Aalto University
School of Science

GenAI/LLMs and Big Data Platforms

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, https://rdsea.github.io

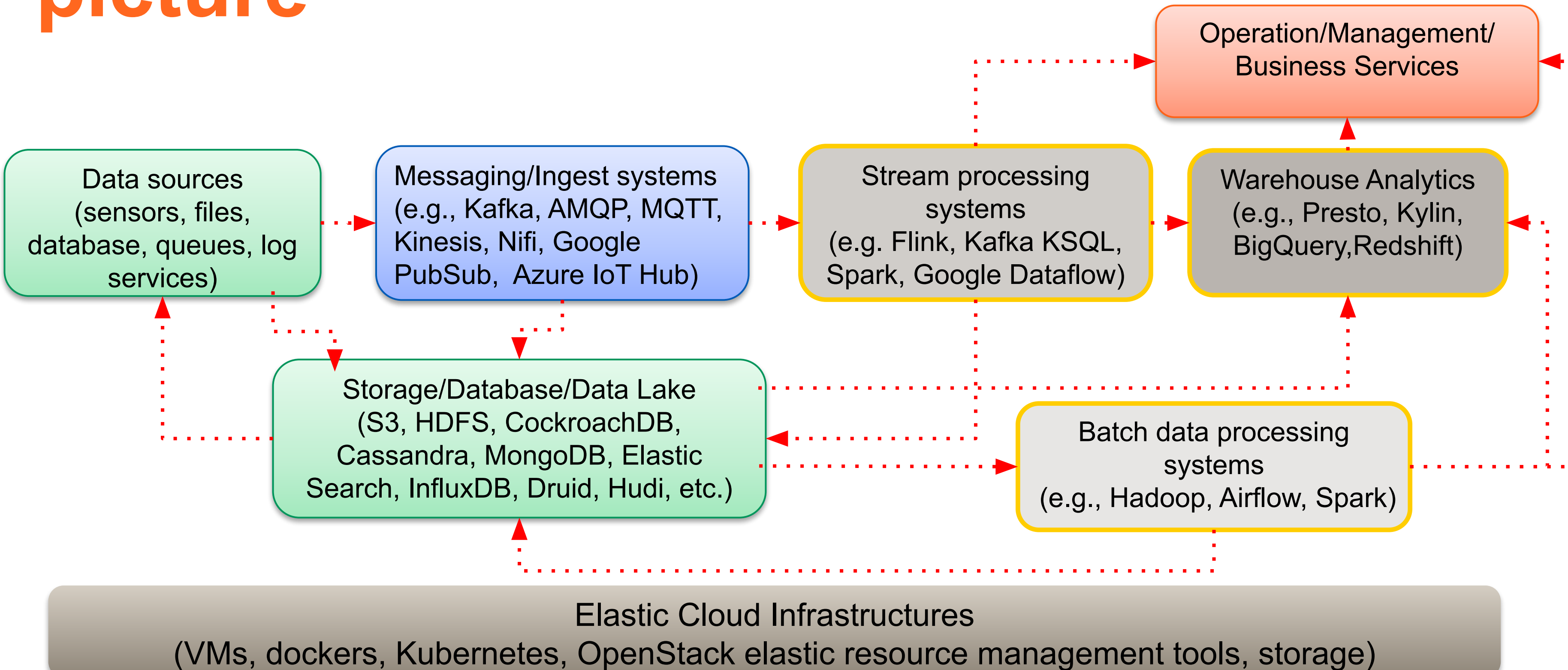
Learning objectives

- **Understand the role of GenAI/LLMs as services for big data platforms**
- **Able to support the development of GenAI/LLMs using techniques and methods studied from big data platforms**

Up to now: what you have learned is very much about the past and the present

- Big data platforms are complex
- We have studied basic, foundational techniques and models
- We have practiced with existing platforms, and developed our own use cases/solutions

Big data at large-scale: the big picture



Extreme big data is still very relevant and hard subject!

“extreme data mining”

“volume, speed, variety, complexity/diversity/mul tilinguality of data”

“sparse/missing/insuffici ent data/extreme variations”

“big data, AI, IoT, HPC, edge/fog/cloud computing”

Topic description

ExpectedOutcome:

Proposal results are expected to contribute to the following expected outcomes:

- provide better technologies, tools and solutions for data mining (searching and processing) of large, constantly growing amounts and varieties of data, and/or extremely sparse/dispersed/heterogeneous/multilingual data (stored centrally or in distributed/decentralized systems), in particular IoT, industrial, business, administrative, environmental, scientific or societal data.

Scope:

The actions under this topic are expected to provide ground-breaking advances in the performance, speed and/or accuracy as well as usefulness of data discovery, collection, mining, filtering and processing in view of coping with “extreme data”: (defined as data that exhibits one or more of the following characteristics, to an extent that makes current technologies fail: increasing volume, speed, variety; complexity/diversity/multilinguality of data; the dispersed data sources; sparse/missing /insufficient data/extreme variations in values). The technologies and solutions are expected to discover and distil meaningful, reliable and useful data from heterogeneous and dispersed/scarce sources and deliver it to the requesting application/user with minimal delay and in the appropriate format. In particular, the advances should enable the development of trustworthy, accurate, green and fair AI systems where quality of data is as important as quantity and/or support industrial distributed decision-making tasks at appropriate level in the computing continuum (edge/fog/cloud). Insofar the results are intended for human use, the design of these tools should take into account the relevant human aspects and interactions with users.

The actions should address the integration of relevant technologies (e.g. big data, AI, IoT, HPC, edge/fog/cloud computing, language technologies, cybersecurity, telecommunications, autonomous systems etc.) as a means towards achieving the goals, and foster links to the respective research, industrial and user/innovator communities (e.g. AI4EU, digital innovation hubs). The use of European data sources (such as Copernicus, Galileo/EGNOS for satellite data) is encouraged in the use cases, where appropriate.

Source:
https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/horizon-cl4-2022-data-01-05

Trends

- **IoT and real time data integration for real time analytics**
- **Data products**
  - Internal Developer Platforms (IDPs) for Big Data/Data intensive applications and systems
  - emerging concepts and platforms for data mesh, lakehouse, etc.
- **Big data platforms as backbone for AI/ML/LLMs**
  - in addition to existing applications and domains, building LLMs requires strong support from big data techniques, models and platforms
- **AI/ML/LLMs services for big data platforms**
  - explore the benefits of ML/LLMs for the development and use cases in big data platforms

GenAI/LLMs

- **GenAI/Large Language Models (LLMs)**
  - using generative models to generate text, code, images, etc.
  - generative models learn patterns and build their capabilities based on data
  - LLMs: foundational models trained with a lot of data, able to handle and generate natural languages and other contents
- **Relation to big data platforms**
  - integrating LLMs services for supporting analytics, design and development
  - leveraging our expertise in big data platforms for building GenAI/LLMs
    - *data pipeline, data services, data transformation, and data quality/governance*

