LLMOps: Production-Ready GenAI Systems

Xebia

getindata
Part of Xebia

by Marek Wiewiórka, Phd

getindata
Part of Xebia

Xebia

Marek Wiewiórka

PhD | Chief Data Architect at GetInData

getindata
Part of Xebia

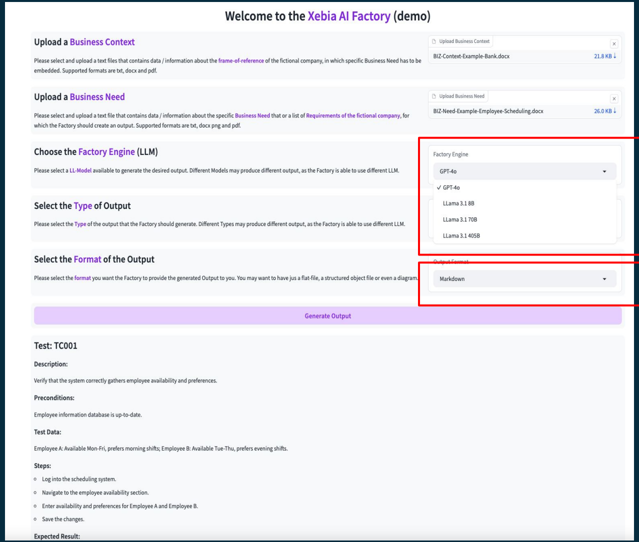
Xebia

Agenda disclaimer

- Not a comprehensive overview of Large Language Model Operations (*LLMOps*)
- Not (primarily) about tools or platforms
- Not about ready-to-implement processes
- ...but about (a few) **concepts that may help to drive *success* of GenAI projects**
- **2 different use cases** for inspiration

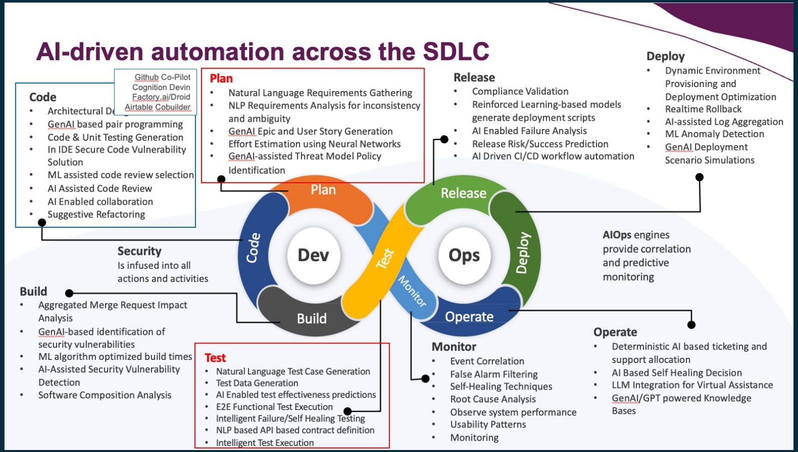
A starting point...

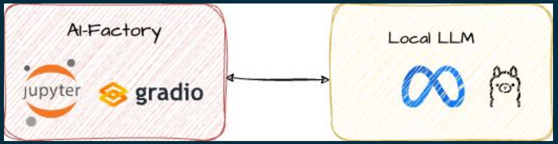
getindata
Part of Xebia
Xebia



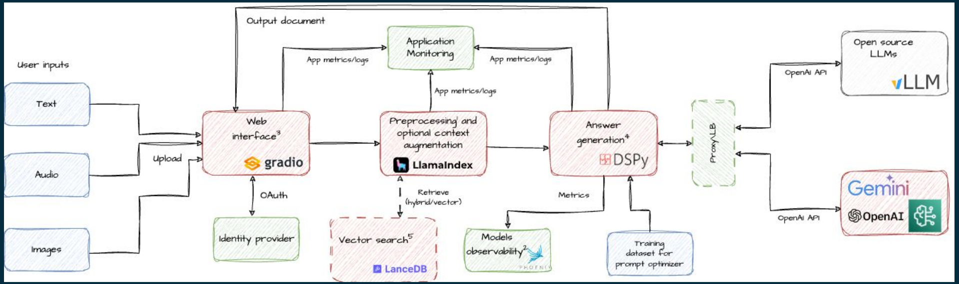
model neutrality

structured output





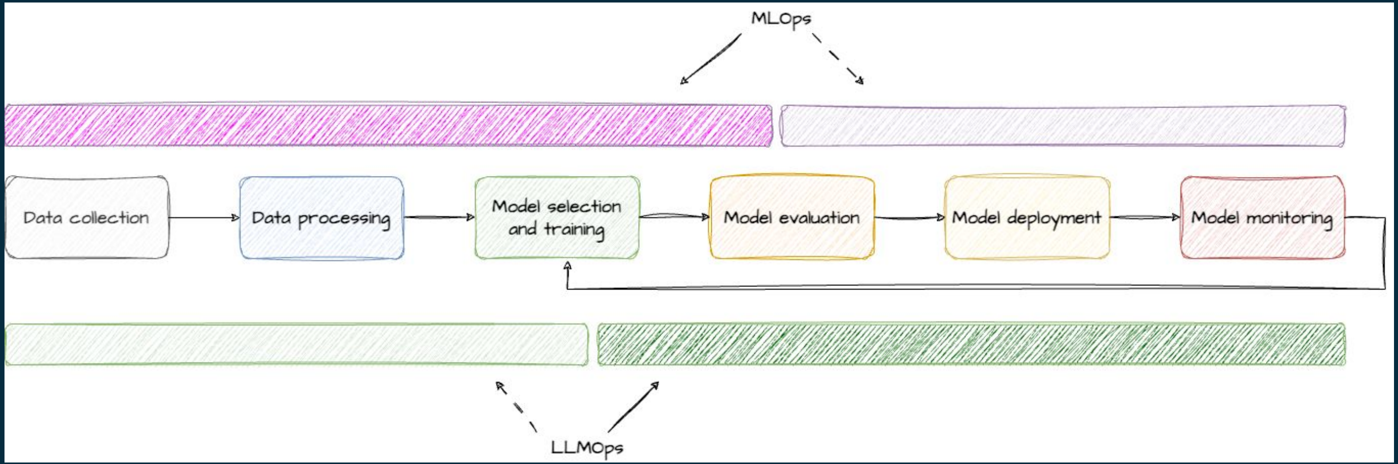
?



