

Plug-and-Play NLP Pipelines With Zero-Shot Models

Raphael Mitsch | Climatiq/Mantis NLP

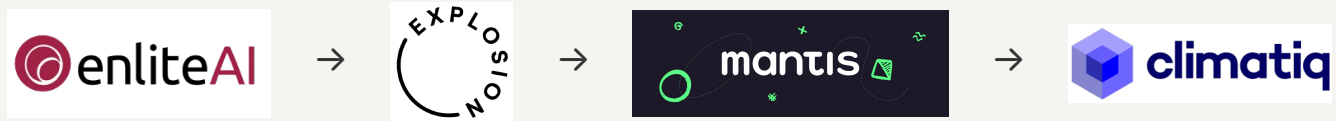
<https://www.linkedin.com/in/raphaelmitsch/> | <https://mantisnlp.com> | <https://climatiq.io>

Questions

<https://tinyurl.com/pydata-sieves>

About me

- Machine Learning Engineer in NLP



- Plenty of consulting/PoC/greenfield projects

Cold starts in NLP projects

- Pre-LLM
 - Collect, label training data
 - Engineer features
 - Train model
- LLM
 - Zero-/few-shots many vanilla problems 🎉
 - Faster idea-to-prototype turnaround

Cold starts in NLP projects

- Types of projects
 - Chatbots/copilots
 - RAG/retrieval
 - Document processing
- How to minimize time for **idea/specification → PoC?**

An NLP engineer's greenfield wishlist (1/2)

- No need for training data
- No reinventing the wheel for well-defined tasks
- Structured output
- Observability

An NLP engineer's greenfield wishlist (2/2)

- Model-agnostic
- Low-effort data handling
 - Input: parse all sorts of file types
 - Output: easy to train models with
- Handle arbitrary document lengths

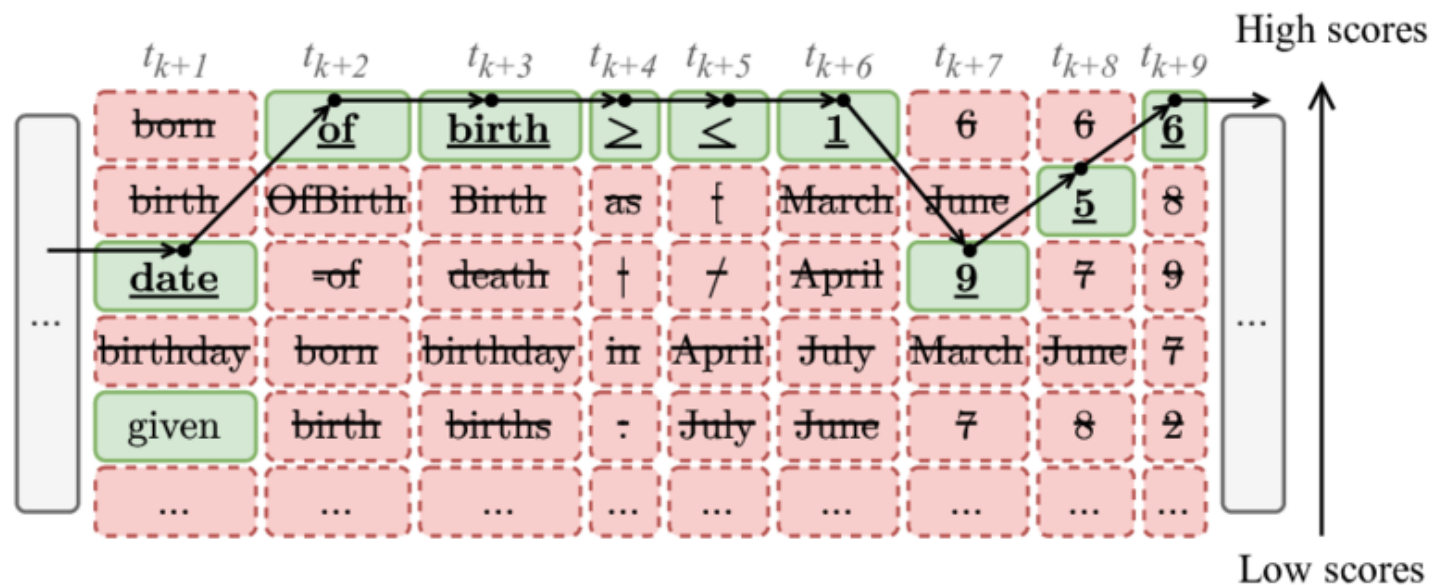
Aside: What is structured generation?

- **What:** restrict outputs to a schema/grammar (JSON, Pydantic, regex/EBNF)
- **Why:** valid, parseable outputs; reliable downstream processing
- **How:**
 - fine-tuning + graceful parsing; constrained token decoding
 - Dedicated libraries such as `outlines`
 - General-purpose LLM frameworks like `dspy` or `langchain`
 - Sometimes LLM vendor feature

When was Danny Boyle born?

Normal: <Danny Boyle> <born> <1960-09-15> .

Constrained: <Danny Boyle> <date of birth> <1956-10-20> .



[Pozzi, R. & Palmonari, M. & Coletta, A. & Bellomarini, L. & Lehmann, J. & Vahdati, S. \(2025\).](#)

[ReFactX: Scalable Reasoning with Reliable Facts via Constrained Generation. 10.48550/arXiv.2508.16983.](#)

Good news: all of this is approximately solved.