Aalto University
School of Science

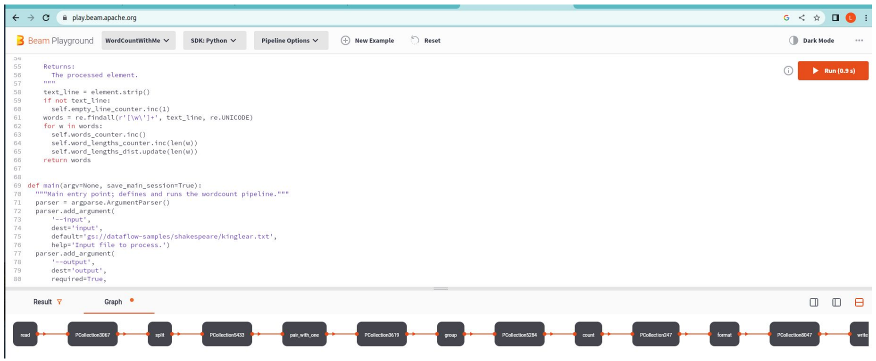
GenAI/LLMs for low code in big data programming

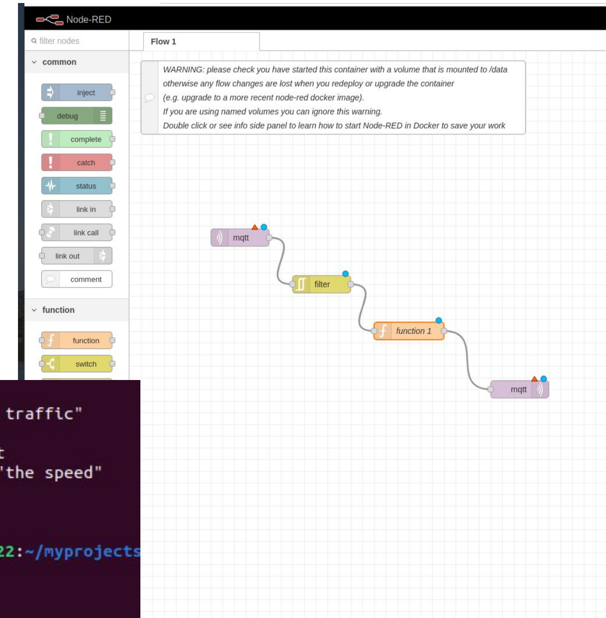
What have we had? look back your code and examples in the tutorials

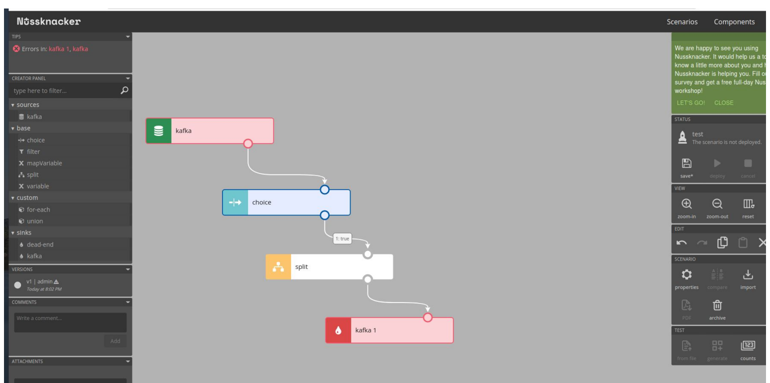
- **Your way:**
  - you have done a lot of ETL, data analytics, etc.
- **The Nifi way:**
  - defining flows using drag and drop features
- **The Airflow-alike way:**
  - programing workflows with APIs and annotations
- **The Flink/Spark way:**
  - low-level data processing API
- **Multiple runtime systems and microservices:**
  - utilizing containers (e.g., Airbyte), mapping to different underlying runtimes (e.g., Apache Beam)

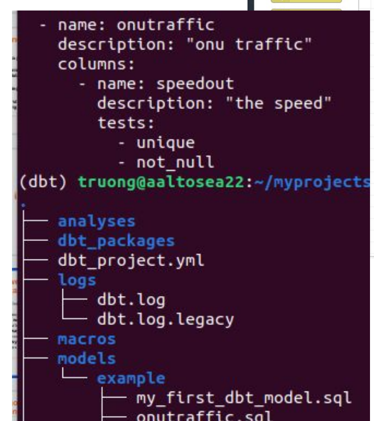
Some snapshots

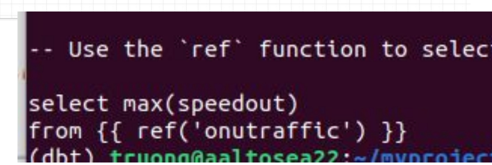
(a snapshot from Beam)











Difficulties

- **Nice UI, simple drag and drop, and easy composition**
  - difficult to solve complex processing and quality control requirements
- **Programming solutions**
  - powerful, but low-level programming, mostly difficult for users who do not know programming languages/technologies well
- **Handling streaming data is still hard**
- **How could low-code and AI/LLMs techniques help to generate powerful transformation code?**
  - lowering cognitive load for users and empowering generalists (who can work on code, data and ML models)

GenAI/LLMs as a solution?

- **Can GenAI/LLMs reduce programming tasks?**
- **Can GenAI/LLMs help for speeding up designs of big data platforms?**
- **Can GenAI/LLMs do the analytics of data by invoking suitable algorithms?**
- **etc.**

Write the question seeking for a solution, but does the answer help much?

