

Evaluating Large Language Models for Retrieval-Augmented Generation in Enterprise Applications

*An Evaluation pipeline for Nokia's
"Network as Code" SDK Documentation Assistant*

Student: Tommaso Canova
Supervisor: Jacopo Staiano

NOKIA

Department of Information
Engineering and Computer
Science

Master's Degree in
Artificial Intelligence Systems



Agenda

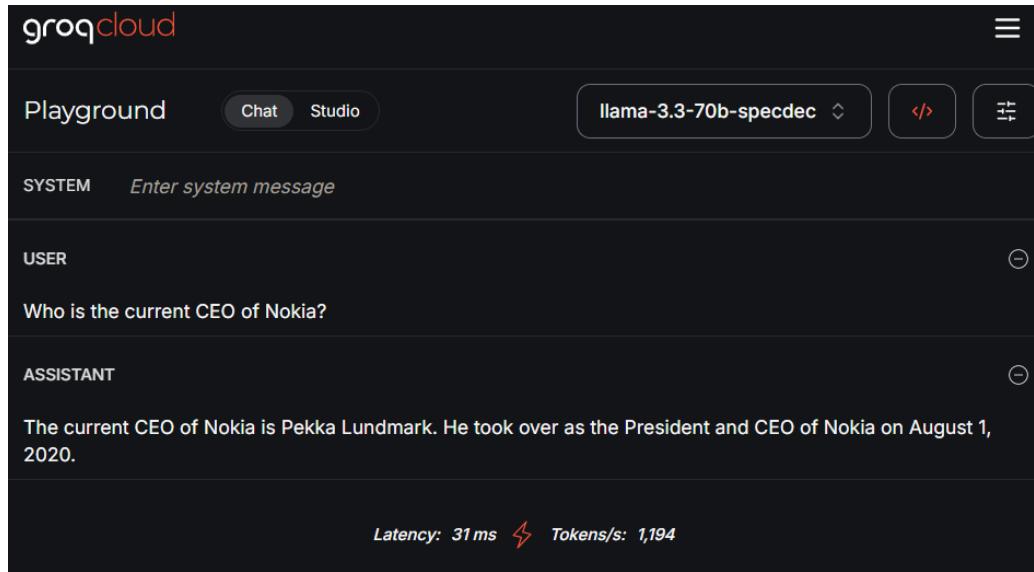
1. Key concepts and Problem statement
2. How can LLMs be evaluated?
3. Proposed evaluation tool
4. Results
5. Limitations and Future Works

Key concepts and Problem statement

Key concepts

Large Language Models

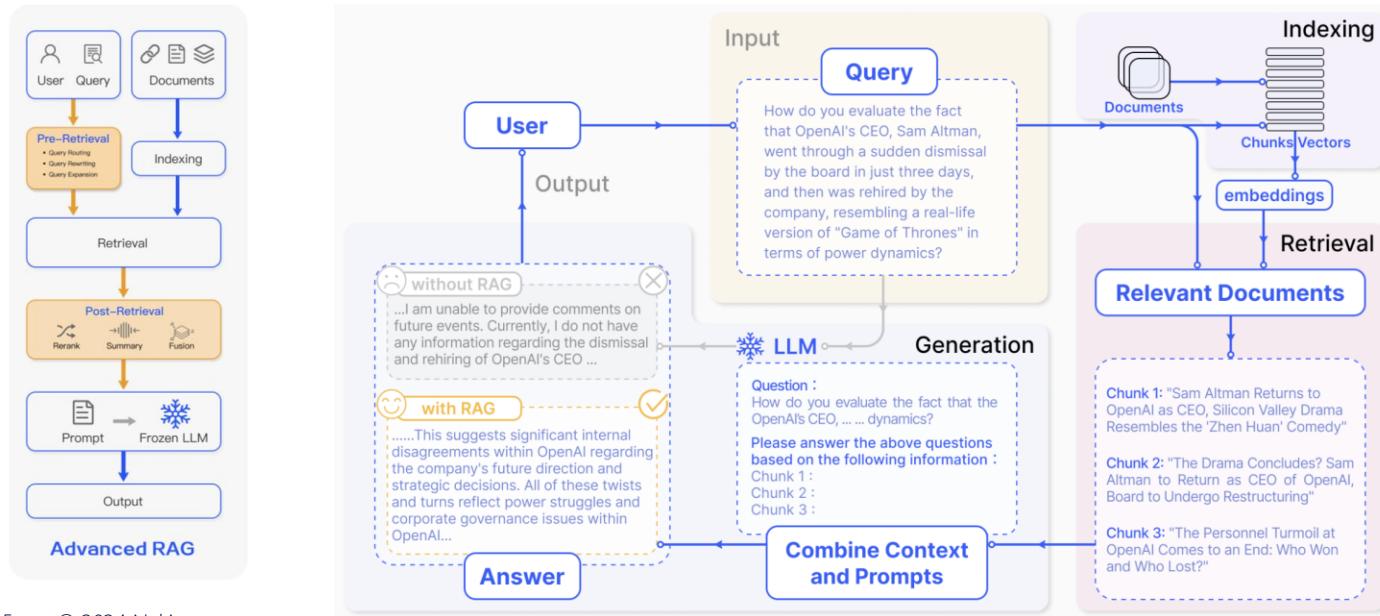
Large Language Models are deep learning models trained on a massive amount of data and capable of understanding and generating human-like language.



Key concepts

Retrieval-Augmented Generation (RAG)