Demo **LLMOps framework!** Production-grade system

MLOps => LLMOps

- Translating ML models into reliable, cost-efficient systems and managing their lifecycle
- MLOps and LLMOps have the same goals and principles but differ in **focus**



getindata
Part of Xebia

Xebia

LLMOps challenges in the enterprise context

- **LLMs *churn*** - new, more capable models are released frequently
- **Multi model strategies** - optimizing for cost, latency, quality
- **LLM specialization** - one size does not fit all
- On-premise vs managed deployments
- Output of LLM is by nature **non-deterministic**
- Security and data protection
- … there is not such a thing as LLM backward compatibility out of the box

Reproducible outputs Beta

Chat Completions are non-deterministic by default (which means model outputs may differ from request to request). That being said, we offer some control towards deterministic outputs by giving you access to the seed parameter and the system_fingerprint response field.

To receive (mostly) deterministic outputs across API calls, you can:

- Set the seed parameter to any integer of your choice and use the same value across requests you'd like deterministic outputs for.
- Ensure all other parameters (like `prompt` or `temperature`) are the exact same across requests.

Sometimes, determinism may be impacted due to necessary changes OpenAI makes to model configurations on our end. To help you keep track of these changes, we expose the system_fingerprint field. If this value is different, you may see different outputs due to changes we've made on our systems.

Complexity requires automation

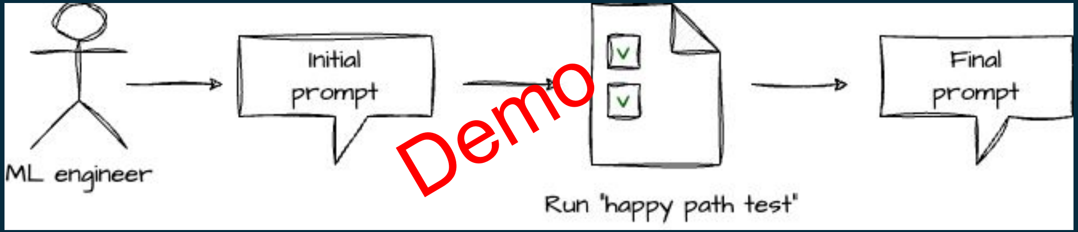
- Prompt optimization
- Controlling LLM Output
- LLM testing and evaluation
- Costs optimization

"It's very easy to make a prototype," Henley, who studied how copilots are created in his role at Microsoft, says. "It's very hard to production-ize it." Prompt engineering—as it exists today—seems like a big part of building a prototype, Henley says, but many other considerations come into play when you're making a commercial-grade product.

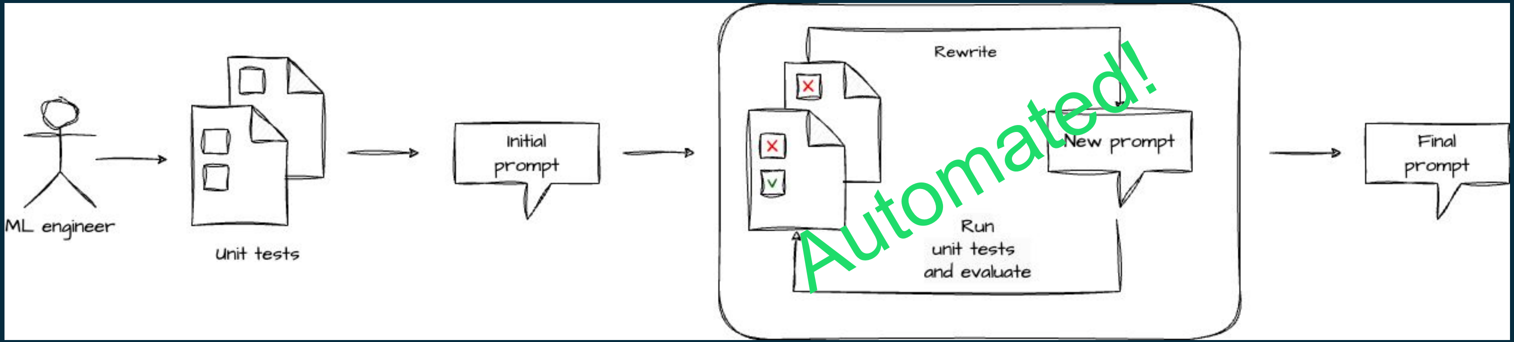
The challenges of making a commercial product include ensuring reliability—for example, failing gracefully when the model goes offline; adapting the model's output to the appropriate format, because many use cases require outputs other than text; testing to make sure the AI assistant won't do something harmful in even a small number of cases; and ensuring safety, privacy, and compliance. Testing and compliance are particularly difficult, Henley says, because traditional software-development testing strategies are maladapted for nondeterministic LLMs.

Genkina, Dina. "Don't Start a Career as an AI Prompt Engineer AI will Take Your Job." *IEEE Spectrum* 61.5 (2024): 30-34.

