Open-source Framework for Synthetic Fraud Dataset via Generative AI

Insights from Industrial Secondment at IBM

Micol Olocco (TU Dortmund)
micol.olocco@cern.ch

Pierre Feillet (IBM France Lab Saclay)
PEILLET@fr.ibm.com

Image: https://esgnews.com/it/IBM-apporta-nuove-funzionalit%C3%A0-al-suo-software-di-sostenibilit%C3%A0-per-aiutare-le-organizzazioni-a-rendere-operativi-in-%E2%80%8B%E2%80%8Bmodo-accurato-i-dati-sulle-emissioni-di-gas-serra-dell%27ambito-3/

SMARTHEP received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement n. 956086

SMARTHEP

Real-time Analysis for Science and Industry

SMARTHEP
REAL-TIME ANALYSIS FOR SCIENCE AND INDUSTRY

technische universität dortmund

The SMARTHEP Network is a European research project funded by the European Commission, focused on applying real-time analysis to particle physics and broader industry challenges.

Two month internship at IBM France Lab Saclay with Pierre Feillet

Open-source Framework for Synthetic Fraud Dataset via Generative AI

Why a synthetic dataset?

Why via Generative AI (specifically LLMs)?

dis

Disclaimer: I am by far an expert of banking operations or LLMs

Open-source Framework for Synthetic Fraud Dataset via Generative AI

Why a synthetic dataset?

Why via Generative AI (specifically LLMs)?

Why a synthetic dataset?

Data Scarcity

Interpretability (dataset control)

Sensitive Data

Simulations in different fraudulent scenarios

Train supervised/unsupervised algorithms for fraud detections

Modelling banking activities

- Sequence of banking activities ~ sequence of states with transition probabilities (Markov Chains)

From \ To	Deposit	ATM Withdrawal	Card Payment	Online Transfer	No Activity
Deposit	0.05	0.30	0.50	0.10	0.05
ATM Withdrawal	0.10	0.20	0.40	0.20	0.10
Card Payment	0.05	0.15	0.60	0.10	0.10
Online Transfer	0.10	0.10	0.30	0.30	0.20
No Activity	0.20	0.10	0.30	0.20	0.20

Simple example: 1st order Markov Chain

Modelling banking activities

- Sequence of banking activities ~ sequence of states with transition probabilities (Markov Chains)

  → *modeling* a *huge* and *sparse* transition matrix (even for a 1st order Markov Chain)

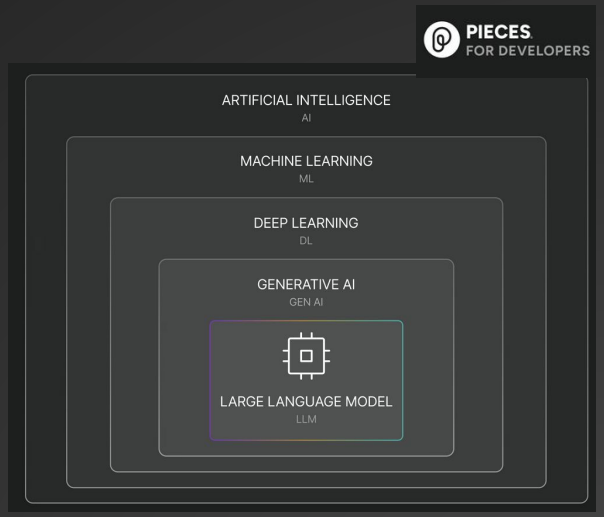
- Idea: replace the probability matrix with a Large Language Model (LLM)

→ Query the LLM to predict the next most probable activity

Generative AI & LLMs (source)

Generative AI: a deep-learning model able of generating diverse form of data (text, video, images, music) depending on the training data used.

- Training data: several pictures of cats.
- Learning process: recognize patterns and the distribution of pixels throughout the images
- Outcome: able to produce its own pictures of cats.
- Example: DALL E, AI photoshops



https://www.infobip.com/blog/large-language-models-vs-generative-ai

Large Language Models: a generative AI model specialized in text input/output (ChatGPT, Gemini, Llama, Mistral, Deepseek...)

Why Large Language Models?

Key Assumption: Pretrained Large Language Models (LLMs) have implicitly learned probabilistic patterns of human and financial behavior from large-scale corpora such as news articles, fraud cases, banking regulations, and user interactions.