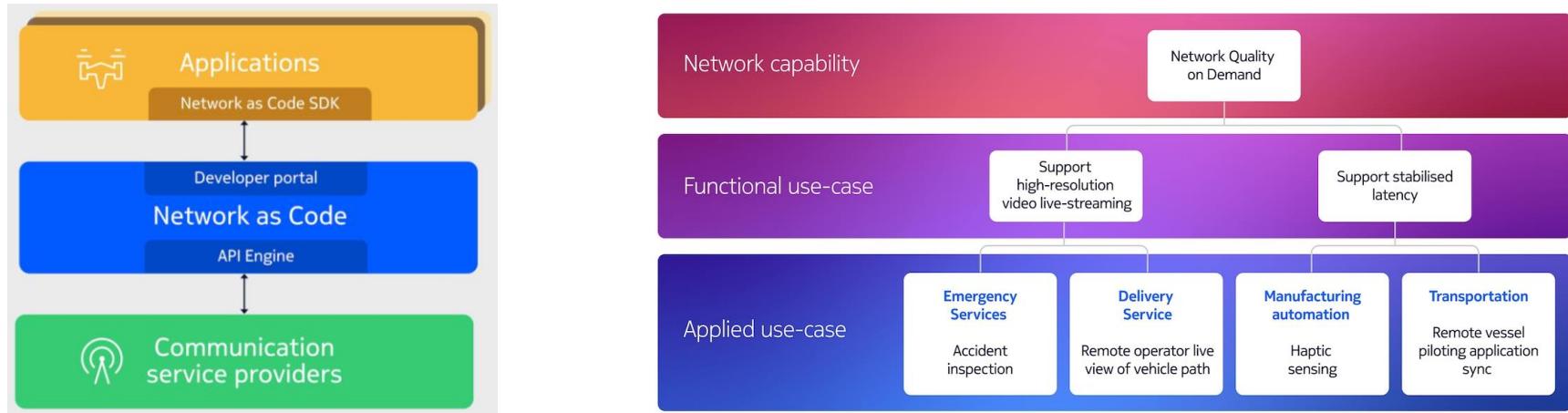
Retrieval-Augmented Generation (RAG) is a machine learning technique that combines a generative model with an external retrieval system to dynamically fetch and incorporate relevant context from large datasets during response generation, enhancing the model's accuracy and grounding.



Key concepts

Network as Code (NaC) SDK

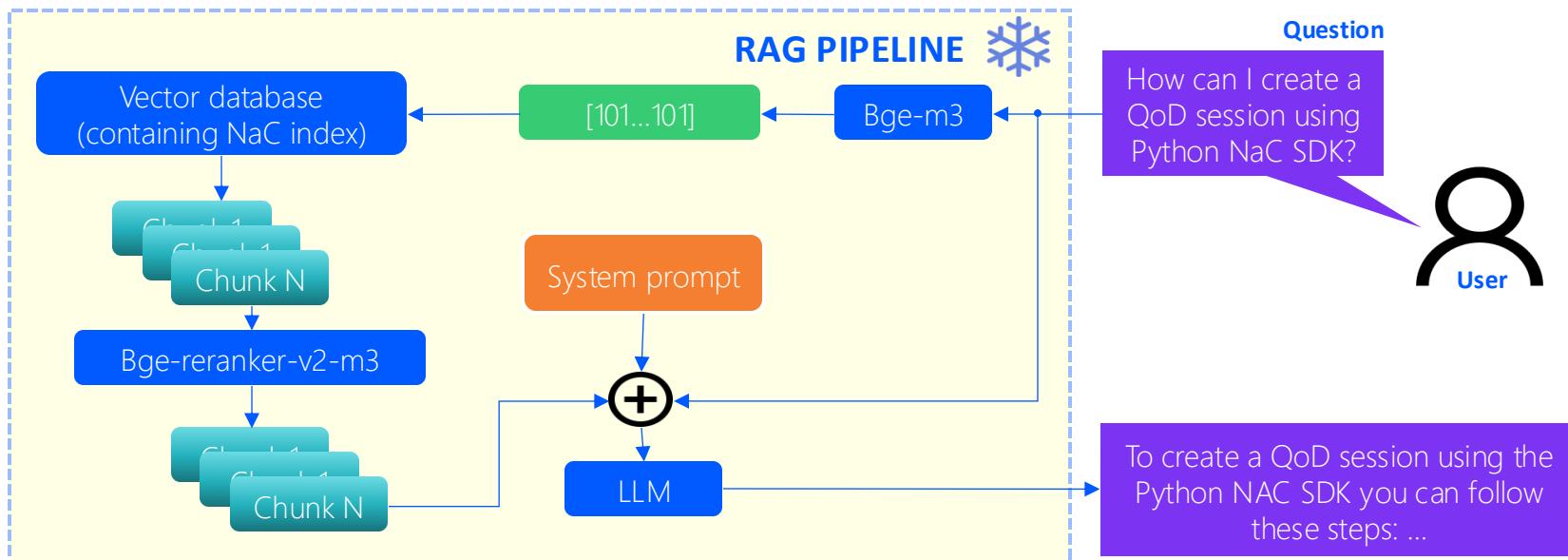
Network as Code is an SDK that **simplifies network management**, enabling developers to integrate advanced 5G features like dynamic resource control, data prioritization, and location-based services through high-level APIs **without telecom expertise**



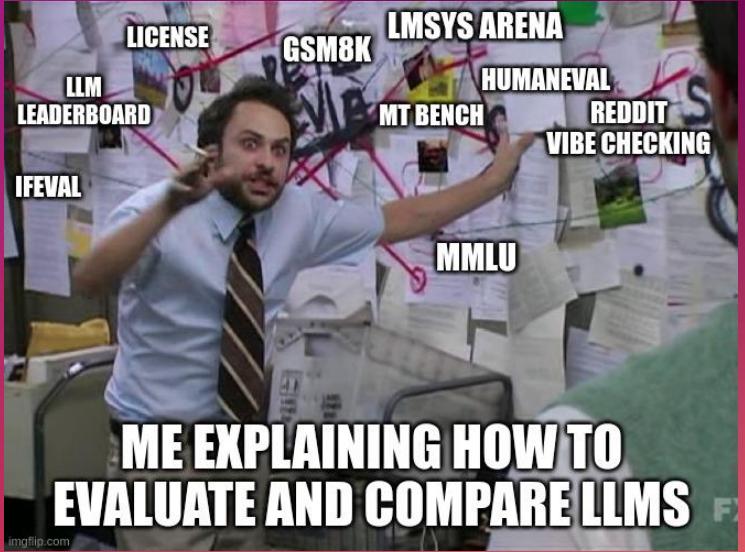
Problem statement

How can we evaluate a Documentation Assistant?

Given a “frozen” RAG pipeline containing the NaC SDK Markdown documentation index (Nokia’s internal Copilot). We want to find a way to **evaluate the performance** of different LLMs on assisting a developer with documentation-related questions.