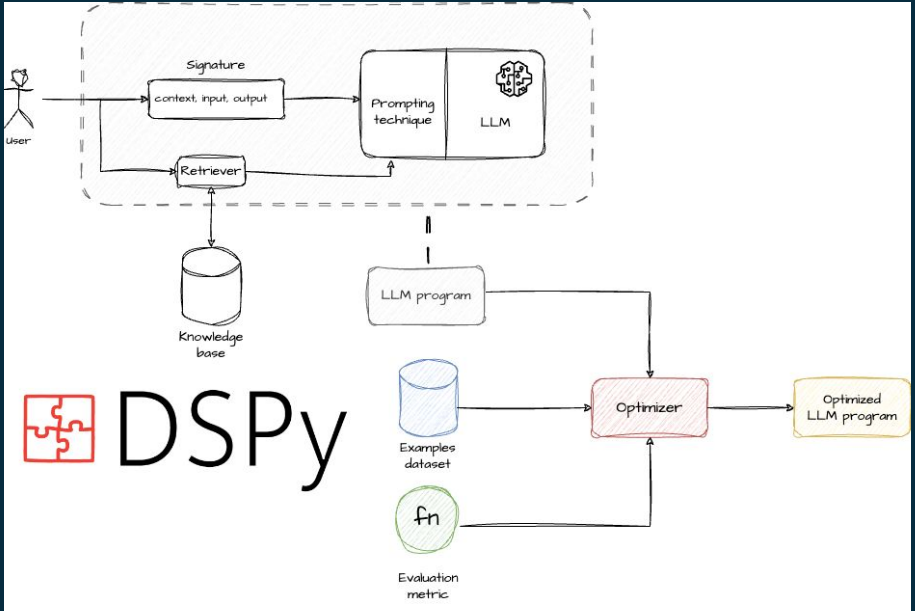
Prompt engineering process



- Similar prompts => <u>different</u> output
- Best prompt specific to a <u>model</u>
- Both instructions and examples (in-context learning) can have great impact on <u>output</u>
- An <u>error-prone</u> and <u>tedious</u> process



Prompt engineering => optimization task



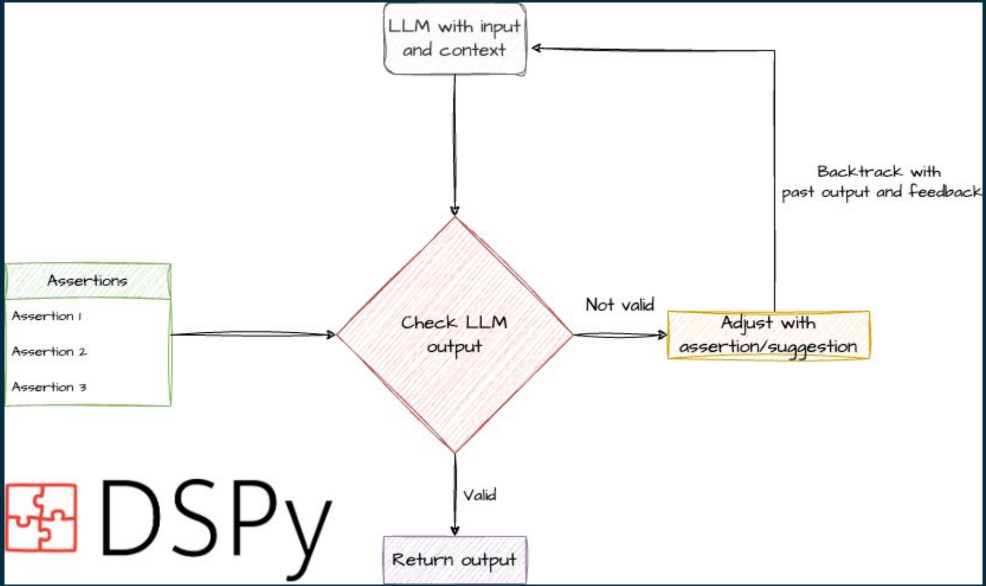
- Different optimizing strategies for both selecting/bootstrapping examples, instructions or models/programs **Ensemble**
- Metric can be an arbitrary function even LLM-based (**LLM-as-a-judge**)
- Can be a **student-teacher** model setup

Zhang, Tuo, Jinyue Yuan, and Salman Avestimehr. "Revisiting OPRO: The Limitations of Small-Scale LLMs as Optimizers." *arXiv preprint arXiv:2405.10276* (2024).

Khattab, Omar, et al. "Dspy: Compiling declarative language model calls into self-improving pipelines." *arXiv preprint arXiv:2310.03714* (2023).

Opsahl-Ong, Krista, et al. "Optimizing Instructions and Demonstrations for Multi-Stage Language Model Programs." *arXiv preprint arXiv:2406.11695* (2024).

Automated strategies for controlling LLM outputs



- **Assertions** - general purpose mechanism for guardrailing LLM output
- **Typed Predictions** - specialized for enforcing specific schema using *Pydantic* models for LLM output
- Both can be used in the prompt optimization process

Singhvi, Arnav, et al. "DSPy Assertions: Computational Constraints for Self-Refining Language Model Pipelines." *arXiv preprint arXiv:2312.13382* (2023).