Traditional approach (k-th order Markov chain): t he next activity depends on the previous k activities:

$$P(A_t \mid A_{t-1}, A_{t-2}, \ldots, A_{t-k})$$

This requires a transition matrix:

$$T \in \mathbb{R}^{N^k \times N}$$

As k increases, the matrix becomes exponentially larger and sparser.

LLM-based approach: instead of storing transition probabilities explicitly, we use an LLM to predict the next activity based on the history:

$$A_t \sim \mathrm{LLM}(A_{<t})$$

This shifts the modeling from sparse matrix storage to flexible, context-aware prediction.

Advantages

- No need to store or learn huge transition matrices
- Doesn't require the developer to have domain expertise
- Exploit LLM's creativity to possibly generate new fraud patterns: make trained Fraud Detection algorithms more robust

Key Assumption: Pretrained Large Language Models (LLMs) have implicitly learned probabilistic patterns of human and financial behavior from large-scale corpora such as news articles, fraud cases, banking regulations, and user interactions.

Framework Infrastructure for LLM calls

- Ollama (local call to downloaded pre-trained LLMs)

Pros:
- free
- local (no WIFI)
- easy to integrate

Cons:
- very limited by GPU availability (to generate a full dataset is impossible!), especially with 'heavy' models
- local (needs to fit models on memory) → not very efficient for model comparison

- IBM Watsonx (local call to remotely hosted pre-trained LLMs)

Pros:
- easy web interface for prompt testing
- possibility of generating big datset

Cons:
- need to pay for input/output tokens (price vary on the model)

Framework Infrastructure for LLM calls

```
prompt = build_generation_prompt(strategy, global_clock, user_id, past_errors)
response = ollama.chat(model=LLM_model, messages=[{"role": "user", "content": prompt}], )
```

Ollama

```
53  def watsonx_chat(prompt, model_id, parameters):
54      """Sends the prompt to WatsonxLLM and returns the text response."""
55      # Initialize your WatsonxLLM instance as shown in your Watson script:
56      llm = WatsonxLLM(
57          model_id=model_id,
58          url=credentials["url"],
59          apikey=credentials["apikey"],
60          project_id=project_id,
61          params=parameters
62      )
63      # Create a simple prompt template that simply passes through the prompt
64      prompt_template = PromptTemplate(
65           input_variables=["prompt"],
66           template="{prompt}"
67      )
68      chain = LLMChain(llm=llm, prompt=prompt_template, output_key='output')
69      response = chain.invoke({"prompt": prompt})
70      return response["output"]
```

IBM watsonx

The simulation framework

Objective: build a realistic time-serie dataset made of banking activities associated to different users.

Approach: associate users to different behaviors (can be fraudulent or legitimate) and simulate an *x* number of activities for each user.

- Banking activity: single instance of a banking operation described by different fields.
- Pattern (strategy): what defines the user's behavior in terms of banking activities.

Field
transaction_id
bank_timestamp
local_timestamp
user_id
account_id
type
amount
currency
balance_before
balance_after
location
ip_address
device_id
network_type
merchant_name
recipient_id
recipient_bank
granted

Example of banking activity field list

The simulation framework

What we need:

- Pattern (strategy): what defines the user's behavior in terms of banking activities.
  - Strategist LLM
- Banking activity: single instance of a banking operation described by different fields.
  - Agent LLM

The Strategist LLM

```
TOP_10_FRAUD_TYPES = [
    "Money Laundering", "Account Takeover", "Synthetic Identity Fraud", "Identity Theft",
    "Card Skimming", "Loan Fraud", "Check Fraud", "Wire Fraud",
    "Ponzi Scheme", "Cryptocurrency Fraud", "Insider Trading"
]

TOP_10_LEGITIMATE_PROFILES = [
    "Saver", "Investor", "Traveler", "Everyday Spender",
    "Business Owner", "Student", "Retiree", "Frequent Online Shopper",
    "Tech Professional", "Freelancer"
]
```

- Task: provide the specifications (typical transactions, location, amount etc) associated to a certain profile.