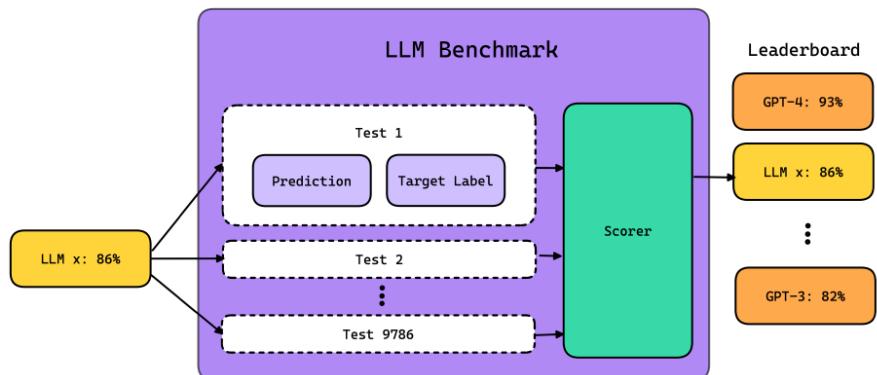
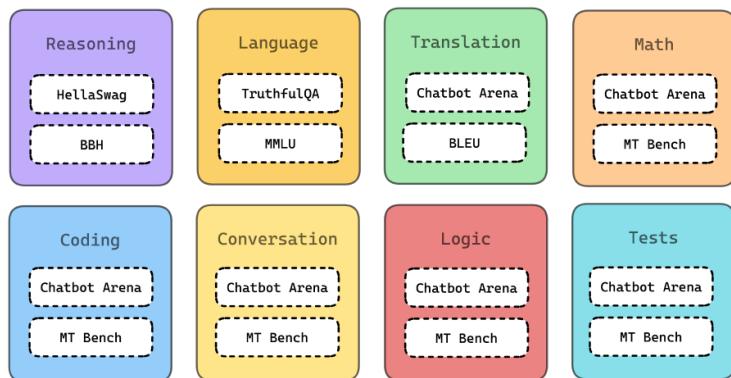
How can LLMs be evaluated?



How can LLMs be evaluated

LLM Benchmark

An LLM benchmark is a **standardized** evaluation framework that uses a **golden dataset** (reference data), and predefined evaluation criteria (such as metrics) to assess the performance, capabilities, and reliability of a large language model across various tasks like comprehension, generation, and reasoning.



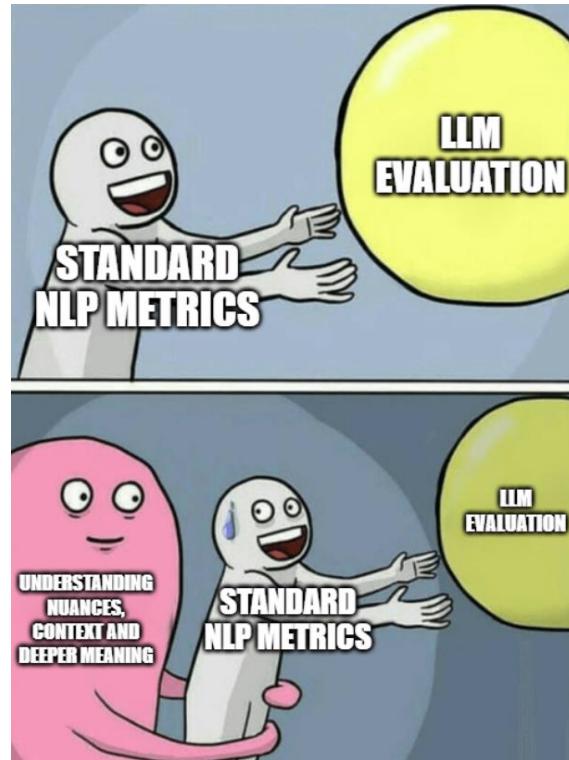
How can LLMs be evaluated

Traditional Natural Language Processing (NLP) evaluation

Traditional NLP metrics often miss the subtleties of language, like context and deeper meaning, which can result in an incomplete picture of how well a model actually performs.

Table 2.3: Common NLP Metrics and Their Formulas

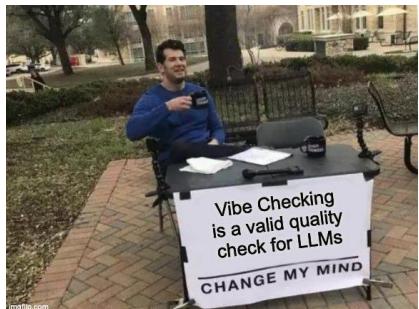
Metric	Formula
Accuracy	$\frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$
Recall	$\frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$
F1-Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
BLEU Score	$\exp \left(\min \left(1 - \frac{\text{Length of reference}}{\text{Length of candidate}}, 0 \right) \right) \prod_{n=1}^N (\text{Precision}_n)^w_n$
ROUGE Score	$\frac{\sum_{\text{match} \in \text{hypothesis}} \text{Count}(\text{match})}{\sum_{\text{n-gram} \in \text{reference}} \text{Count}(\text{n-gram})}$
Perplexity	$2^{-\frac{1}{N} \sum_{i=1}^N \log_2 P(w_i)}$
Word Error Rate (WER)	$\frac{\text{Substitutions} + \text{Insertions} + \text{Deletions}}{\text{Total words spoken}}$
METEOR ¹	$10 \times \frac{\text{Precision} \times \text{Recall}}{\alpha \times \text{Precision} + (1-\alpha) \times \text{Recall}}$



How can LLMs be evaluated

Human Evaluation vs Automatic Evaluation (LLM as judge)

Compared to human evaluation, which can be expensive, biased, and slow, the **LLM-as-judge** approach involves using large language models to **autonomously** assess and score outputs based on **predefined criteria**, offering a more efficient, scalable, and consistent alternative.



Chatbot Arena LLM Leaderboard

Backed by over 1,000,000+ community votes, our platform ranks the best LLM and AI chatbots. Explore the top AI models on our LLM leaderboard!

Chat now!

Expand to see the descriptions of 74 models

Model A

Can you explain to me which tasks the benchmark MT-Swag-QA assesses and who the authors are?

Model B

Can you explain to me which tasks the benchmark MT-Swag-QA assesses and who the authors are?

Tasks Assessed by MT-Swag-QA

The MT-Swag-QA benchmark is a multitask evaluation dataset designed to test a model's ability to perform various natural language understanding and reasoning tasks, particularly in a question-answering (QA) context. It draws from different existing NLP benchmarks and focuses on several core tasks:

- Commonsense Reasoning:**
 - This task evaluates the model's ability to make everyday inferences based on general knowledge. It is derived from the SWAG (Situations With Adversarial Generations) dataset. A context sentence is provided, and the model must choose the most plausible continuation from multiple options.
- Natural Language Inference (NLI):**
 - NLI tasks assess the model's ability to determine logical relationships between two sentences. The possible relationships include entailment, contradiction, and neutral.
- Question Answering (QA):**
 - This task tests the model's ability to answer questions based on a provided context or background text. The model is expected to extract or infer the correct answers from the given information.
- Abductive Reasoning:**
 - In this task, the model is asked to choose the most likely explanation for a set of observations or events. It involves reasoning about what might have caused a particular outcome or sequence of events.