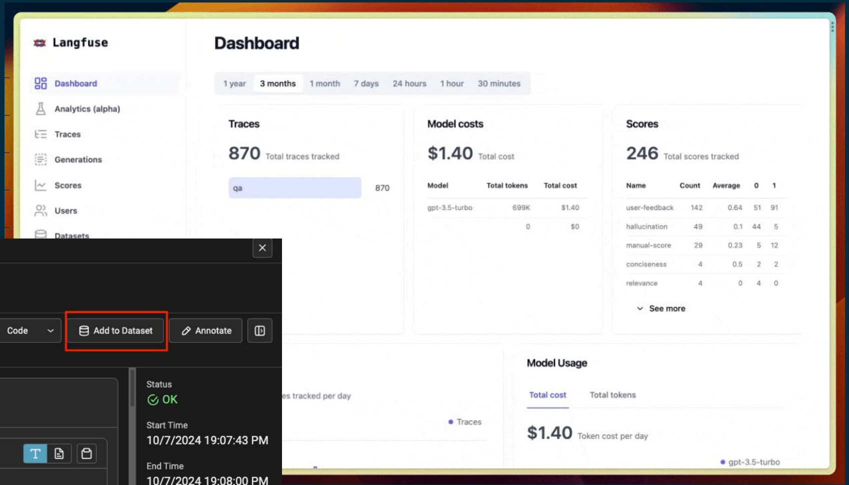
LLM Evaluation

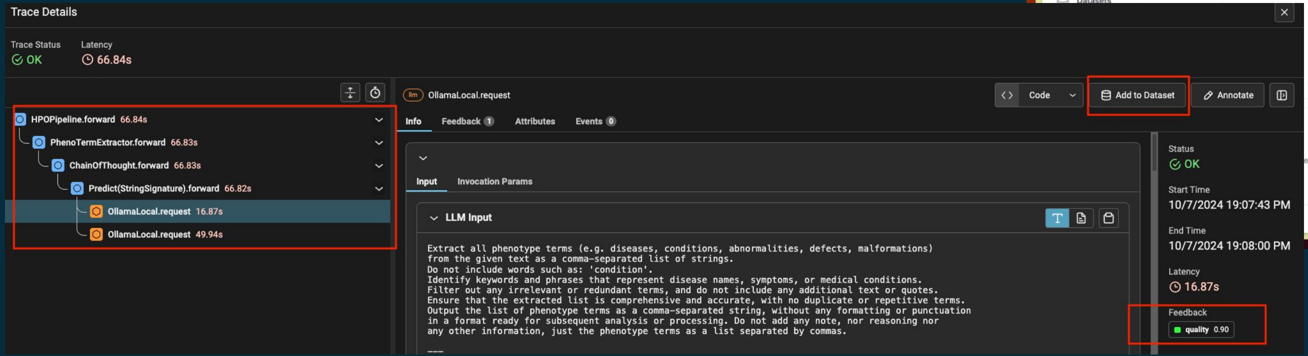
- Absolutely crucial when building a <u>reliable</u> LLM-system
- Depending on the problem can be statistical (e.g. precision, recall, F1) or model-based (LLM-as-a-judge) in more generic cases
- Problem of aligning LLM evaluation with human preferences
  - G-Eval, Prometheus and Evalgen
- Human annotated LLM outputs for calibration
- LLM-assisted criteria and assertion generation

Shankar, Shreya, et al. "Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences." *arXiv preprint arXiv:2404.12272* (2024).

Complexity requires observability

- Open-source tools such as LangFuse, Phoenix and LangSmith emerge, putting high emphasis on LLM observability, including:
  - program metrics, e.g. latency, tokens, costs
  - evaluations scores (optimization process)
  - traces
  - user feedback (annotations)
- user feedback => new datasets



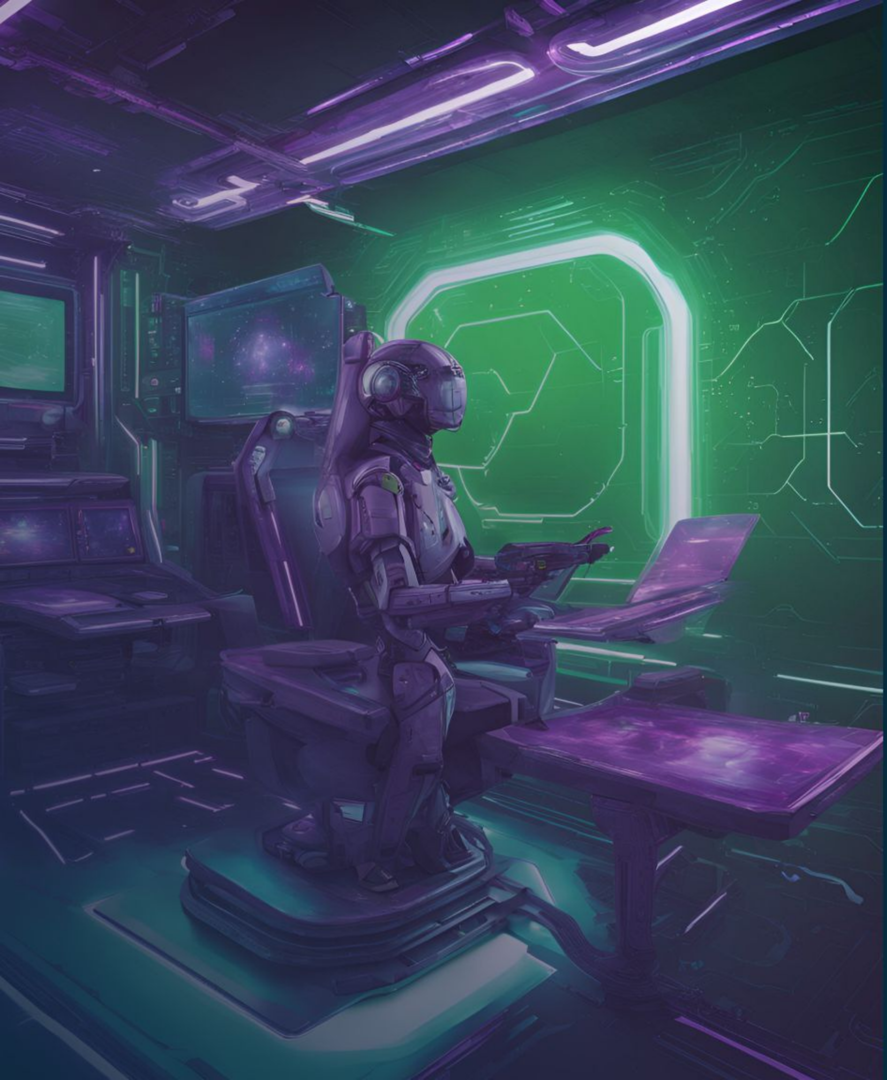


Costs optimization

- Different strategies (from most to least complex)
  - hardware optimization
  - LLM optimization (e.g. quantization, scaling down parameters, fine-tuning)
  - LLM routing
  - LLM ensemble optimization, collective wisdom - Mixture-of-Agents
  - prompt optimization
- ... but in order to try to do it you need to have:
  - portable LLM pipelines
  - evaluation datasets and metric functions
  - observability platform

Wang, Junlin, et al. "Mixture-of-Agents Enhances Large Language Model Capabilities." *arXiv preprint arXiv:2406.04692* (2024).