Example of generated strategy

```
profile_or_fraud_type": "Cryptocurrency Fraud",
  "n_accounts": 5,
  "involves_hijacking": true,
  "transaction_types_involved": ["Purchase", "Transfer Out",
"Withdrawal"],
  "typical_amount_range": "$100 - $5000",
  "geographic_focus": ["Domestic US", "East Asia", "Southeast
Asia"],
  "velocity": "3-5 transactions per day",
  "expected_time_gap": "5-30 minutes between transactions",
  "network_types": ["Wi-Fi", "Cellular", "VPN Connection"],
  "common_devices": ["iPhone-13", "MacBook Pro", "Samsung Galaxy
S22"],
  "ip_ranges": ["73.x.x.x", "203.x.x.x", "45.x.x.x"],
  "common_merchant_names": ["Binance", "Coinbase", "Kraken"],
  "common_recipient_ids": ["ACC-774683nf", "ACC-836gfu98",
"ACC-467382hf"],
  "common_recipient_banks": ["Bank of America", "Wells Fargo",
"HSBC"],
  "context": "This strategy involves fraudulent activities using
compromised accounts to purchase, transfer, and withdraw
cryptocurrencies across various exchanges and banks, often with a
mix of domestic and international transactions."
}
```

The Agent LLM

Task: for each user, predict the next most probable activity (or a set of activities) based on the user past activities and the user behavior (defined by the related strategy).

Output must be returned in parsable JSON (objective is to build a dataframe)

Field	Legitimate Transaction Example	Fraudulent Transaction Example
transaction_id	123e4567-e89b-12d3-a456-426614174000	987f6543-e21b-45d6-a987-123456789abc
bank_timestamp	2025-02-12T14:24:14+00:00	2025-02-13T02:24:14+08:00
local_timestamp	2025-02-12T09:24:14-05:00	2025-02-13T02:24:14+08:00
user_id	USER-00112233	USER-00998877
account_id	ACC-12345678	ACC-87654321
type	Purchase	Transfer Out
amount	125.50	3,000.00
currency	USD	USD
balance_before	5000.00	10,000.00
balance_after	4,699.25	7,000.00
location	New York, USA	Manila, Philippines
ip_address	73.112.45.12	203.167.45.88
device_id	iPhone-13	Samsung Galaxy S22
network_type	Wi-Fi	Cellular
merchant_name	Amazon	null
recipient_id	null	REC-09876543
recipient_bank	null	Bank of XYZ
granted	True	True
fraud_label	0	1
behavior_type	Legitimate	Identity Theft

The simulation framework: summary

The Strategist LLM	The Agent LLM
TASK: produce a "strategy" for different legitimate and fraudulent behaviors (ex: everyday spender, traveller / identity theft, money laundering..)	**TASK**: Given the user profile described by the strategy and the relative account history, predict the next most probable activity/chain of activities.
OUTPUT: behavior catalogue with specifications about number of accounts, time between transactions, geography distribution, amounts involved, if it involves hijacking, common devices, network types, recipients etc	**OUTPUT**: predicted activity with associated type, time, geographical location, amount etc
LLM REQUIREMENTS : creative and discursive, chain-of-thoughts	**LLM REQUIREMENTS** : concise, good at JSON structuring, good balance between creativity and adherence to prompt
MODEL: `llama-3-405b-instructor`	**MODEL**: `mistral-large (123b)`

Deepseek-r1 (via Ollama)
Example: cryptocurrency schema

Reasoning (<think> block)

<think>
Okay, so I need to define a detailed strategy for simulating cryptocurrency fraud. The user has specified that it's about Cryptocurrency Fraud and wants me to follow certain instructions.

First, I should clearly describe the fraudulent behavior. It seems like they want an example of how this fraud would manifest in transactions. Let me think about common cryptocurrency fraud schemes. One effective method is insider trading or market manipulation where someone with non-public information makes unauthorized trades.

Next, I need to determine if it's a single-event or multi-event fraud. Insider trading can be done in multiple steps: gathering info, making the first trade, then following up based on subsequent developments. That sounds like it requires multiple transactions over time, so I'll mark it as multi-event.

Now, how many activities are needed for recognition? Since insider trading often involves incremental steps and monitoring the trader's behavior after the fact can help detect patterns, maybe around 5 to 7 activities would be sufficient. This gives enough data points without being too lengthy.

The key characteristics of these activities should include transaction types—purchases, sales, transfers. The amounts might vary but should be significant for impact. Geographically, it could start locally and expand to international markets as the fraud becomes more noticeable. Velocity is crucial because rapid trades can capitalize on short-term info, so high velocity makes sense.

In terms of distance between transactions, they likely happen quickly after gathering information or in between regular trading hours. The timeframe should be tight to allow for quick execution based on timely data but also spaced enough to avoid immediate suspicion if monitored closely.

For the context section, I need a narrative that explains how this fraud unfolds naturally over time, with each transaction adding more evidence of insider knowledge and manipulation. Mentioning the psychological aspect might help explain why fraudsters engage in this behavior repeatedly—reinforcement through repeated success after gathering non-public information.

