLM base: Mistral 7B			RL base: ShareGPT with C-RLFT										9.5
Pythia-Chat-Base-7…	✔	✔	✔	✔	✘	✔	✔	✔	~	✘	~	~	✔	✘
togethercomputer	LLM base: EleutherAI pythia			RL base: OIG										9.5
Cerebras GPT 111…	~	✔	✔	✔	✔	~	✘	✔	~	✘	✘	✔	✘	✔
Cerebras + Schramm	LLM base: Cerebras			RL base: Alpaca (synthetic)										8.5
RedPajama-INCITE…	~	✔	✔	✔	✔	~	~	~	✘	✘	✔	✔	✘	~
TogetherComputer	LLM base: RedPajama-INCITE-7B-Base			RL base: various (GPT-JT recipe)										8.5
dolly	✔	✔	✔	✔	✘	✔	✔	✔	~	✘	✘	✘	✔	✘
databricks	LLM base: EleutherAI pythia			RL base: databricks-dolly-15k										8.5
Tulu V2 DPO 70B	✔	✘	~	✔	✔	~	~	~	✔	✘	~	~	✘	✔
AllenAI	LLM base: Llama2			RL base: Tulu SFT, Ultrafeedback										8.0
MPT-30B Instruct	✔	~	✔	~	✘	✔	✔	~	✘	✘	~	✘	✔	~
MosaicML	LLM base: MosaicML			RL base: dolly, anthropic										7.5
MPT-7B Instruct	✔	~	✔	~	✘	✔	✔	~	✘	✘	✔	✘	✔	✘
MosaicML	LLM base: MosaicML			RL base: dolly, anthropic										7.5

Benchmarks for openness (Liesenfeld et al., 2023)

Link to Github

Model Selection: Open LLM Leaderboard

Tracks carbon dioxide emission!



Open LLM Leaderboard

Comparing Large Language Models in an **open** and **reproducible** way

Search by model name • try "meta @architecture:llama @license:mit" 3210 / 3211 Advanced Filters

Supports strict search and regex • Use semicolons for multiple terms

Quick Filters: For Edge Devices · 586 | For Consumers · 296 | Mid-range · 2177 | For the GPU-rich · 148 | Only Official Providers · 436

table options · column visibility

Rank	Type	Model	Average	IFEval	BBH	MATH	GPQA	MUSR	MMLU-PRO	CO_2 Cost
1	🔶	MaziyarPanahi/calme-3.2-instruct-78b	● 52.02 %	80.63 %	62.61 %	39.95 %	20.36 %	38.53 %	70.03 %	33.01 kg
2	💬	dfurman/CalmeRys-78B-Orpo-v0.1	● 51.24 %	81.63 %	61.92 %	40.71 %	20.02 %	36.37 %	66.80 %	13.00 kg
3	💬	MaziyarPanahi/calme-3.1-instruct-78b	● 51.20 %	81.36 %	62.41 %	38.75 %	19.46 %	36.50 %	68.72 %	32.22 kg
4	💬	MaziyarPanahi/calme-2.4-rys-78b	● 50.71 %	80.11 %	62.16 %	40.41 %	20.36 %	34.57 %	66.69 %	12.98 kg
5	🔶	newsbang/Homer-v1.0-Qwen2.5-72B	● 47.35 %	76.28 %	62.27 %	48.34 %	22.15 %	17.90 %	57.17 %	14.77 kg
6	🔶	Sakalti/ultiima-72B	● 46.58 %	71.40 %	61.10 %	52.42 %	21.92 %	18.12 %	54.51 %	18.48 kg
7	🔶	shuttleai/shuttle-3	● 45.99 %	81.54 %	64.05 %	41.69 %	21.59 %	14.64 %	52.40 %	23.52 kg
8	🤝	rombodawg/Rombos-LLM-V2.5-Qwen-72b	● 45.91 %	71.55 %	61.27 %	50.68 %	19.80 %	17.32 %	54.83 %	16.03 kg
9	🔶	zetasepic/Qwen2.5-72B-Instruct-abliterated	● 45.29 %	71.53 %	59.91 %	46.15 %	20.92 %	19.12 %	54.13 %	18.81 kg
10	🔶	dnhkng/RYS-XLarge	● 45.13 %	79.96 %	58.77 %	41.24 %	17.90 %	23.72 %	49.20 %	13.58 kg