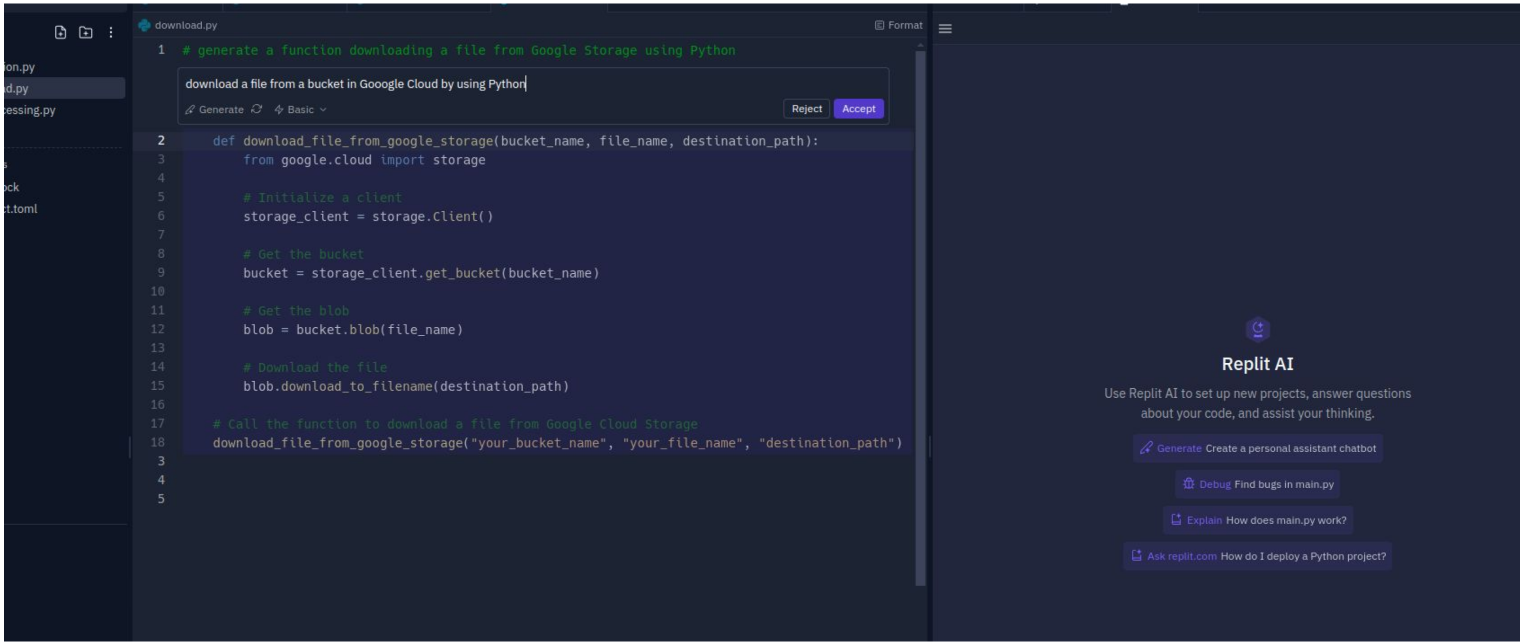
TR

pls tell me how can I determine which types of data and which types of algorithms that can be used to detect anomalies in a mobile network for 3G data?

Detecting anomalies in a mobile network, specifically for 3G data, requires a combination of domain knowledge and data analysis techniques. To determine which types of data and algorithms to use for anomaly detection in a 3G mobile network, consider the following steps:

1. **Understand the Data**:
   - First, gain a deep understanding of the data you have access to. This could include various metrics and logs related to network performance, traffic, and user behavior.
2. **Data Preprocessing**:
   - Clean and preprocess the data to handle missing values, outliers, and inconsistencies. Data preprocessing may involve imputing missing values, normalizing data, and transforming it into a suitable format.
3. **Select Relevant Features**:
   - Choose the most relevant features or metrics that are likely to be indicative of anomalies in a 3G mobile network. Features may include signal strength, latency, packet loss, traffic volume, and more.
4. **Define Anomalies**:
   - Determine what constitutes an anomaly in your context. Anomalies in a mobile network might include unusual spikes in traffic, dropped calls, sudden changes in signal quality, or unexpected patterns in user behavior.
5. **Choose Anomaly Detection Algorithms**:
   - Select appropriate anomaly detection algorithms based on the nature of your data. Some common algorithms for anomaly detection include:

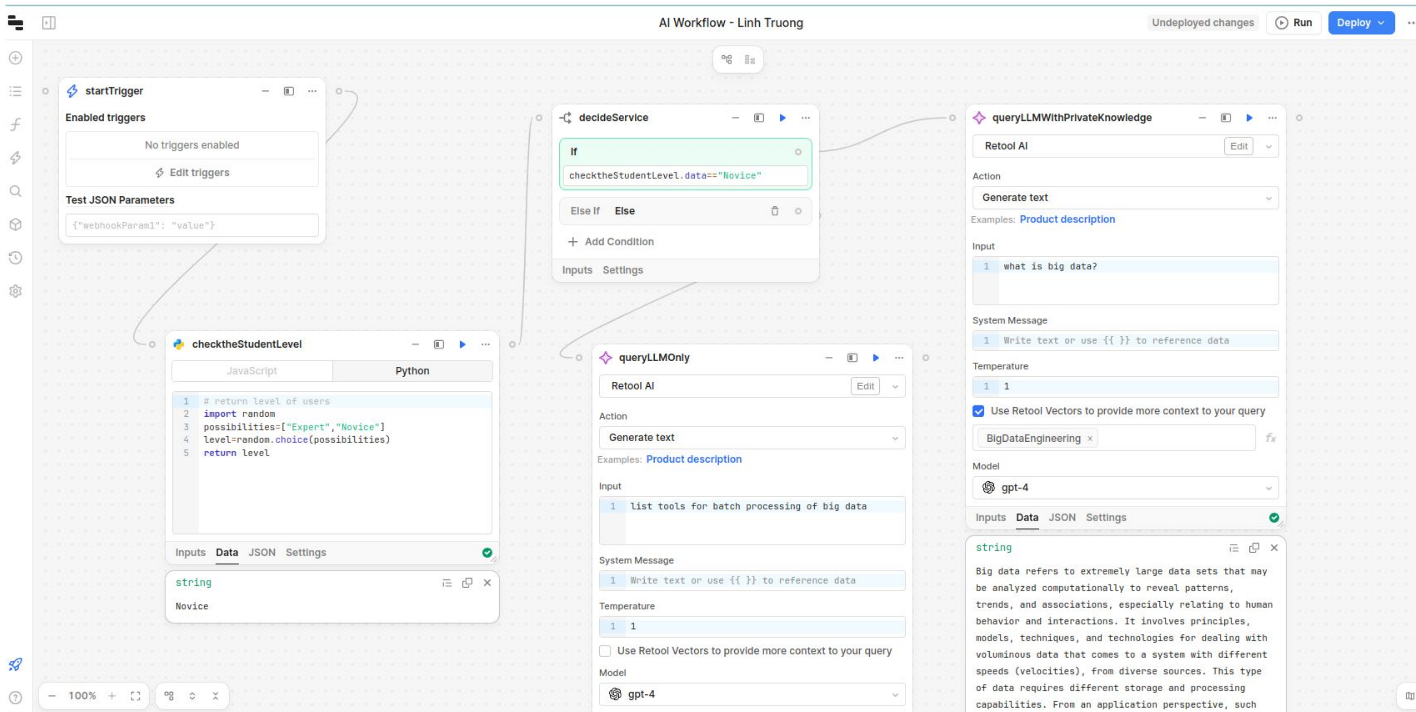
GenAI/LLMs for generating code



Note: Not necessary for big data platforms only!