Putting it all together, I'll structure the JSON output according to the given example, ensuring that each field is filled appropriately with specific details about the transaction types, amounts, geographic patterns, velocity, and the context that explains how this fraud would typically develop over time.
</think>

Deepseek-r1 (via Ollama)

Example: cryptocurrency schema

Effective output


```
{
  "Fraud Type": "Insider Trading - Cryptocurrency Market Manipulation",
  "Scope": "Multi-event",
  "Minimum Activities": 5,
  "Description": {
    "Transaction Types Involved": [
      "Unauthorized Stock Purchases",
      "Unauthorized Stock Sales"
    ],
    "Typical Transaction Amounts": [
      "Between $10,000 and $100,000 per transaction"
    ],
    "Geographic Patterns": [
      "Concentrated within the organization's region, occasional international trades to minimize detection"
    ],
    "Velocity": "High velocity with rapid execution after gathering non-public information",
    "Distance Between Transactions": "Minutes to hours between trades based on timely data"
  },
   "Context": "Insider Trading involves employees or individuals with access to non-public information executing unauthorized stock trades. These trades often occur in rapid succession, with the individual purchasing stock before a major positive announcement or selling it before a negative one. The activity is characterized by sudden, unexplained trading behavior inconsistent with the individual's usual patterns. This fraud can be facilitated by gathering non-public information, such as insider knowledge of upcoming events, product launches, or financial data. The trades are often concentrated in the company's geographic region but may also include international trades to minimize detection and avoid scrutiny within the company."
}
```

Llama (via IBM Watsonx)

Example: cryptocurrency schema

Effective output only


```
profile_or_fraud_type": "Cryptocurrency Fraud",
  "n_accounts": 5,
  "involves_hijacking": true,
  "transaction_types_involved": ["Purchase", "Transfer Out", "Withdrawal"],
  "typical_amount_range": "$100 - $5000",
  "geographic_focus": ["Domestic US", "East Asia", "Southeast Asia"],
  "velocity": "3-5 transactions per day",
  "expected_time_gap": "5-30 minutes between transactions",
  "network_types": ["Wi-Fi", "Cellular", "VPN Connection"],
  "common_devices": ["iPhone-13", "MacBook Pro", "Samsung Galaxy S22"],
  "ip_ranges": ["73.x.x.x", "203.x.x.x", "45.x.x.x"],
  "common_merchant_names": ["Binance", "Coinbase", "Kraken"],
  "common_recipient_ids": ["ACC-774683nf", "ACC-836gfu98", "ACC-467382hf"],
  "common_recipient_banks": ["Bank of America", "Wells Fargo", "HSBC"],
  "context": "This strategy involves fraudulent activities using compromised accounts to purchase, transfer, and
withdraw cryptocurrencies across various exchanges and banks, often with a mix of domestic and international
transactions."
}
```

Llama (via IBM Watsonx)

Example: cryptocurrency schema

Effective output only

```
profile_or_fraud_type": "Cryptocurrency Fraud",
  "n_accounts": 5,
  "involves_hijacking": true,
  "transaction_types_involved": ["Purchase", "Transfer Out", "Withdrawal"],
  "typical_amount_range": "$100 - $5000",
  "geographic_focus": ["Domestic US", "East Asia", "Southeast Asia"],
  "velocity": "3-5 transactions per day",
  "expected_time_gap": "5-30 minutes between transactions",
  "network_types": ["Wi-Fi", "
  "common_devices": ["iPhone-1
  "ip_ranges": ["73.x.x.x", "2
  "common_merchant_names": ["B
  "common_recipient_ids": ["AC
  "common_recipient_banks": ["
  "context": "This strategy in
withdraw cryptocurrencies acro
transactions."
}
```

	Deepseek-r1:7b	Mistral:7b
Avg Response Time	5.7 minutes	1 minutes
Min Response Time	48 seconds	18 secondes
Max Response Time	28.3 minutes	2.2 minutes
Failure rate	~40%	~10%

Preliminary performance study for sequence generation. Deepseek-r1:7b VS Mistral:7b run locally via Ollama on MacBook Pro (Apple M1 Pro chip, 16GB memory).