Link to Hugging Face

Model Selection

6 Factors to consider (Part 1):

1. *Reproducibility:* Ensure results can be replicated by documenting and using a fixed LLM version.
2. *Ethics and legality:* Adhere to ethical and legal standards, respecting privacy and data regulations (e.g., not storing research data).
3. *Transparency:* Clearly document methodologies, assumptions, and limitations.

Model Selection

6 Factors to consider (Part 2):

4. *Culture and language:* Select models proficient in relevant languages and cultural contexts to avoid bias.
5. *Scalability:* Match the model to data size and resource constraints; consider semi-supervised options for large datasets.
6. *Complexity:* Use sophisticated models (e.g., GPT-4) for tasks requiring advanced reasoning or subtle meaning parsing.

$\rightarrow$ Simple decision tree in ”Appendix”

Model Choice

Best Practice:

→ Use *self-hosted, open-source models* with *publicly known training data.*

→ But, *trade-offs* between factors 1-3 and 4-6 can complicate use of open-source models.

→ Always offer *motivations* for model choice.

The best model ultimately depends on the task at hand!

The Least One Can Do...

Me typing confidential company info into ChatGPT so I know how to reply to an email
imgflip.com

Never use proprietary models with sensitive, secret, or private data!

Prompt Engineering and Parameter Tuning

Workflow: Systematic Coding Procedure

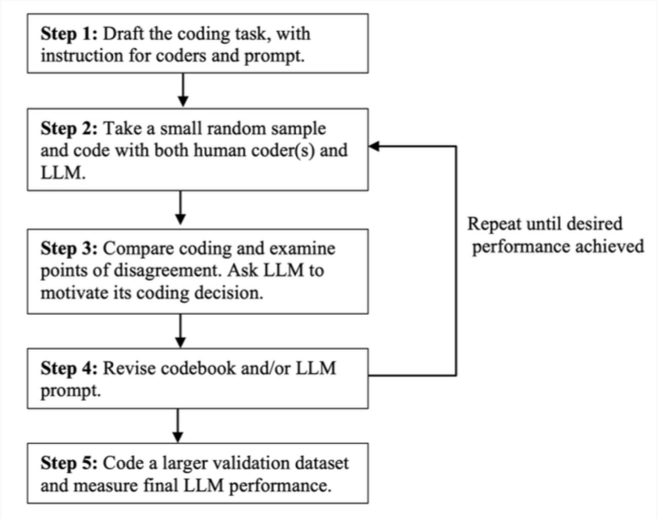

Step 1: Draft the coding task, with instruction for coders and prompt.
Step 2: Take a small random sample and code with both human coder(s) and LLM.
Step 3: Compare coding and examine points of disagreement. Ask LLM to motivate its coding decision.
Step 4: Revise codebook and/or LLM prompt.
Step 5: Code a larger validation dataset and measure final LLM performance.
Repeat until desired performance achieved

Figure 1: Example of a systematic coding procedure.

Source: Törnberg (2024)

Prompt Engineering

What is Prompt Engineering?

- Designing effective instructions (inputs) to guide LLM behavior.
- Requires multiple iterations of modification and testing.
- Poor theor. understanding of why some techniques work better.

A good prompt contains these five elements (in this order):

1. Context
2. Question/Instruction
3. Constraints
4. (Examples)
5. Input data

1. Context

Include background information:

- Add context that helps the model ”understand” its *role* and the *domain* of the input data.

Example: Context

You are an expert annotator for natural language processing tasks focusing on social media content analysis. Your role involves fact-checking Twitter messages related to the US 2020 election.

2. Question/Instruction

Start with a simple Question/Instruction

- Define Concepts and Tasks
- Use direct and instructional language.

Example: Instruction

Your expertise is critical in identifying misinformation that shares false or misleading content, which can distort facts, omit critical context, or misrepresent the truth. For each Twitter message, assess whether it contains misinformation regarding the US 2020 election by evaluating the factual accuracy of claims, sources, and context. Does this message contain misinformation?