MT-Swag-QA is a specialized benchmark designed to evaluate the capabilities of natural language understanding models, particularly focusing on multitask aspects of commonsense reasoning, situational understanding, and question answering (QA). This benchmark builds upon the foundations of the original SWAG (Situations With Adversarial Generations) dataset, extending its scope to encompass a broader range of tasks and complexities.

Tasks Assessed by MT-Swag-QA

- Commonsense Reasoning:**
 - Objective:** Assess the model's ability to apply everyday knowledge to infer plausible scenarios and outcomes.
 - Example Task:** Given a partial sentence or situation, the model predicts the most likely continuation based on common sense.
- Multiple-Choice Question Answering (MC-QA):**
 - Objective:** Evaluate the model's proficiency in selecting the correct answer from a set of predefined options.
 - Example Task:** Presenting a question with four possible answers, where the model must identify the most appropriate choice based on the context.
- Situational Understanding:**
 - Objective:** Test the model's capability to comprehend and interpret the nuances of specific situations or contexts presented within the questions.
 - Example Task:** Understanding the context of a dialogue or narrative to answer questions that require grasping the underlying situation.

A is better B is better Tie Both are bad

Enter your prompt and press ENTER

Random Image New Round Regenerate Share

How can LLMs be evaluated

Human Evaluation vs Automatic Evaluation (LLM as judge)

LLM-as-judge is the most promising evaluation method due to its flexibility, scalability, and ability to assess models across diverse tasks.

Question: If the FED buys bonds in the secondary market
(A) the money supply will increase. (B) the money supply will decrease. (C) the money supply will not be affected. (D) the discount rate would be affected. Output the answer only.

Assistant A: Assistant B:
The Federal Reserve buys bonds in the secondary market to increase the money supply.

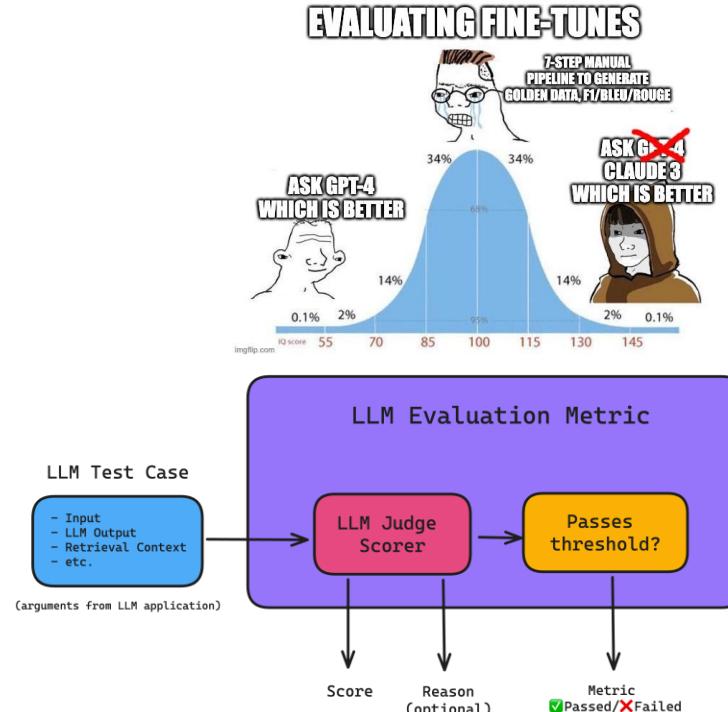
Follow-up Question: How does it affect my daily life? Give 3 examples.

Assistant A: Assistant B:
When the Federal Reserve buys bonds in the secondary market, it can have several effects on the economy that can impact your daily life. Here are three examples:

1. Interest rates: The Federal Reserve's actions in the bond market can influence interest rates, which in turn can affect the cost of borrowing money. For example, if the Federal Reserve buys bonds and drives up demand for them, interest rates on things like mortgages, car loans, and credit cards may go up, making it more expensive to borrow money.
2. Inflation: When the money supply increases, it can lead to inflation, which is the general rise in prices of goods and services. This can impact your daily life by making it more expensive to buy things like groceries, gas, and other necessities.
3. Employment: The Federal Reserve's actions in the bond market can also have an impact on employment. For example, if the Federal Reserve's actions lead to economic growth and increased demand for goods and services, it can create more job opportunities and boost employment.

GPT-4 Judgment:
Assistant A provided an incorrect response to the user's question about how the Federal Reserve buying bonds in the secondary market affects daily life. The answer given is repetitive and lacks clear examples of how the action impacts daily life.
Assistant B provided relevant and accurate responses to the user's question about the Federal Reserve buying bonds. The answer includes three clear examples of how the action impacts daily life, such as interest rates, inflation, and employment.