Prompting

- Define the context: *"You are an expert in simulating **realistic banking strategies**. Your task is to define the startegy for the profile "{strategy_type}" in a structured JSON format enclosed within <<<START_JSON>>> and <<<END_JSON>>> tags.*
- Give an example: *"**Example Output Format**, don't copy verbatim:"*
- Stress out important concepts: *"Return **ONLY** valid JSON with all fields. **Do NOT add any explanations or extra text**."*
- Order matters
- Be as specific as possible: *"The strategy should include the following details..."*

Dataset requirements and quality

1) Output in *valid* JSON → must be parsed to fill a dataframe
    a) all the specified fields must be filled: no less, no more
    b) valid type fields
    c) correct sequence delimiters
2) Output is not a copy of the given example
3) Dataset must be a time-series
4) Time information must be consistent with location
5) Realistic

Fallback systems

To handle incorrect output generation:

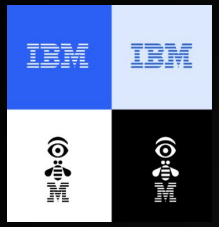
- Manual Corrections: implement ad-hoc corrections based on the most observed errors.
- Reject the generated sequence and create a new one.
- Adaptive Learning: reinforce the prompt with past errors.

Fallback systems

To handle incorrect output generation:

- Manual Corrections: implement ad-hoc corrections based on the most observed errors.
- Reject the generated sequence and create a new one.
- Adaptive Learning: reinforce the prompt with past errors.

Spoiler: it was reinforcing the errors



Fallback systems

To handle incorrect output generation:

- Manual Corrections: implement ad-hoc corrections based on the most observed errors.
- Reject the generated sequence and create a new one.
- Adaptive Learning: reinforce the prompt with past errors.
- Threaten the LLM

Fallback systems

To handle incorrect output generation:

- Manual Corrections: implement ad-hoc corrections based on the most observed errors.
- Reject the generated sequence and create a new one.
- Adaptive Learning: reinforce the prompt with past errors.
- Threaten the LLM

If your output is generated according to my instructions, you will be rewarded a point. If your output is invalid, you will loose points”

Fallback systems

To handle incorrect output generation:

- Manual Corrections: implement ad-hoc corrections based on the most observed errors.
- Reject the generated sequence and create a new one.
- Adaptive Learning: reinforce the prompt with past errors.
- Threaten the LLM

If your output is generated according to my instructions, you will be rewarded a point. If your output is invalid, you will loose points"

Spoiler: it kept on failing

Fallback systems

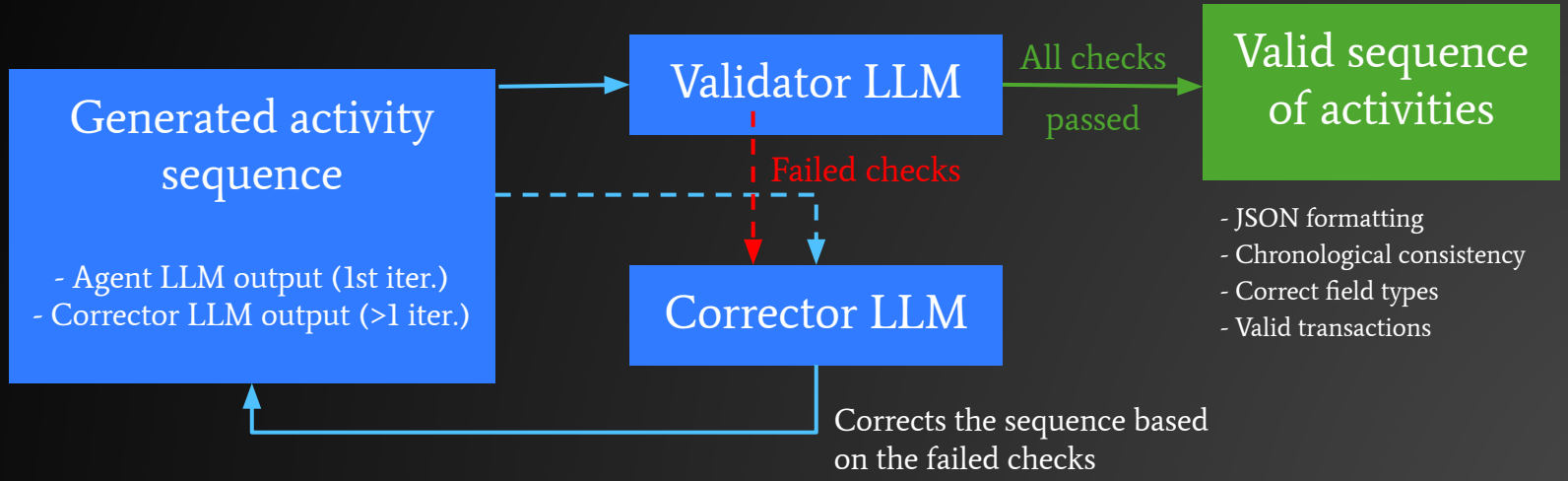
To handle incorrect output generation:

- Manual Corrections: implement ad-hoc corrections based on the most observed errors.
- Reject the generated sequence and create a new one.
- Adaptive Learning: reinforce the prompt with past errors.
- Threaten the LLM
- Generated sequences are validated and eventually corrected by lightweight LLMs

Self-correcting system via LLMs

Generated sequences are validated and eventually corrected by lightweight LLMs:

- Recovers invalid outputs
- Reducing overall operational costs.



Validator and Corrector LLMs

Microsoft

Phi-2: The surprising power of small language models

Validator LLM (Gemma) (2b)

- Compact size and efficienct
- Ideal for rapid, local validation on hardware with limited resources.

Corrector LLM (Phi2) (2b)

- Strong instruction-following capabilities, and reduced tendency to hallucinate compared to larger models.
- Fix structuralerrors without generating extensive new information, focusing strictly on correcting specific, identified issues by the Validator LLM.

Open points

- Is the chain-of-thoughts enough for the ‘LLM interpretability’?
- Best design choice: generation by single activity or by sequence of activities?
- Coupled accounts (i.e “identity theft”) need particular attention
- How can we optimize the realism of the generated dataset?
  - Right now “human-in-the-loop” approach..
  - Reinforcement Learning? Which reward function?
  - Downstream fraud detection accuracy

Possible applications in the LHC context
LLMs and Agent-based modeling

- Simulate realistic insider threats and social engineering (e.g. phishing, impersonation) to test and improve LHC cybersecurity defences.
- Simulate user behaviour across computing and storage systems to predict peak loads and optimise resource allocation and scheduling.
- LLMs can help translate complex or ML-based trigger logic into human-readable explanations, supporting debugging and interpretability.
- AI-assisted methods for data-taking operations:
    - supporting piquets or assisting in data quality evaluation
    - automating parts of the trigger configuration process
    - LLMs to generate technical documentation and onboarding tools like chatbots to help newcomers learn procedures and know who to contact when issues arise.

HEP community and LLMs

chATLAS: An AI assistant for the ATLAS collaboration

Speakers: Joe Egan (University College London), Daniel Thomas Murnane (Niels Bohr Institute, University of Copenhagen)

TARP: Twikis for ATLAS using RAG Protocols

Speaker: Pratik Jawahar (University of Manchester (UK - ATLAS))

Towards Control Room Automation with a Chatbot for DAQ and Shifter Assistance

Speakers: Luigi Podda (CERN), Sylvain Chapeland (CERN)

IML Machine Learning Working Group: Chatbots at CERN

Tuesday 1 Jul 2025, 15:00 → 18:00 Europe/Zurich

LINK

Leveraging Chatbots in daily HEP work - Open Source and Commercial

Speaker: Gordon Watts (University of Washington (US))

Starting a common effort towards a CERN LLM

Speaker: Maurizio Pierini (CERN)

HEPilot: Integrating Embeddings, Vector Search, and RAG for Physics Use Case

Speakers: Michael David Sokoloff (University of Cincinnati (US)), Mohamed Elashri (University of Cincinnati)

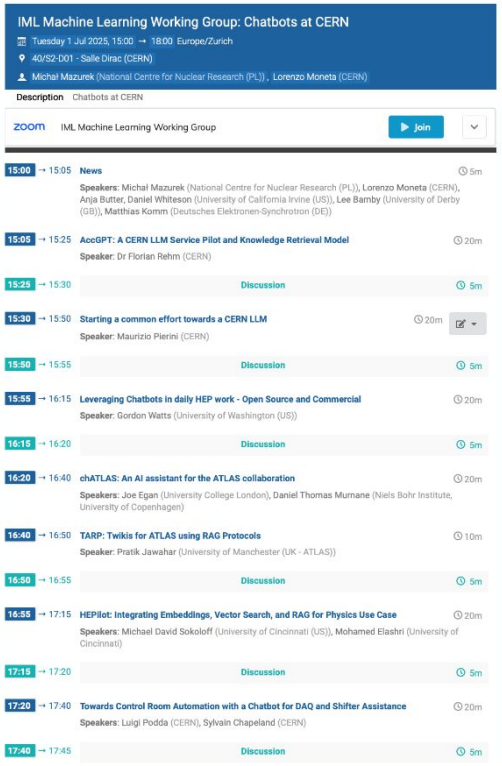
AccGPT: A CERN LLM Service Pilot and Knowledge Retrieval Model