2. Question/Instruction

Adapt specificity based on performance:

- Provide additional guidelines

Example: Additional Guidelines

Verify statements against reputable and publicly available sources like government websites, credible news outlets, and fact-checking organizations.

- Provide specific instructions where the model fails

Example: Failure Adaptation

Look for signs of altered or fabricated images, videos, or text like deepfakes, fake screenshots, or doctored evidence.

3. Constraints

Provide Categories and Constraints

- Give Classification Options
- Allow for Uncertainty

Example: Options

Options: Yes/No/Uncertain. If you are uncertain about the classification, choose ‘Uncertain’ and provide a rationale for this uncertainty.

- Use JSON output format

Example: Output format

Provide your response in JSON format, as follows: { ”contains_misinformation”: ”Yes/No/Uncertain”, ”justification”: ”Provide a brief justification for your choice.” }

Prompt Engineering: 4. (Examples)

Zero-shot prompting might not always deliver good results!

Improve prompt using in-context learning techniques:

- *Zero-shot prompting*: instruction without examples.
- *One-shot prompting:* instruction + one example.
- *Few-shot prompting:* instruction + two or more examples.
- *Negative prompting:* instruction + example of unwanted output

*For an Example Structure of each technique see ”Appendix”

5. Input Data

Insert Input Data:

- One-by-one or grouping multiple queries (i.e., Batching)

Example: Input Data

Twitter message: [MESSAGE]

*For Batching see ”Appendix”

Parameter Tuning: Temperature

What is Temperature?

- Controls randomness in output.
- Range: 0 (deterministic) to 1 (creative and diverse).
  - Low temperature (e.g., 0.0 or 0.2): Ensures consistency and accuracy.
  - High temperature (e.g., 0.8): Useful for subjective or exploratory tasks.

→ Use low temperature for annotation tasks to ensure that similar inputs produce similar outputs (*consistency*) and that the model focuses on the most probable completion (*accuracy*).

*For other parameter settings, see section ”Advanced”

Additional Tips & Materials

Me saying "thank you" to ChatGPT, so they spare my life when AI takes over the world

→ Impolite prompts reduce performance, while overly polite ones offer no guarantee of improvement.

→ optimal politeness varies by language/culture (Yin et al., 2024).

Additional Tips & Materials

Additional Tips for Prompting:

- Give an ”I don’t know” option.
- Break down tasks into several simpler intermediate steps.
- Use LLMs (like ChatGPT) to improve prompt language.
- Use affirmations (e.g., ”do” and ”don’t” or capitalize ”you MUST”).
- Use delimiters for structuring (e.g., ###Context###, ...).
- Request output in JSON format.
- Repeat important phrases multiple times.
- Avoid long prompts (i.e., balance brevity and specificity).

Some Additional Resources:

- Prompt Engineering Guide by DAIR.AI
- Prompt engineering by OpenAI
- Brex’s Prompt Engineering Guide

Two Common Problems

Hallucination:

- LLMs may generate incorrect or fabricated information, potentially compromising the validity of results.

”Guardrails”:

- Models tuned to avoid controversy may refuse to annotate certain topics, posing challenges for tasks involving controversial content (e.g., radical political messages) or sensitive annotations (e.g., identifying an author’s gender).

Prompt Validation

Prompt Validation

Validate your prompt before undertaking any model validation steps, such as manually coding substantial amounts of data!

$\rightarrow$ Prompt validation safes ⏱️ and 💵

Prompt Validation

Asses Prompt Performance:

- Compare results against labeled data (i.e., small hand-coding sample) using classic performance metrics.
- Identify patterns of errors:
  - Does the output align with task expectations?
  - Are similar inputs yielding similar outputs?
  - Are the outputs unbiased and ethical?

Prompt Stability Analysis

LLMs can be very sensitive to trivial things like punctuation!

Calculate Stability Scores:

- Create several paraphrases of the prompt and run the analysis on a subset of the data.
- Estimate stability by comparing results using measures like Krippendorff's Alpha.