Workflow of conventional tasks and LLMs-based tasks

Examples with retool, some knowledge is trained based on our basic big data materials (https://github.com/linhsolar/basicbigdata)



Aalto University
School of Science

If we want to include GenAI/LLMs as service in a big data platform

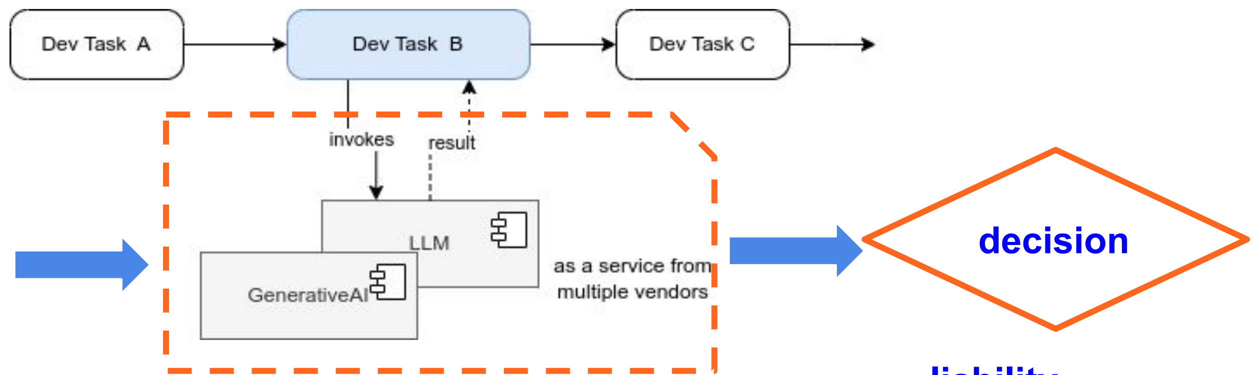
Understand the expected quality of results from GenAI/LLMs

Aspects/Views

as a process
(e.g., inference process for an input)

as an artefact
(e.g., output)

as a service
(e.g., an ML service for a type of tasks)

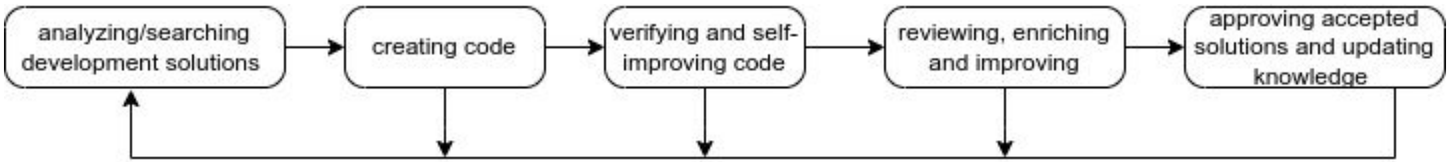


trusted, reliable, bias/fair, secure, cost-effective results?

liability, regulation, risk, ethics, suitainability

Understand the need to coordinate LLMs and services for development tasks

- **Targets: focused tasks and customized environments for the developer in big data**

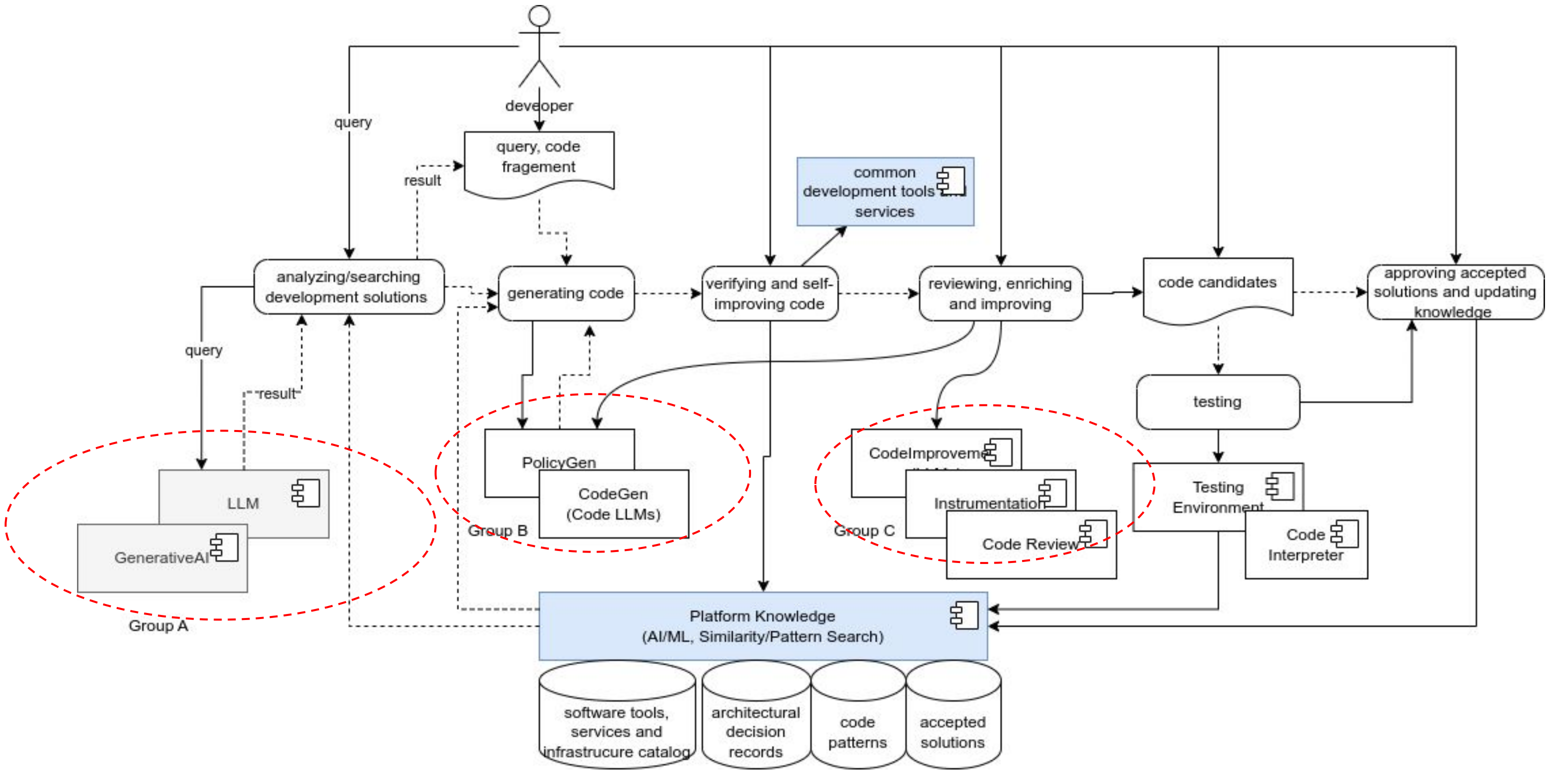


- **Start from the Platform Knowledge**
  - encapsulating knowledge about the big data platform and application domains
  - knowledge not easily found in the public space/open sources
  - a kind of "internal" data services

Understand the need to coordinate LLMs and services for development tasks

- **LLMs with Platform Knowledge**
  - composition of LLMs with other development tools/knowledge sources, which are highly accurate and relevant
  - support for both pay-per-use and private deployment LLMs services
- **Human-in-loop for uncertainty handling and quality feedback**
- **Adaptive coordination workflows based on quality associated with individual LLM capabilities**
  - observability; explainability; reliability