Speaker: Dr Florian Rehm (CERN)

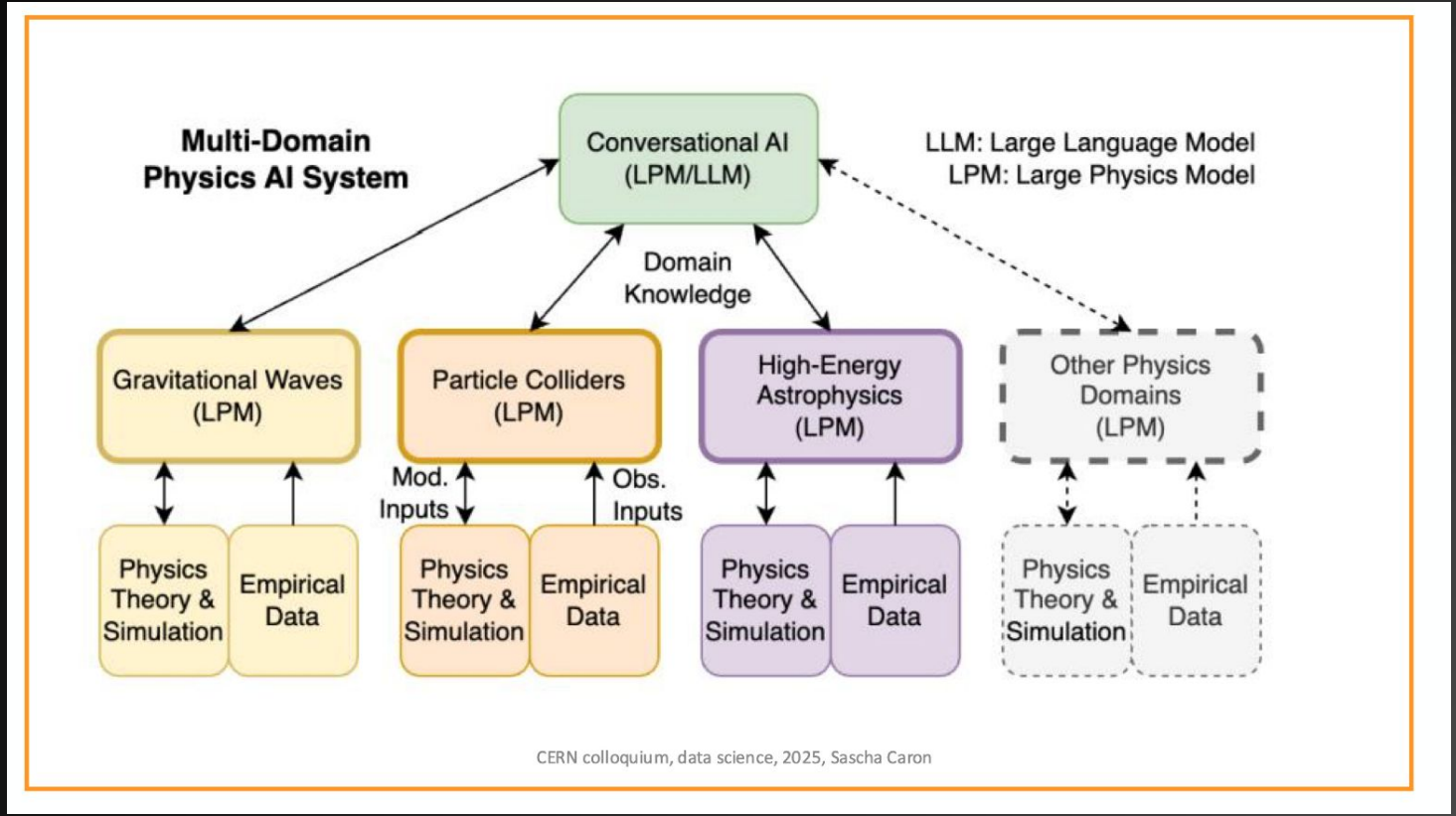
See: “Starting a common effort towards a CERN LLM”