→ Barrie et al. (2024) have released a package for prompt stability scoring (Link to GitHub).

Model Evaluation

Model Evaluation

Must take place after prompt has been finalized!

How to evaluate LLM model outputs?

1. Label sufficient number of texts (min. 20–30 units per category)
    - Use larger validation datasets for imbalanced categories or high-stakes tasks
    - Power analysis recommended
2. Compare model results to manual labels using multiple performance metrics (e.g., Accuracy, F1 Score, Cohen's Kappa)
    - Validate across diverse subsets (e.g., languages, cultural contexts)
3. Examine and explain failures
    - Asses downstream risks for later analysis.

Applied Example 💻

Replication with LLMs

”Research tools are not merely passive instruments, but active participants in research procedures (Latour & Woolgar, 2013).”

Replication with LLMs

Exact replication is impossible with LLMs (same for hand-coding)!

Best Practice:

- Give motivation for model choice.
- Provide exact model version and parameter settings used.
- Save all versions and performance metrics of optimized prompts.
- Share random sample of input/output pairs as additional benchmarks.
- Store and share code/scripts for prompt automation.

References and More

Open Zotero Library: ”LLMs in Social Sciences”

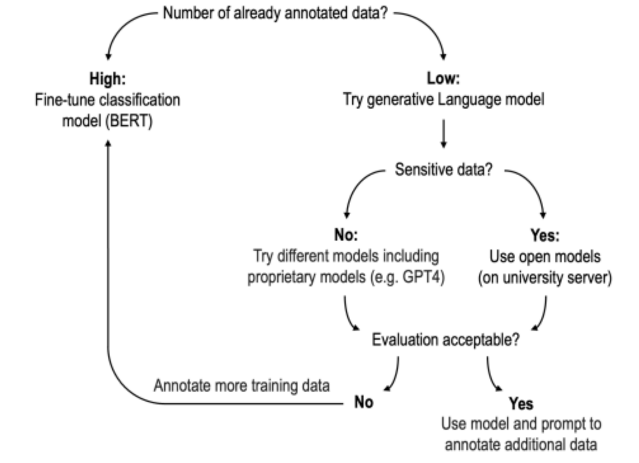
End of session 🎉

Thank you!

*This presentation was designed with the help of ChatGPT

Appendix

Model Selection: Simple Decision Tree



Number of already annotated data?
High:
Fine-tune classification model (BERT)
Low:
Try generative Language model
Sensitive data?
No:
Try different models including proprietary models (e.g. GPT4)
Yes:
Use open models (on university server)
Evaluation acceptable?
Annotate more training data
No
Yes
Use model and prompt to annotate additional data

Figure 1: Decision tree for the use of generative LLM for text annotation

Source: Weber and Reichardt (2024)

Advanced: Chain-of-Thought (CoT) Technique

What is CoT?

- A technique for improving reasoning and instruction-following in LLMs.
- Breaks tasks into simpler intermediate steps to mimic human problem-solving.

Why use CoT?

- Elicits step-by-step reasoning (Wei, Wang, et al., 2024).
- Enhances performance on complex tasks (Chung et al., 2024).
- Can be triggered with a prefix such as: *"Let's think step by step."*

Other Parameters

Top-P/K:

- Focuses on diversity by setting a probability- or top-K most probable tokens threshold for selecting tokens (requires temperature > 0)
  - Low Top-P/K (e.g., 0.2 to 0.4/ e.g., 5 or 10) ensures precision.
  - High Top-P/K promotes diverse outputs but may reduce accuracy.

Max Tokens:

- Sets a limit on the length of the model's response.
- Helps prevent overly lengthy or irrelevant outputs.

Best Practices:

- Start with low temperature for consistency.
- Adjust Top-P/K for controlled creativity.
- Use max tokens to prevent overly verbose responses.

*What exactly are Tokens? See Appendix

Advanced: Fine-Tuning

What is Fine-Tuning?

- Adapting a pre-trained model to specific tasks or domains using additional data.
- Requires labeled datasets and computational resources.