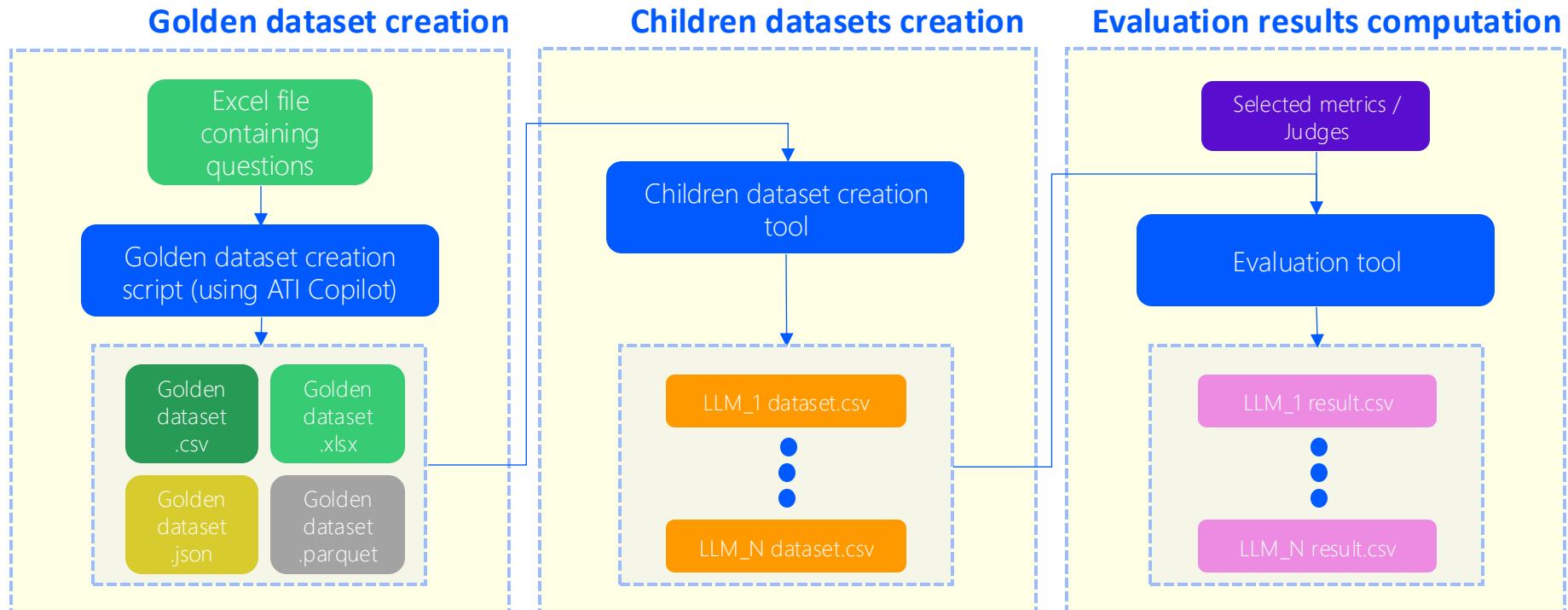
Assistant A should improve by providing relevant, clear, and accurate examples to address the user's question. Assistant B's response is more helpful, relevant, and detailed.



Proposed evaluation tool

Proposed evaluation tool

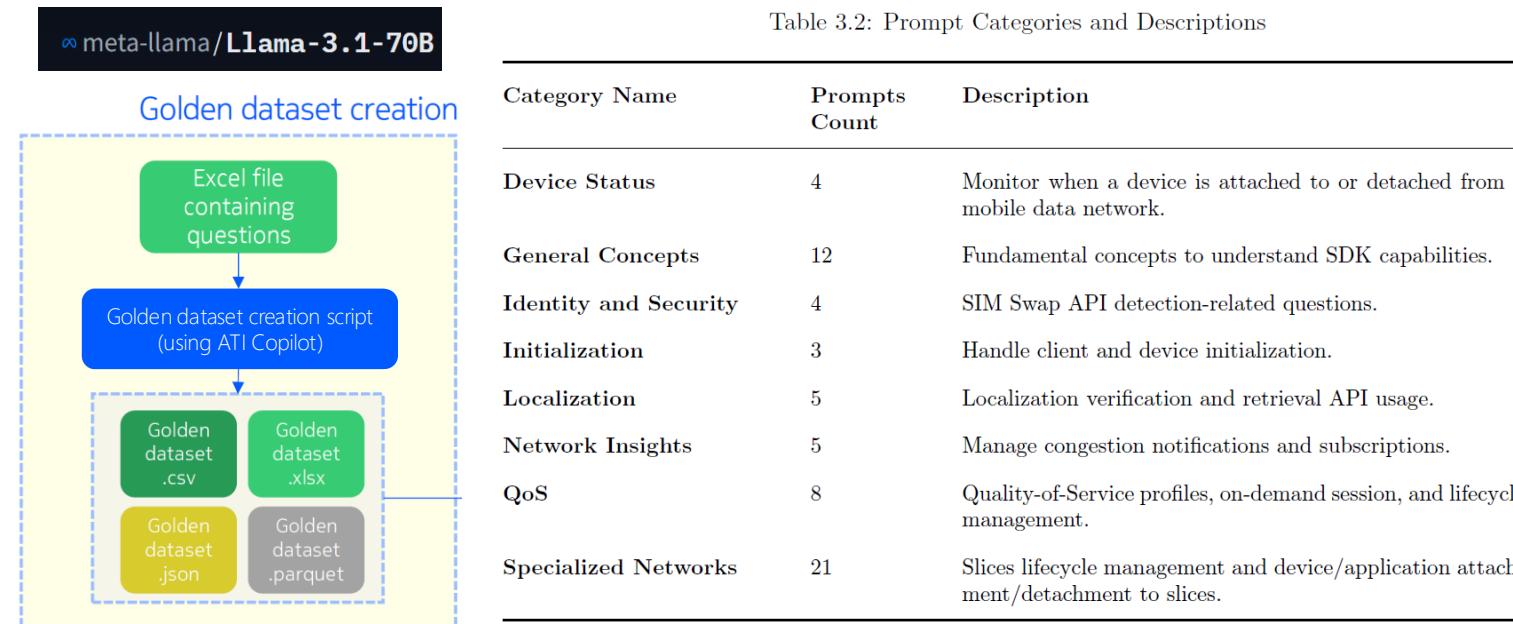
Pipeline



Proposed evaluation tool

Step 1: Golden dataset generation

Golden dataset contains a set of 62 questions and answers among 8 different categories.



Proposed evaluation tool

Step 1: Golden dataset generation

How the LLM must behave + extra context from knowledge base

	Id	System Prompt	User Prompt	LLM Answer	Category	Execute code	Model Name
0	installation	// The assistant is specialized for network installations.	How can I install NaC Py?	```bash pip install network_a```\nInitialization	Initialization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
1	device identification	// The assistant is specialized for device identification.	In which ways can I identify a device?	**Parameter Name** **Detail**	General concepts	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
2	scopes list	// The assistant is specialized for listing scopes.	Can you tell me all the available scopes?	### Device Status scopes **Scopes**	General concepts	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
3	qod session list	// The assistant is specialized for QoS session management.	Can you list all of the available QoS sessions?	### Network as Code QoS profiles **QoS**	QoS	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
4	qod life cycle	// The assistant is specialized for QoS life cycle.	Can you display the life cycle of a QoS session?	# QoD Session Life Cycle A QoS	QoS	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
5	nac localization capabilities	// The assistant is specialized for NAC localization.	Which localization features does NAC support?	**Feature** **Description**	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
6	location parameter info	// The assistant is specialized for location parameters.	What is the maximum age parameter?	The maximum age parameter is 30 minutes.	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
7	location parameter default values	// The assistant is specialized for location parameter defaults.	In case maximum age parameter is not specified, what is the default value?	If the maximum age parameter is not specified, the default value is 30 minutes.	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
8	slice definition	// The assistant is specialized for slice definitions.	What is a Specialized Network?	A Specialized Network, also known as a Slice, is a	Specialized Networks	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
9	slice id	// The assistant is specialized for slice identifiers.	Can you provide some context about Network Identifier?	A Network Identifier is a unique identifier for a	Specialized Networks	NO	meta-llama/Meta-Llama-3.1-70B-Instruc

Proposed evaluation tool

Step 1: Golden dataset generation

The screenshot shows a user interface for generating a golden dataset. At the top, there's a dropdown menu labeled "Select a conversation to visualize" with the value "15". To the right of the dropdown are two buttons: "Previous" (highlighted with a red border) and "Next". Below the dropdown is a checkbox labeled "Show system prompt".