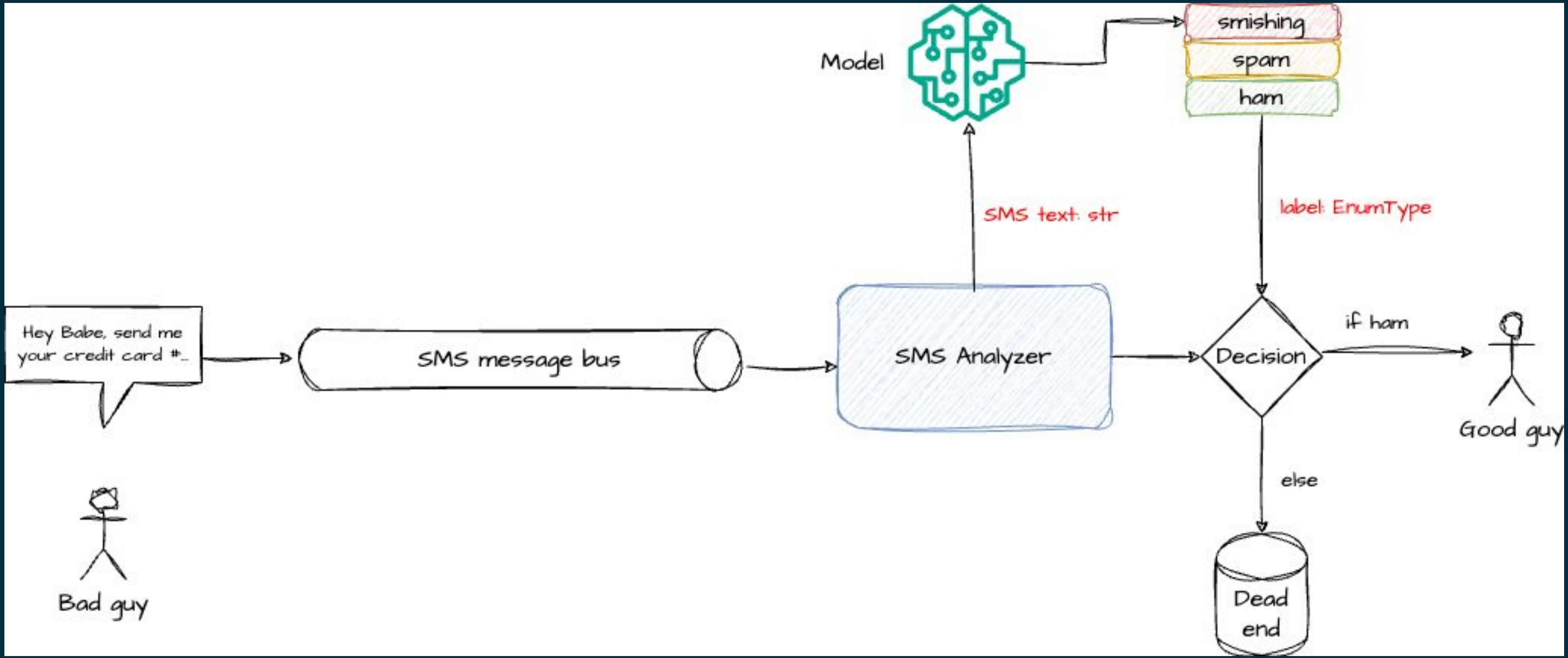
Ong, Isaac, et al. "Routellm: Learning to route llms with preference data." *arXiv preprint arXiv:2406.18665* (2024).



Demo scenario

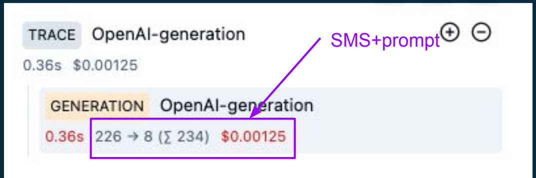
https://github.com/mwiewior/llmops-webinar

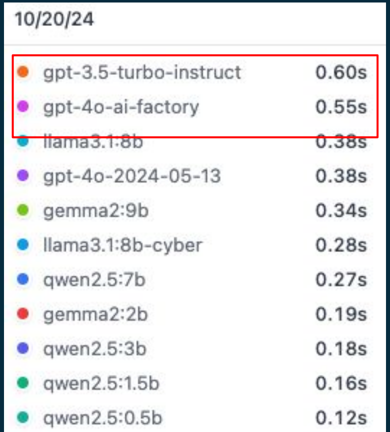
Scenario: SMS Phishing Detection System

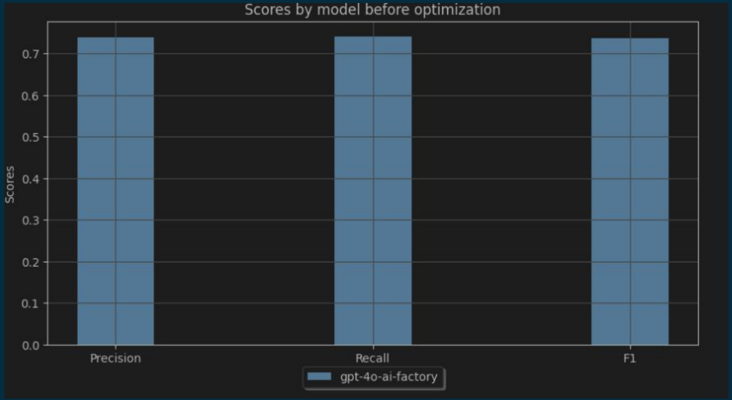


Demo first!