When to Consider Fine-Tuning:

- Highly domain-specific tasks (e.g., medical or legal text).
- Need for custom outputs not achievable with prompt engineering.

Advanced: Batching

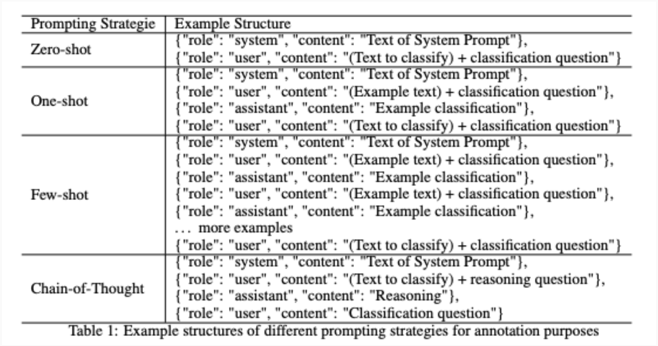
What is Batching?

- Grouping multiple input queries together for simultaneous processing.
- Reduces latency and improves throughput.
- Often used in high-volume workflows or API integrations.

Advantages of Batching:

- **Efficiency:** Minimizes overhead for repeated queries.
- **Cost-Effectiveness:** Lowers cost per request when using APIs.
- **Consistency:** Ensures uniform settings across similar tasks.

Example Structure for in-context learning techniques



Prompting Strategie	Example Structure
Zero-shot	{"role": "system", "content": "Text of System Prompt"}, {"role": "user", "content": "(Text to classify) + classification question"}
One-shot	{"role": "system", "content": "Text of System Prompt"}, {"role": "user", "content": "(Example text) + classification question"}, {"role": "assistant", "content": "Example classification"}, {"role": "user", "content": "(Text to classify) + classification question"}
Few-shot	{"role": "system", "content": "Text of System Prompt"}, {"role": "user", "content": "(Example text) + classification question"}, {"role": "assistant", "content": "Example classification"}, {"role": "user", "content": "(Example text) + classification question"}, {"role": "assistant", "content": "Example classification"}, . . . more examples {"role": "user", "content": "(Text to classify) + classification question"}
Chain-of-Thought	{"role": "system", "content": "Text of System Prompt"}, {"role": "user", "content": "(Text to classify) + reasoning question"}, {"role": "assistant", "content": "Reasoning"}, {"role": "user", "content": "Classification question"}

Table 1: Example structures of different prompting strategies for annotation purposes

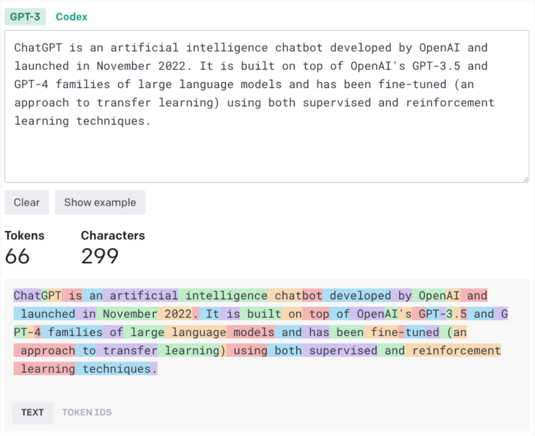
Source: Weber and Reichardt (2024)

**Chain-of-Thought*, see section ”Advanced”

Tokens

What are Tokens?

The atomic unit for language models is a token. Tokens resemble syllables, with 750 words per 1,000 tokens, and include punctuation, sentence boundaries, and document ends.



ChatGPT is an artificial intelligence chatbot developed by OpenAI and launched in November 2022. It is built on top of OpenAI's GPT-3.5 and GPT-4 families of large language models and has been fine-tuned (an approach to transfer learning) using both supervised and reinforcement learning techniques.

Tokens 66

Characters 299

```
ChatGPT is an artificial intelligence chatbot developed by OpenAI and
 launched in November 2022. It is built on top of OpenAI's GPT-3.5 and G
PT-4 families of large language models and has been fine-tuned (an
 approach to transfer learning) using both supervised and reinforcement
 learning techniques.
```