An NLP engineer's greenfield wishlist (1/2)

- No need for training data → **LLMs / zero-shot models**
- Structured output → **Constrained decoding**
- No reinventing the wheel for well-defined tasks → **Task library**
- Observability → **Modularity**



**NOOO YOU CAN'T JUST MIX
UP ALL THE STEPS OF YOUR TASK
AND ASK AN LLM TO DO IT ALL.
HOW WILL YOU EVER MAKE A RELIABLE
AND EXTENSIBLE SYSTEM THAT WAY?**

imgflip.com



HAHA LLM GO BRRR

It's never great to realize you're the guy on the left. But, here I am.

An NLP engineer's greenfield wishlist (2/2)

- Model-agnostic → **API design**
- Low-effort data handling
 - Input: parse all sorts of file types → **x-to-markdown parsers** (`docling` , `markdown` , ...)
 - Output: easy to train models with → 🙌 **datasets**
- Handle arbitrary document lengths → **Chunking + consolidation**

Bad news: no single tool like this exists yet.

What is **sieves** ?

- Toolkit to speed up prototyping document processing use cases
- Design focus:
 - document-centric
 - all outputs are structured outputs
 - modular execution via pipeline
 - small API surface

What is `sieves`?

- Design focus cont.:
 - opinionated choice of tools - no heavy lifting in `sieves`
 - model-agnostic, tool-agnostic
 - zero-/few-shot first
 - Batching by default
- Per [Anthropic terminology](#): AI workflow

What is it not?

- RAG, chatbot
- Per [Anthropic terminology](#): AI agent
- Replacement for structured generation libraries per se
- Fully-fledged workflow/orchestration system

```
from sieves import Engine, Doc
from sieves.tasks import Classification, QuestionAnswering

docs = [Doc(
    text="Special relativity applies to all physical phenomena in the "
        "absence of gravity. It was introduced by Albert Einstein in 1905."
)]

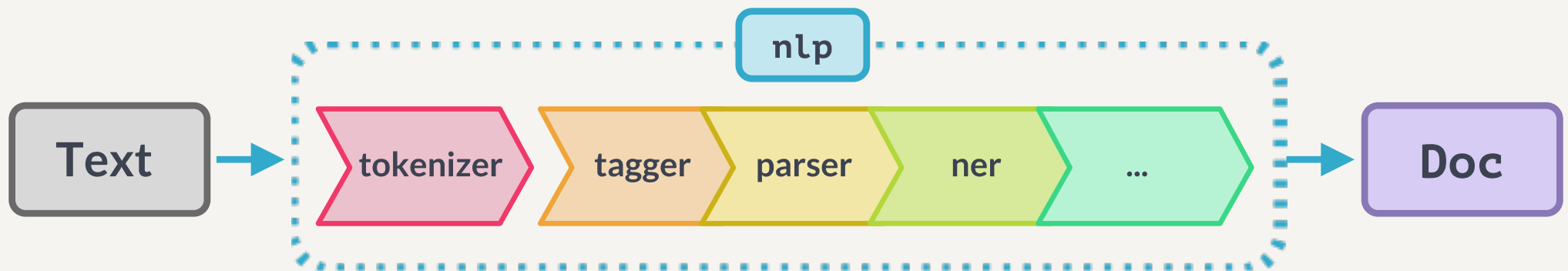
eng = Engine()
pipe = Classification(labels=["science", "politics"], engine=eng) + \
    QuestionAnswering(["Who introduced special relativity?"], engine=eng)

for doc in pipe(docs): print(doc.results)
# {
#     'Classification': [('science', 1.0), ('politics', 0.3)],
#     'QuestionAnswering': ['Einstein']
# }
```

Abstractions

1. `Doc` : document with text, metadata and results
2. `Pipeline` : pipeline orchestrating task execution
3. `Task` : any single unit of work
4. `Engine` : wraps underlying structured generation tool
 - `outlines` , `dspy` , `instructor` , `langchain` , `vllm` , `ollama` , `gliner` , `transformers`
5. `Bridge` : implements a `Task` with a specific `Engine`

spaCy





Task library

- Complex problems can be broken down into simpler tasks
- Simple tasks might be sufficiently similar for templated solutions
- Pre-implemented tasks
 - Classification, NER, sentiment analysis, summarization, information extraction, PII anonymization, translation, multi-QA
- Easy to extend

Demo: EU countries' stances on Chat Control

- Which country is pro/contra/undecided on chat control proposal?

1. Ingest news article

2. Extract countries' stances

3. Output

Demo: EU countries' stances on Chat Control

<https://fightchatcontrol.eu/>

Other QoL features




- **Persistence**: (de-)serialization of pipelines and docs
- **Distillation**: integrated model training on pipeline output¹
- **Caching**: pipeline caches results so that identical docs are processed exactly once

¹ Currently only for classification.

When (not) to use **sieves**

Question	✓	?	✗
Use case	Document processing	RAG	Chatbot
Modularization	Important	Don't care	Don't want overhead
Structured output	Important	Maybe	Don't need
Utilities handling	Built-in	Don't care	Minimize dependencies

When (not) to use **sieves**

Question			
Existing struct. gen. stack	None	Some	A lot
Project stage	Greenfield	Bluefield	Brownfield
Priority	Shipping speed	In between	Optimization
Project complexity	M	L	S

QA

<https://tinyurl.com/pydata-sieves>

Thanks! / Dank u wel!

- <https://www.linkedin.com/in/raphaelmitsch/>
- <https://pypi.org/project/sieves/> (currently v0.13.0)
- <https://sieves.ai> | <https://github.com/MantisAI/sieves>
- <https://mantisnlp.com>
- <https://climatiq.io>