- demo with GPT4-o successful…but your boss have 3 concerns:
  - costs
  - latency
  - security







Idea: Let's productionize it with a *<u>much smaller open-source LLM</u>* that can be hosted *<u>locally</u>*.

getindata
Part of Xebia
Xebia

Scenario: SMS PHISHING - the LLMOps way!

- prepare a train-eval-test dataset
- define your metric functions
- make the output structured
- optimize-evaluate-observe loop
- version the LLM-program
- use GitOps for deployment

Mendeley Data

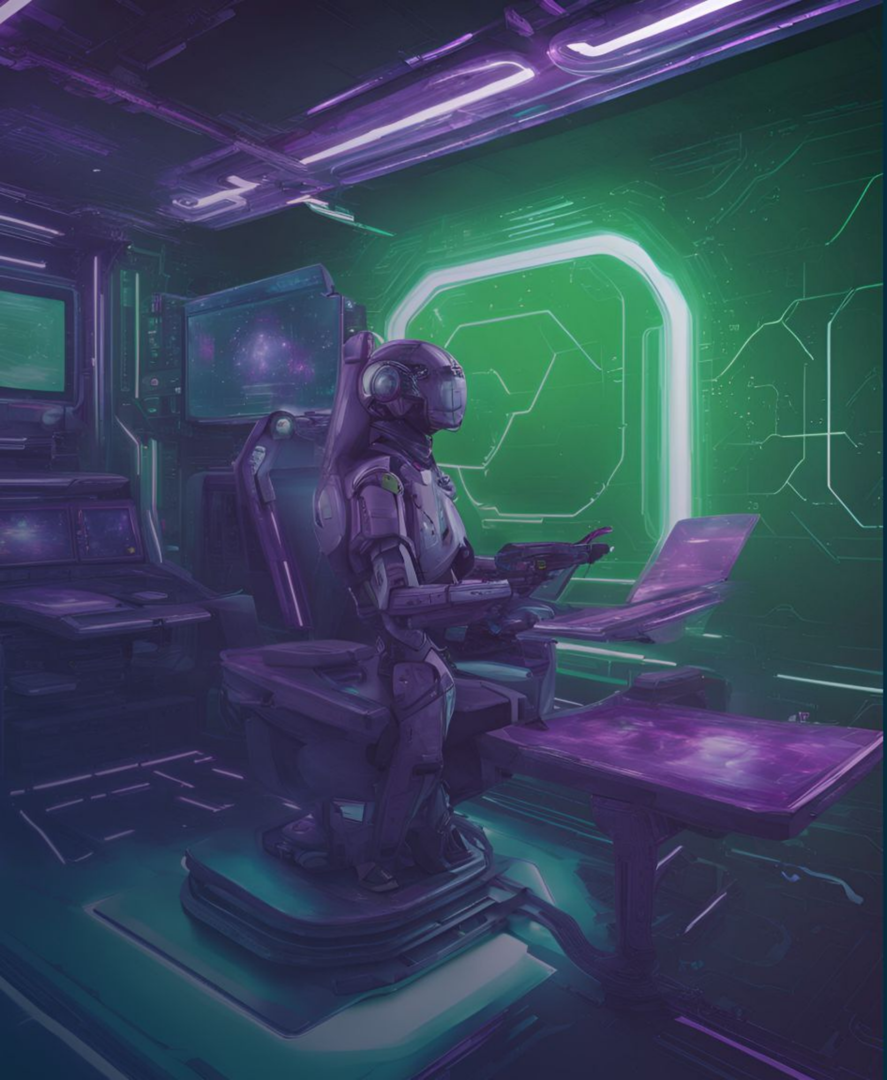
SMS PHISHING DATASET FOR MACHINE LEARNING AND PATTERN RECOGNITION

Published: 20 June 2022 | Version 1 | DOI: 10.17632/f45bkkt8pr.1
Contributors: sandhya mishra, Devpriya Soni

Description

The dataset is a set of labelled text messages that have been collected for SMS Phishing research. It has 5971 text messages labeled as Legitimate (Ham) or Spam or Smishing. It includes 489 spam messages, 638 smishing messages, and 4844 ham messages. This dataset contains raw message content that can be used as labelled data in Deep Learning or for extracting further attributes. The dataset contains extracted attributes from malicious messages that can be used for Classification of messages as malicious or legitimate. This dataset also includes python code that are used for extracting attributes. The data has been collected by converting the images obtained from the Internet to text using Python code. Attributes have been selected based on their relevance. The details of dataset attributes are given below:

LABEL- Classification label categorizing the message as ham, spam, or Smishing
TEXT- The raw content of the message.
URL- Gives out whether the message contains a URL or not.
EMAIL- Gives out whether the message contains an email id or not.
PHONE - Gives out whether the message contains a phone number or not.
Python code for extraction of the above listed dataset attributes is attached. The snapshot of this dataset is also attached. Frequency chart of the attributes are also attached.



Notebook time!