Collms approach: a high level view



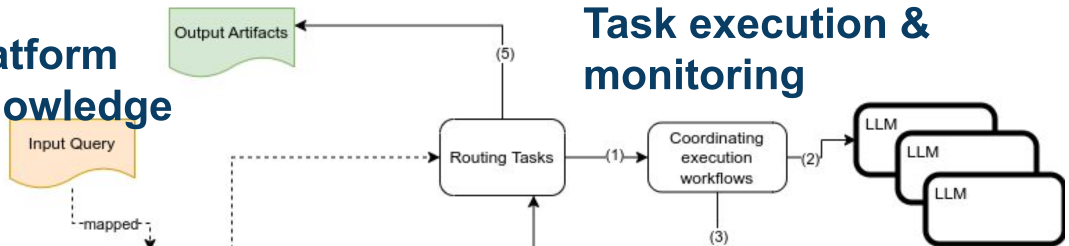
Service group: for specific type of tasks, at runtime an ensemble from the group will be used

Collms approach: a high level view

Cross tasks lineage

Platform Knowledge

Task execution & monitoring




```
"task": {
    "task_id": "5e6ac4bb-b465-44ec-912e-7816ba788a3b",
    "task_type": "codegen",
    "execution_mode": "parallel",
    "input_data": "show total 5G downlink volume in a month",
    "input_context": null,
    "services": [
        "Defog"
    ],
    "params": null,
    "experiment_id": "970f99e0-614a-4b4e-984c-7b70b90438f1"
}
```


Platform Knowledge

Quality evaluation

Task models & (de)composition

Trails of task execution & quality


```
{
    "task_instance_id": "2f25cd24-e5dc-4437-8949-95788371c353",
    "start_ts": 1698430168.4745324,
    "end_ts": 1698430184.2641177,
    "service_name": "Defog",
    "service_qoa": {
        "total_token": null,
        "responsetime": 15.789584357000422,
        "cost": null
    },
    "task": {
        "task_id": "5e6ac4bb-b465-44ec-912e-7816ba788a3b",
        "task_type": "codegen",
        "execution_mode": "parallel",
        "input_data": "show total 5G downlink volume in a month",
        "input_context": null,
        "services": [
            "Defog"
        ],
        "params": null,
        "experiment_id": "970f99e0-614a-4b4e-984c-7b70b90438f1"
    }
}
```

Employing LLMs: service selection and integration problems

- **Big LLMs**
  - unless you develop a big platform, mostly you use existing ones
  - which tasks can we use as we may not know how the GenAI/LLMs services deal with our own data
- **Small LLMs**
  - can be hosted within a platform, dedicated for individual tenants
- **Diversity of LLMs**
  - functionality: generating reports, code, requirement templates, etc
- **Engineering challenges**
  - scalability, data privacy/governance, and multitenant models

Tools

- **Many cloud services (with close and open source LLM models)**
  - ChatGPT, Azure Mistral, together.ai, Text-to-SQL Vanna - https://vanna.ai/docs/

Table 4: Pass@1 performance of pretrained code models (top), instruction finetuned code models (middle), in comparison with some of the best general language models (bottom), with models in each category ordered chronologically. The sources of these figures can be found in Section 5.3, Section 5.4, and Table 1.

Model	Size	HumanEval	MBPP
PolyCoder	2.7B	5.6	-
CodeGen-Mono	16.1B	29.3	35.3
InCoder	6.7B	15.2	19.4
PyCodeGPT	110M	8.3	-
Pangu-Coder	2.6B	23.8	23.0
SantaCoder	1.1B	14.0	35.0
CodeGeeX	13B	22.9	24.4
StarCoder	15.5B	33.6	52.7
CodeT5+	16B	30.9	-
Phi-1	1.3B	50.6	55.5
CodeFuse	13B	24.8	-
DeepSeek Coder	33B	56.1	66.0
InstructCodeT5+	16B	35.0	-
WizardCoder	15.5B	57.3	51.8
Pangu-Coder 2	15.5B	61.6	-
OctoCoder	15.5B	46.2	-
CodeFuse	34B	74.4	-
DeepSeek Coder-Instruct	33B	79.3	70.0
GPT-4	-	67.0/82	-
PaLM 2*	S	37.6	50.0
Code LLaMA	34B	53.7	56.2
Phi-1.5	1.3B	41.4	43.5

Source: Zhang et al., "**Unifying the Perspectives of NLP and Software Engineering: A Survey on Language Models for Code**", https://arxiv.org/abs/2311.07989