The main content area displays the title "Index: 15 - Title: congestion level parameters". Below the title is a message card with a clock icon and the text: "Which are all the congestion notification parameters? Provide a Markdown table with two columns (parameter, description) without including any code snippets."

Below the message card is a table with the following data:

Parameter	Description
<code>device</code>	Information about the device for which congestion data is requested or subscribed to.
<code>id</code>	It represents the subscription identifier.
<code>subscription_expire_time</code>	Expiration time for the congestion subscription (using ISO 8601 format)
<code>notification_url</code>	URL where notifications about congestion updates will be sent.
<code>notification_auth_token</code> (Optional)	Authentication token for accessing the notification URL.
<code>maxNumber0fReports</code> (Optional)	Specify an integer value for the maximum number of reports to be sent in the subscription.
<code>start</code>	Start timestamp for retrieving congestion data. It should be an ISO 8601 formatted date string or a date-time object as shown above.
<code>end</code>	End timestamp for retrieving congestion data. It should be an ISO 8601 formatted date string or a date-time object as shown above.

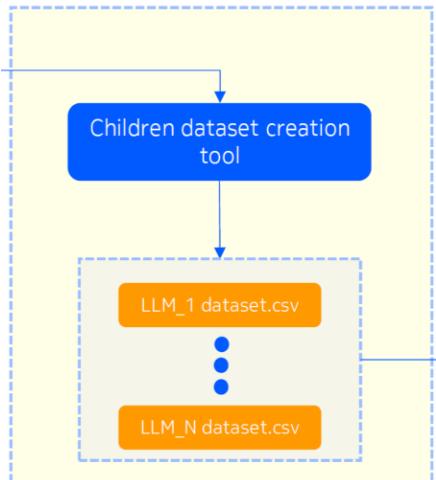
Proposed evaluation tool

Step 2: Generation of child datasets

Derivate datasets (*children*) can be generated from the reference one defining the available LLMs using a *YAML* configuration. They are the datasets that will be evaluated against the golden dataset. 18 LLMs under test.

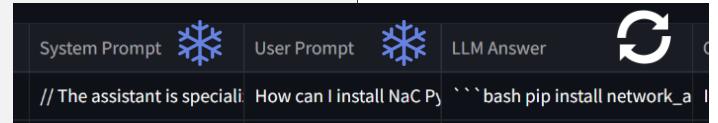
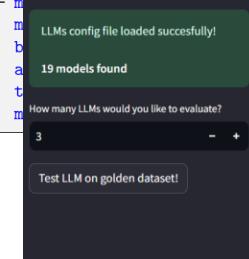
To generate one, we “*freeze*” the *system prompt* and the *user query* of the reference dataset to obtain a new LLM answer

Children datasets creation



Listing 3.2: Example LLM Configuration File

```
models:  
- model: "llama3"  
  model_full_name: "Llama 3 70B"  
  base_url: "http://localhost:11434/v1/"  
  api_key: "none"  
  temperature: 0.1  
  max_tokens: 2048
```



Run batch inference jobs!

Llama 3.1 8B

Choose the LLM you want to evaluate

llama3.1

Mixtral 8x7B

Choose the LLM you want to evaluate

mixtral

Qwen2.5 7.6B

Choose the LLM you want to evaluate

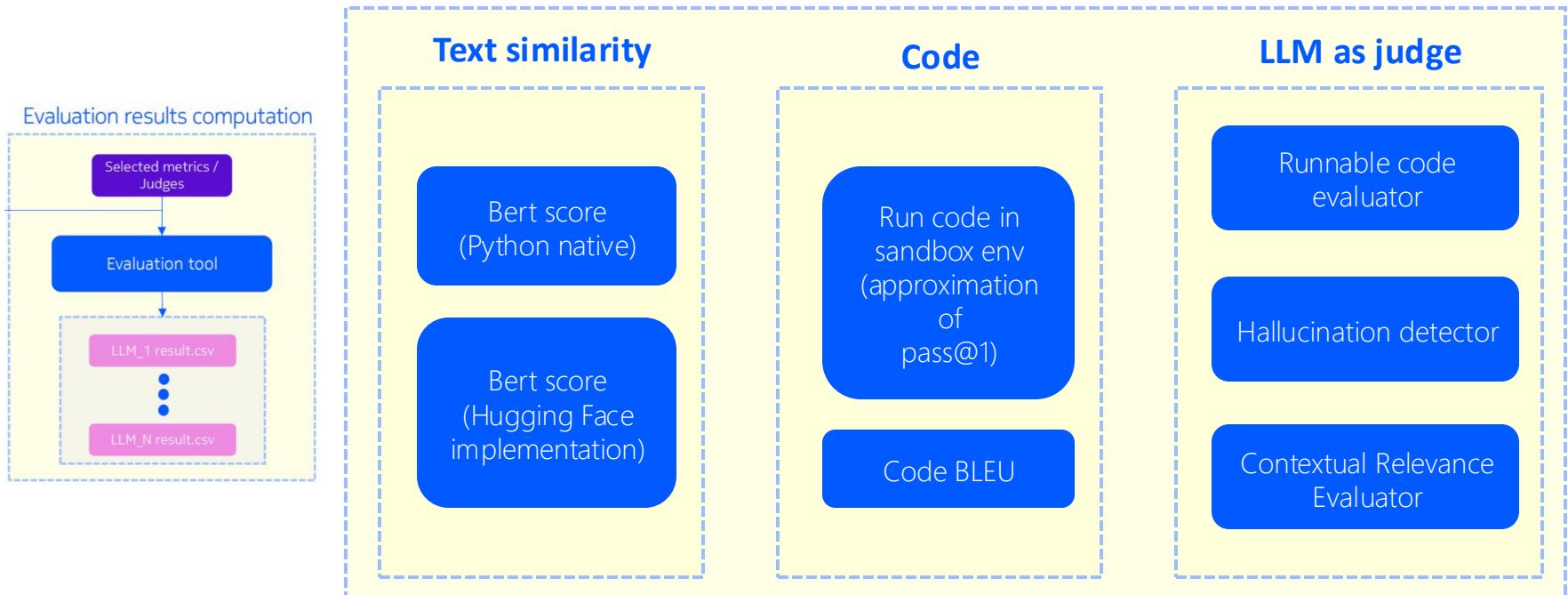
qwen2.5

Proposed evaluation tool

Step 3: Define evaluation methods

The evaluation has been structured in 3- easy to plug and play – “pillars”