How? Many Ongoing Efforts

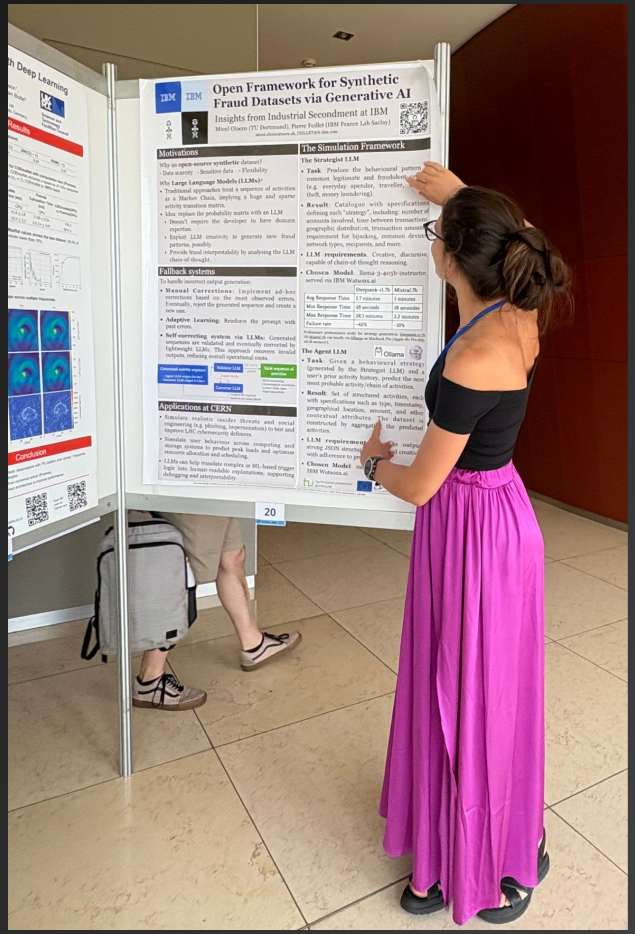
- *Various existing efforts on various fronts*
  - *A lot of progress made in the last few years*
  - *An overview today and here*
- *Many of these efforts struggle with resources*
- *Many of these efforts overlap in scope, but operate in separate environments (e.g., LHC collaborations develop independently)*
- *Can we unify these efforts under a common program, towards a generic LLM for CERN and HEP?*



Large Physics Model



Questions?


Open Framework for Synthetic Fraud Datasets via Generative AI
Insights from Industrial Secondment at IBM
Motivations
The Simulation Framework
Fallback systems
Applications at CERN
20

Disadvantages of Using LLMs Instead of Transition Matrices

1. **Lack of Transparency**
   Transition matrices are interpretable (you can inspect probabilities).
   LLMs act as black boxes — hard to understand *why* a prediction was made.

2. **Higher Computational Cost**
   nference with LLMs is **much more resource-intensive** than matrix lookups.
   Especially problematic for large-scale simulations or low-latency systems.

3. **No Guarantees of Markov Property**
   LLMs don't strictly follow the Markov assumption.
   This can be an issue if you need **formal control** over state transitions.

4. **Dependency on Pretraining Distribution**
   LLMs may generalize from **textual banking knowledge** , but not always in a domain-specific or up-to-date way.
   Risk of **hallucinating unlikely or invalid activities** .

5. **Harder to Validate or Debug**
   A Markov chain is easy to test and audit.
   LLM predictions require more involved evaluation (e.g., sampling, prompt tuning).

6. **Reproducibility and Determinism**
   Matrix-based models are deterministic.
   LLMs often rely on stochastic sampling, which may reduce reproducibility unless you fix seeds and settings.

Transition matrix in 2nd order Markov Chain

Previous (A_{t-2}, A_{t-1})	P(A)	P(B)	P(C)
(A, A)	0.1	0.7	0.2
(A, B)	0.3	0.4	0.3
(A, C)	0.5	0.3	0.2
(B, A)	0.2	0.2	0.6
(B, B)	0.6	0.2	0.2
(B, C)	0.4	0.5	0.1
(C, A)	0.3	0.6	0.1
(C, B)	0.1	0.7	0.2
(C, C)	0.4	0.3	0.3

Let's say our activity space is:

- A = {A, B, C}
  So, N = 3 (number of possible activities)

With a **2nd-order Markov chain** , the transition probability is:

$$P(A_t \mid A_{t-1}, A_{t-2})$$

The transition matrix T now maps each pair of past activities to a probability distribution over next activities:

$$T \in \mathbb{R}^{N^2 \times N} = \mathbb{R}^{9 \times 3}$$

If the last two activities were B then C we look up for the row (B, C): th netx most probable activity is *B*.