Aalto University
School of Science

Using techniques/methods studied in Big Data Platforms for GenAI/ML/LLMs

“No AI Without Data”: before LLMs

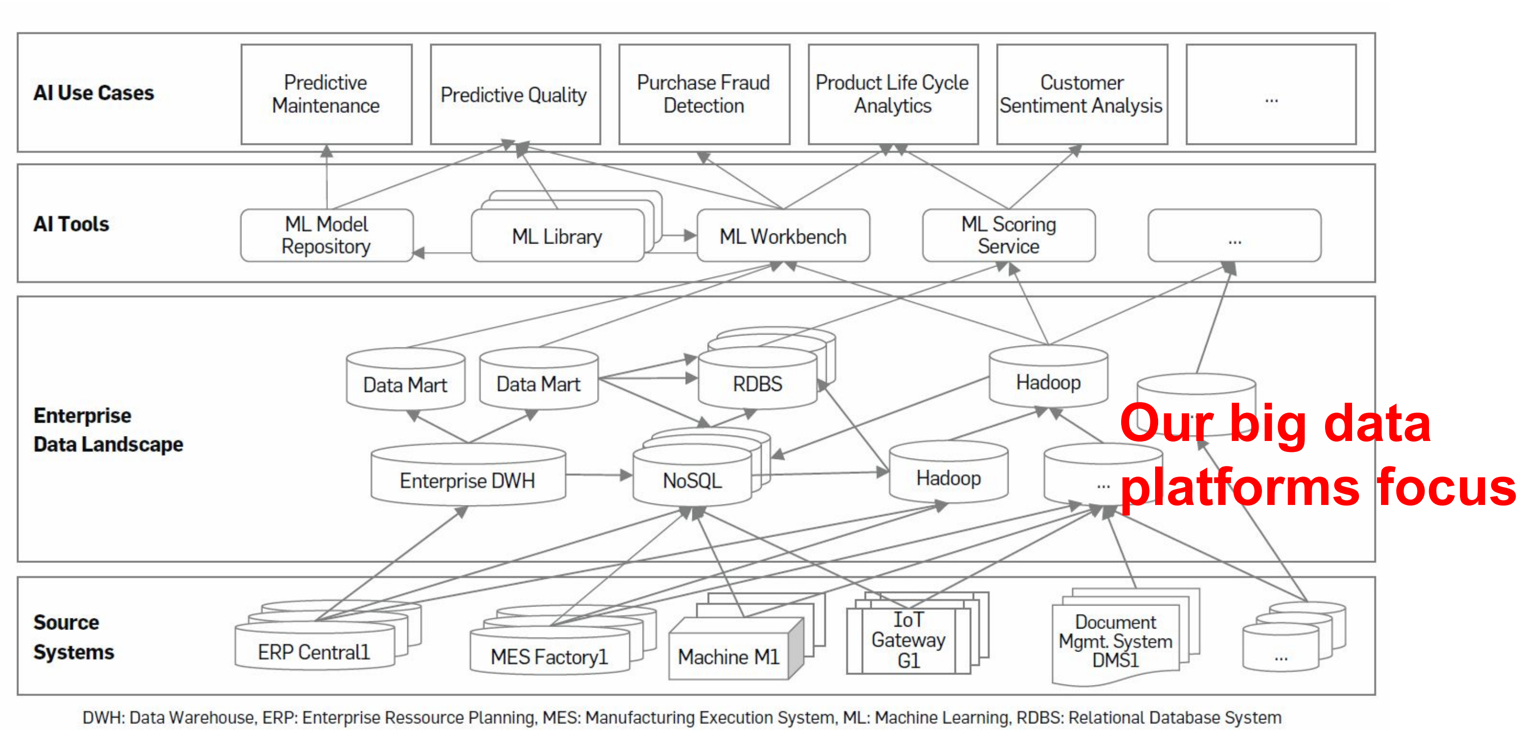
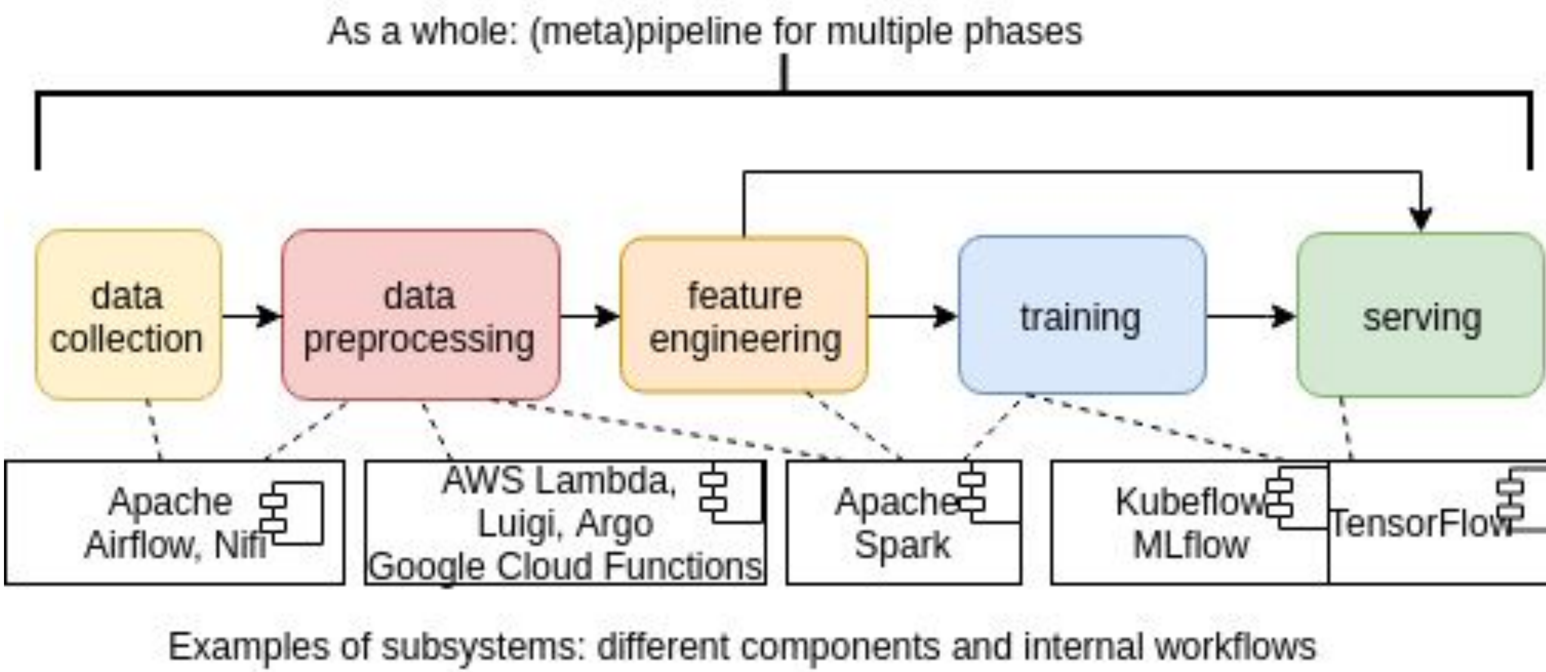


Figure source: There Is No AI Without Data, by Christoph Gröger, Communications of the ACM, November 2021, Vol. 64 No. 11, Pages 98-108, 0.1145/3448247