What you've learned today

- use LLM for a classification problem
- compare SLMs vs SOTA GPT-4o
- show how to use ***DSPy*** for automatic prompt optimization, structured output and ***Langfuse*** for observability
- analyze the evaluation results
- optimize costs and boost LLM performance with model fine-tuning and automatic prompt optimization

GetInData | Part of Xebia

getindata
Part of Xebia
Xebia

Experts in Data, Cloud, Analytics and ML/AI, and GenAI solutions

Experience in: media, e-commerce, retail, fintech, banking & telco

Solution Areas

- LLMOps/MLOps Platforms
- Data, AI & Cloud Engineering
- Stream Processing & Real-time Analytics
- Data Platforms Modernization & Migration

Selected technologies

Apache Airflow, Flink, dbt, Kedro, Looker, Airbyte, mlflow

Selected Customers

ING, PLAY, iZettle, Kcell, truecaller, VEOLIA, Acast

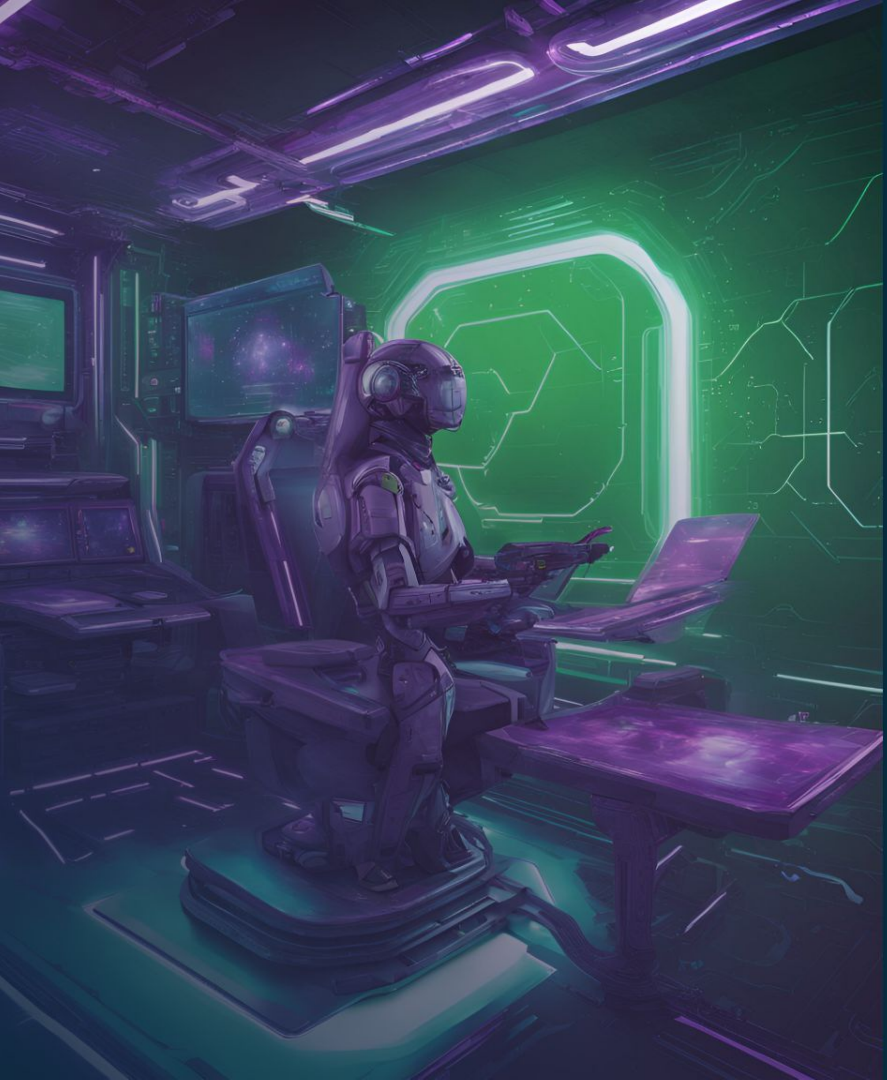
getindata
Part of Xebia

Xebia

Want to talk about DATA & AI with US?

hello@getindata.com

SCHEDULE A CONSULTATION

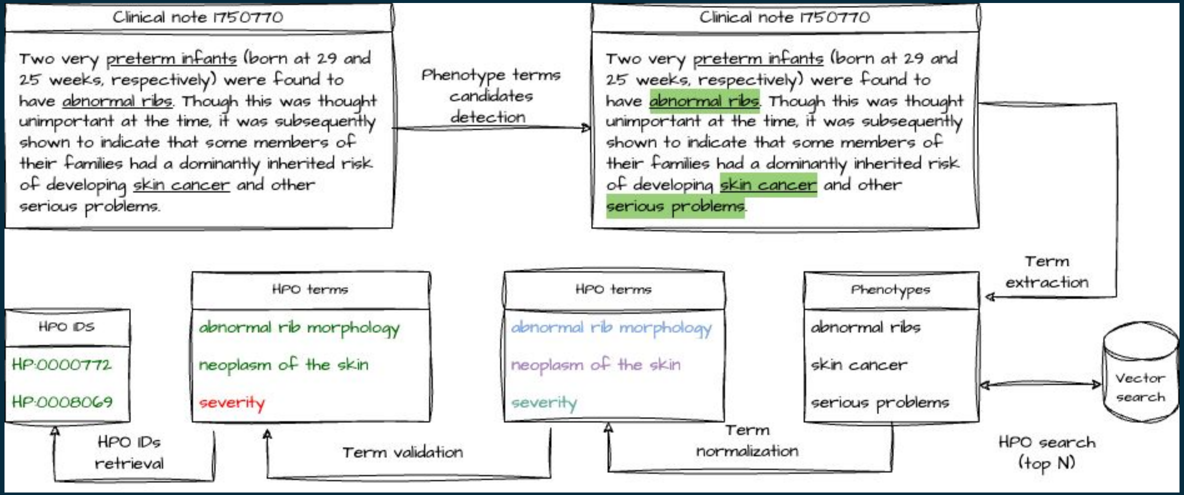


getindata
Part of Xebia
Xebia

Encore

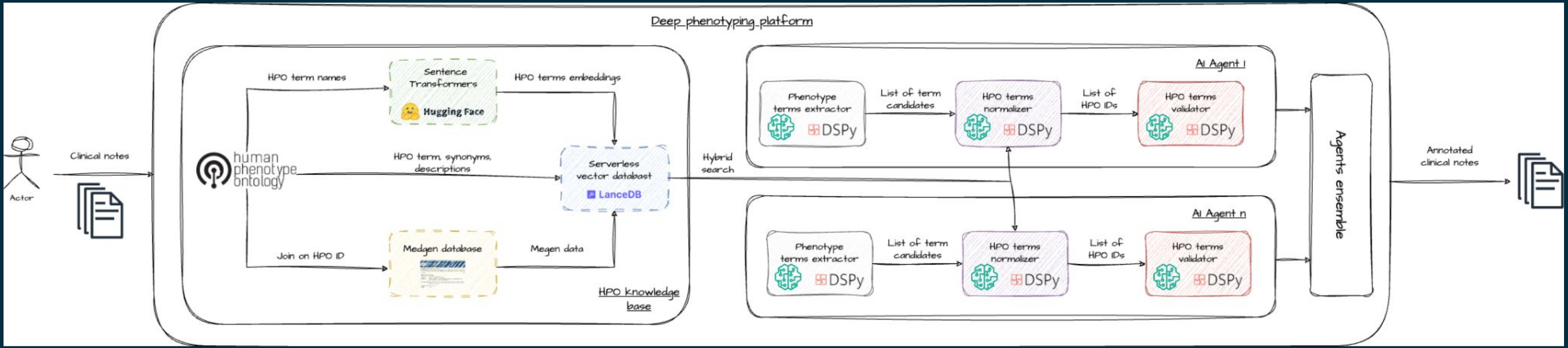
PhenoAgent - use case

- ***Deep phenotyping*** refers to the comprehensive and detailed analysis of phenotypic traits in organisms
- Two-step procedure involving:
  - concept recognition (finding phenotypic information in the unstructured text) and
  - concept normalization (mapping recognized concepts to the standardized Human Phenotype Ontology (HPO) identifiers)

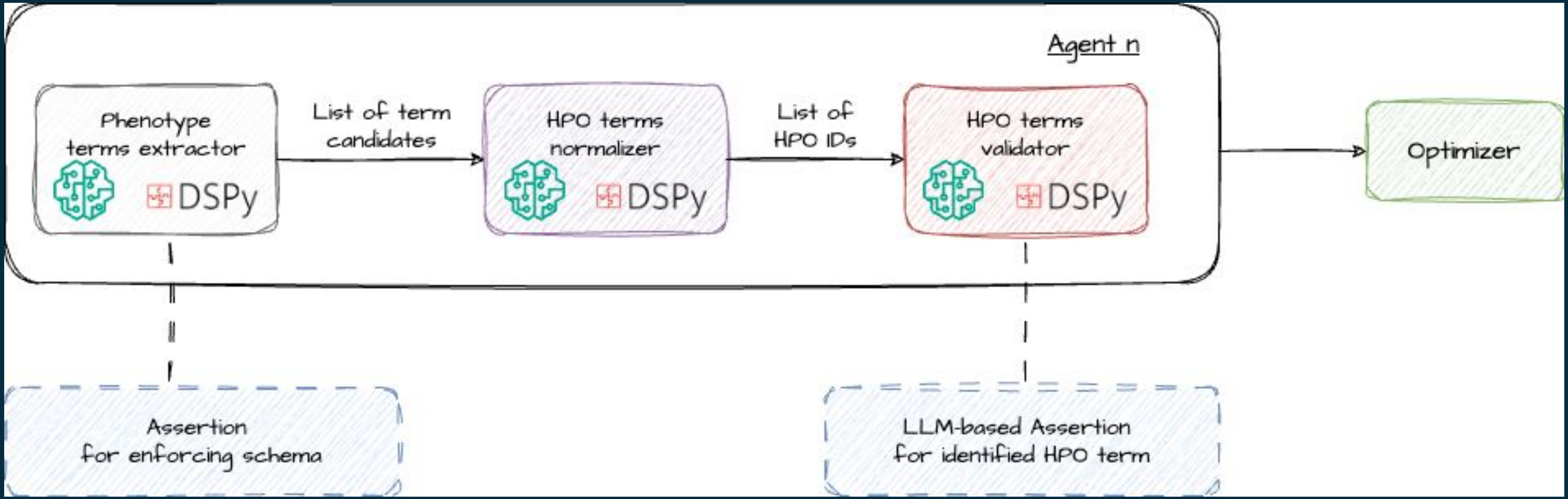


PhenoAgent - architecture

**PhenoAgent - **first LLM-based tool for an automatic HPO terms annotation powered by <u>RAG</u> and small <u>LLMs ensemble</u> architecture



PhenoAgent - deep dive



PhenoAgent - results and conclusions

- Optimized ensemble of LLM programs of small (and quantized) LLMs can easily **outperform SOTA models** (i.e. Llama-3.1-405/70B)
- RAG architecture can **outperform fine-tuned models** of comparable size
- Using concepts like Assertions and automated prompt optimization help to deliver portable LLM-pipelines
- Using model ensemble and prompt optimization can reduce costs of infrastructure

Tool	Model	Precision	Recall	F1
PhenoGPT	Llama2-7B	0.3136	0.2805	0.2961
PhenoAgent	Llama3-8B	0.5699	0.5511	0.5603
PhenoAgent-MoA-8-3	MoA-8	0.6275	0.6241	0.6258
PhenoAgent-Llama-405	Llama3.1-405B	0.6248	0.5616	0.5915