Big data and ML pipelines



Big data platforms and ML

It is easy to see the role of big data components we study in this figure

Example: Michelangelo ML from Uber

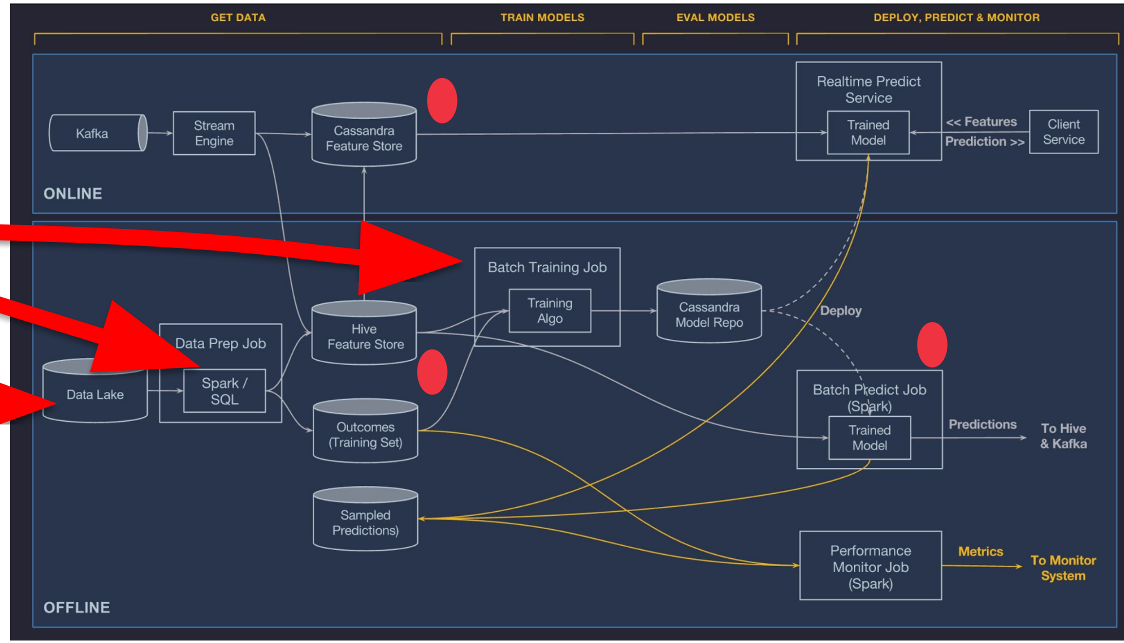


Figure source: https://eng.uber.com/michelangelo-machine-learning-platform/

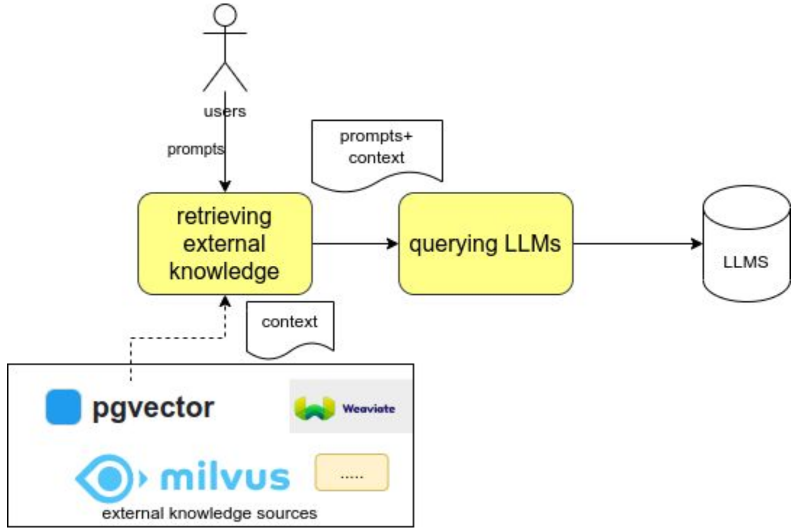
How techniques and methods in big data platforms help to build GenAI/LLMs

- **GenAI/LLM as a key part that big data platforms must support**
  - analytics workflows and service integration
- **Data management for GenAI/LLMs**
  - enable sharing, retrieving and managing big data
- **Techniques and methods for GenAI/LLMs development**
  - connectors, transformation, data pipelines, data governance

Support LLMs: the big data pipeline for RAG in GenAI/LLMs

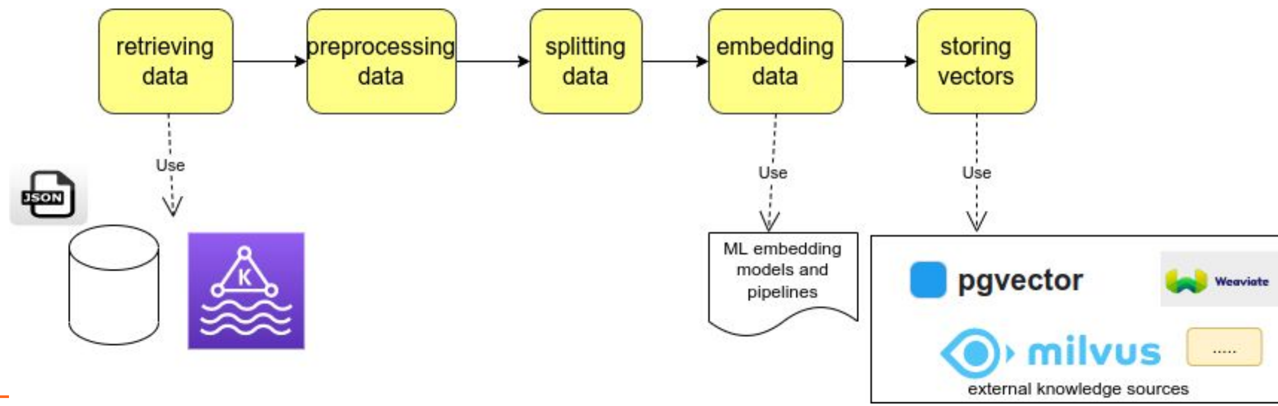
- **RAG: Retrieval-Augmented Generation**
  - getting facts/relevant accurate knowledge from an external source for LLMs
  - providing context for improving LLM inferences
- **Key components**
  - pipelines and databases

RAG for LLMs

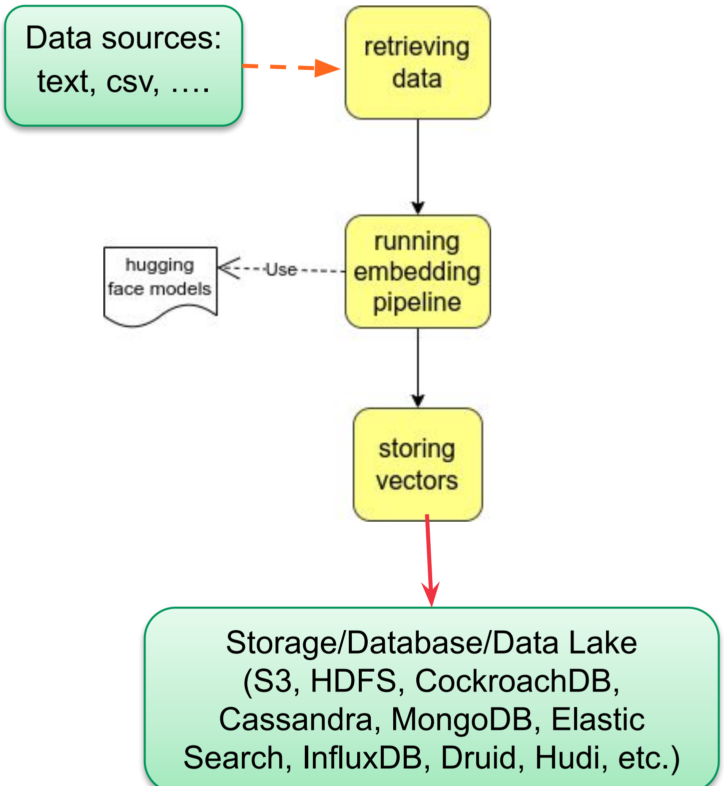


Support LLMs: the big data pipeline for RAG in GenAI/LLMs

- **The process to build RAG is a "big data" pipeline**
  - getting data and determining vectors as indexes for data
  - storing data and indexes into databases
  - using common vector databases
    - *Postgres with pgvector, ChromaDB, Milvus, Weaviate, …*



An example with LlamaIndex

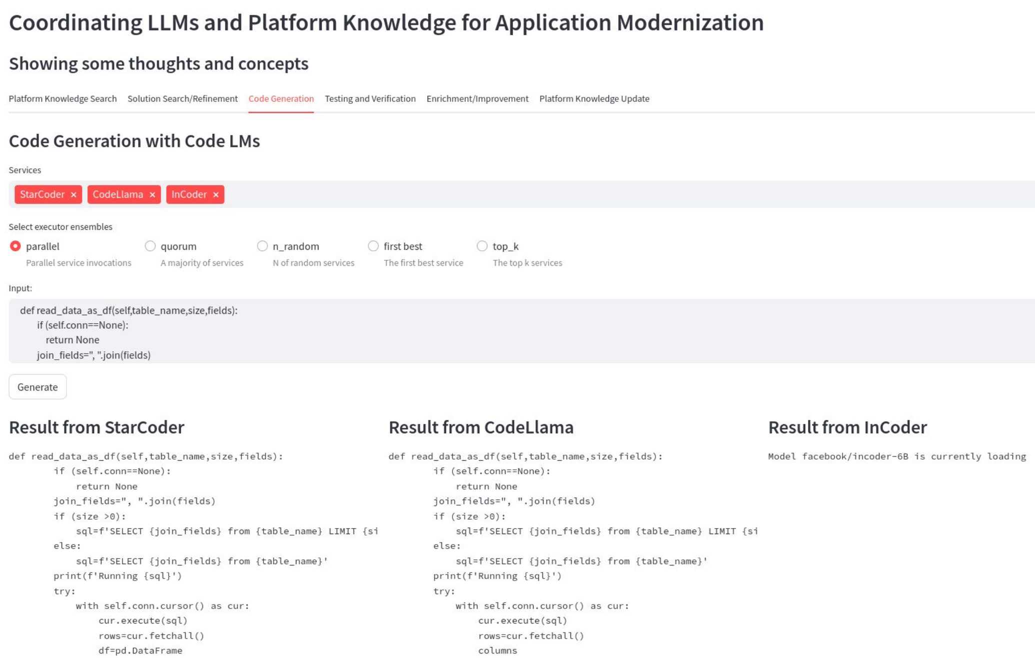


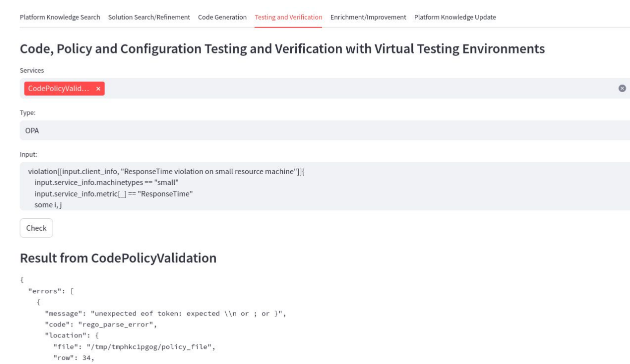

```python
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader, Settings
from llama_index.core.embeddings import resolve_embed_model

#assume the data is under "data" (e.g., csv and text file)
documents = SimpleDirectoryReader("data").load_data()

#specify embedding model for extracting vectors
Settings.embed_model = resolve_embed_model("local:BAAI/bge-small-en-v1.5")

#configure LLM, set None as it illustrates the RAG
Settings.llm = None
# specify the embedding pipeline/task
index = VectorStoreIndex.from_documents(
    documents,
)
#Now we can query Vector to get a context
retriever = index.as_retriever()
query=input("Enter your query:")
contexts = retriever.retrieve(query.strip())
# show context, assume that it context=NodeWithScore with node=TextNode
for context in contexts:
    print(context.node.text)
# also query LLM (but none in this code)
query_engine = index.as_query_engine()
result = query_engine.query(query.strip())
print(result)
```

Collms approach: current prototype

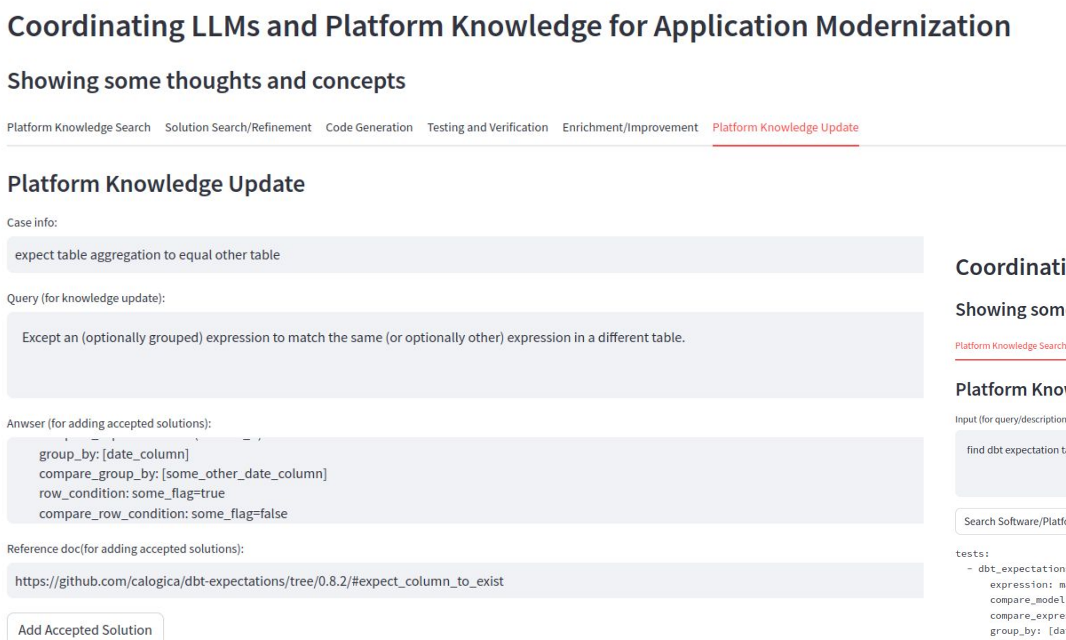
- **AaltoSEA prototype**
- **Free/trial LLMs from Hugging Face and from self-deployment**
  - capabilities are quite limited with REST and AMQP protocols
- **Some basic services for software development**
- **Platform Knowledge**
  - vector databases: Weaviate (Milvus is another possible for complex integration)
  - Sample code/policies: mainly from teaching, industrial trainings, and research prototype codes for IoT, edge and clouds

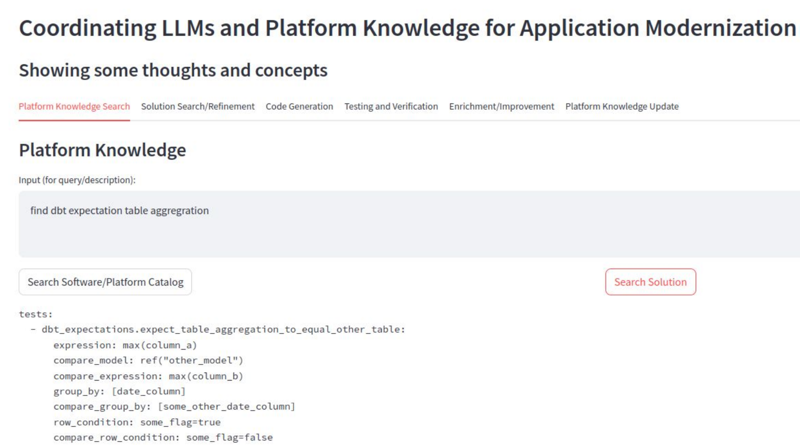
Collms approach: few snapshots of the Collms prototype (AaltoSEA)





Collms approach: few snapshots of the Collms prototype (AaltoSEA)





Summary

- **Complex design and engineering issues in big data platforms**
  - many big systems, very complex designs and requirements
  - require us to continuously upskill
- **What we learn so far are just “foundations”**
  - not much on emerging techniques and models due to advances of systems and new problems
- **New GenAI/LLMs can help simplifying certain tasks**
- **Your expertises from Big Data Platforms can help**
  - integrating GenAI/LLM services
  - building GenAI/LLMs capabilities for big data problems

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io