NaC evaluation



Visualizations

Aggregate stats



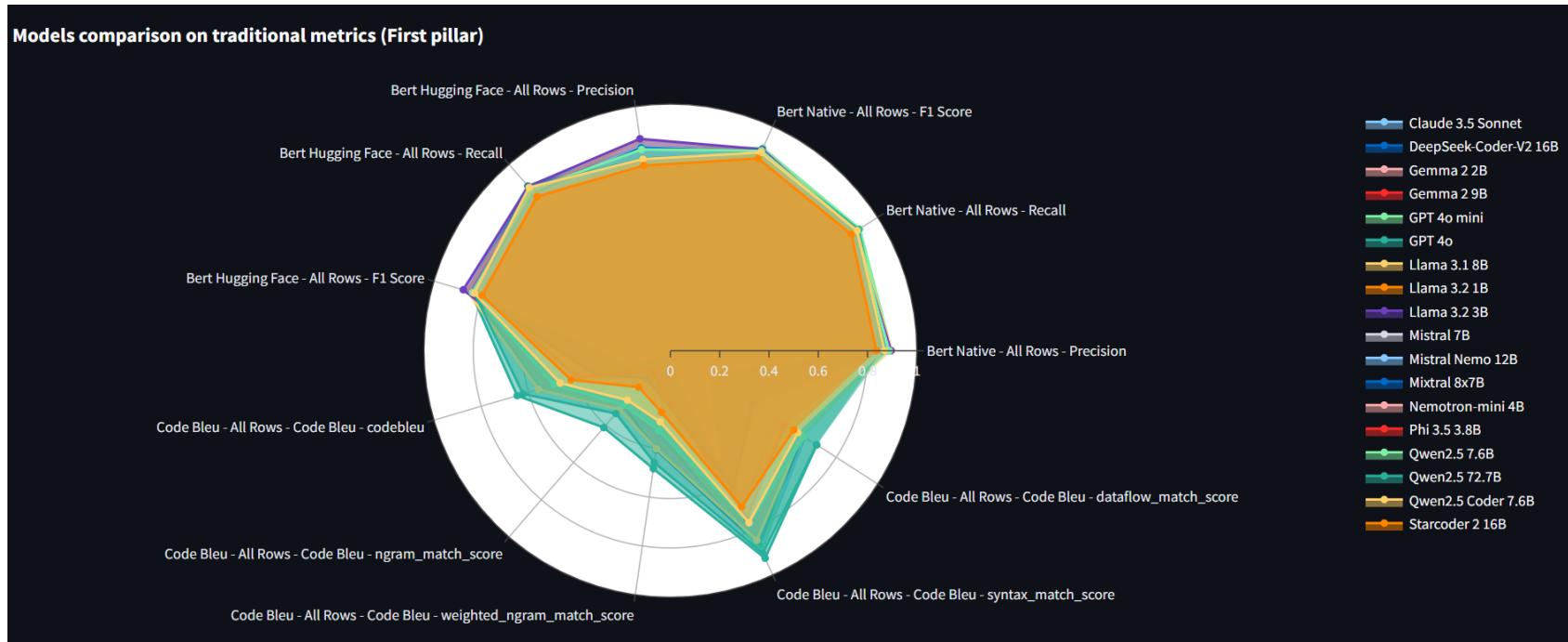
Visualizations

Radar plots



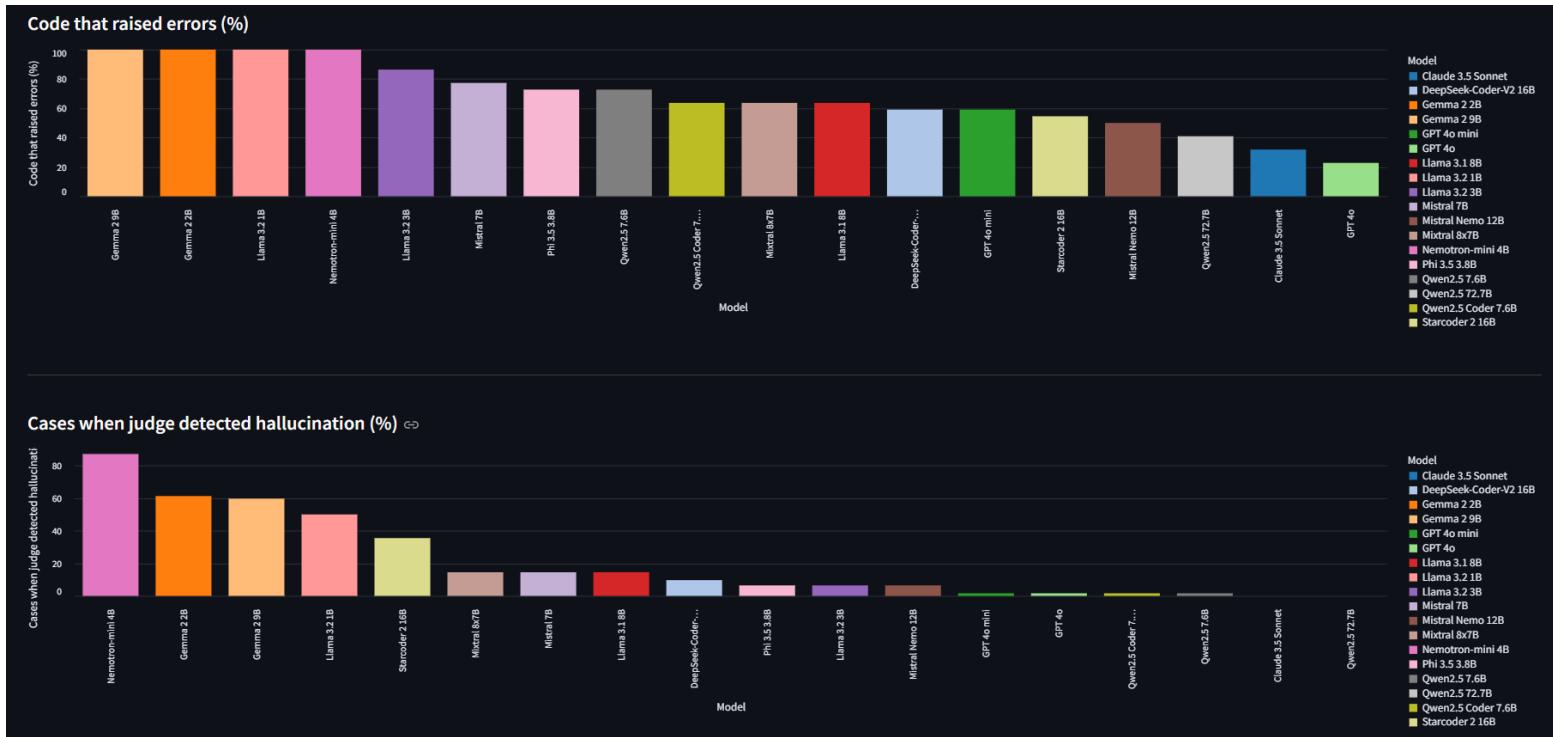
Visualizations

Models compared (first pillar)



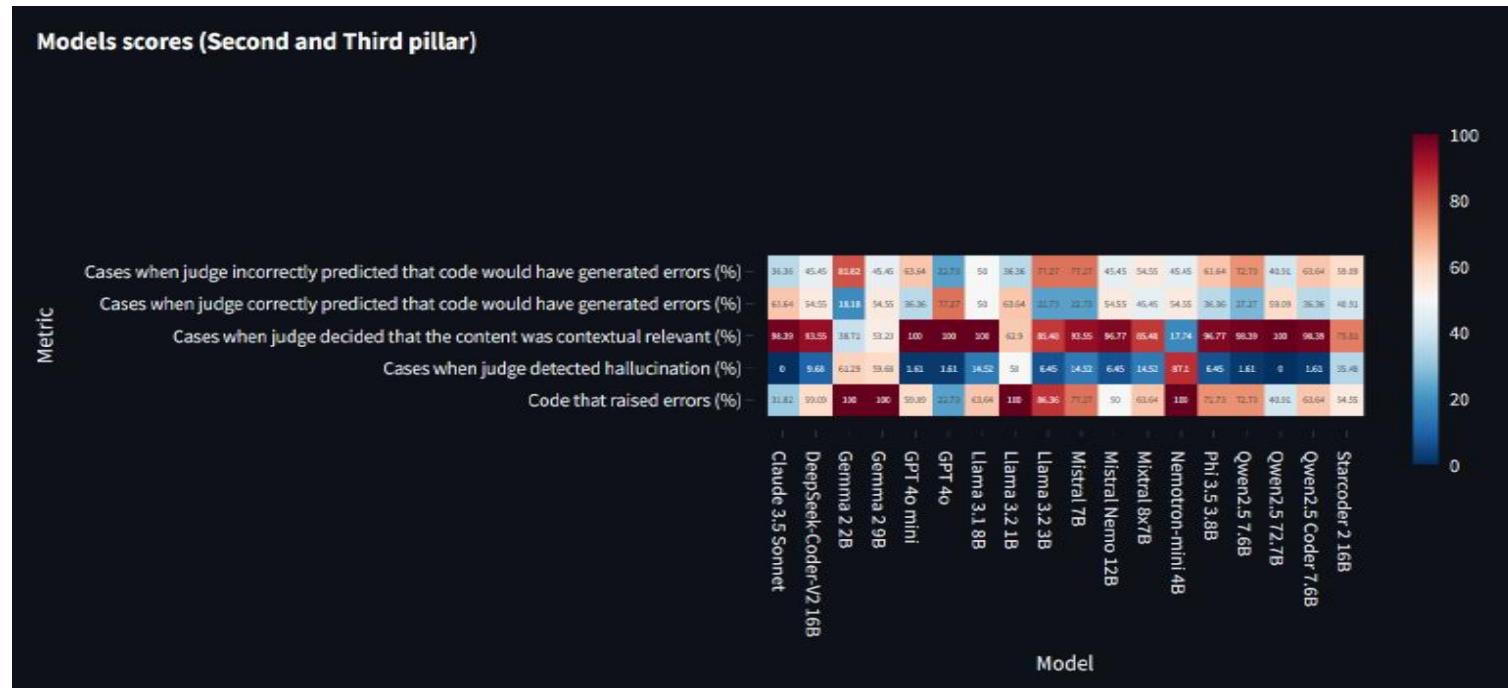
Visualizations

Models compared



Visualizations

Models compared



Results

Results

Aggregate results

- Average **code error** rate is notably high (67.7%, while median is 63.6%), with a std of 23.5% (range 22.7% to 100%)
- Despite high scores, Bert score often **rates non-functional code nearly as highly as working code**, highlighting its unreliability as a quality indicator (consistently exceeds 0.8 with low std).
- CodeBLEU** may appear relatively low, however syntax and dataflow match scores are high. This shows that models **prioritize syntax consistency** over data structures, a key achievement since adherence to syntax is crucial when using an SDK.

Table 4.5: Code Bleu Statistics (Value range: [0.000,1.000])

Model	Code Bleu	Ngram Match	Weighted Ngram	Syntax Match Score	Dataflow Match Score
Mean	0.429	0.202	0.253	0.735	0.518
Std	0.144	0.121	0.143	0.139	0.178
Min	0.172	0.0045	0.0115	0.47	0.172
25%	0.347	0.122	0.178	0.623	0.448
50%	0.432	0.1965	0.24725	0.7575	0.58
75%	0.552	0.293625	0.336375	0.8615	0.637125
Max	0.648	0.413	0.484	0.9255	0.706

Table 4.3: Bert Native Statistics (Value range: [0.000,1.000])

Model	Bert Precision	Native Recall	Bert F1 Score
Mean	0.868	0.889	0.878
Std	0.022	0.016	0.017
Min	0.820	0.863	0.841
25%	0.852	0.874	0.866
50%	0.870	0.892	0.883
75%	0.886	0.900	0.889
Max	0.899	0.914	0.904

Model	Scripts that raised errors (%)
Mean	67.678
Std	23.483
Min	22.730
25%	55.685
50%	63.640
75%	84.088
Max	100.000



Print
to debug

Just
execute the code
and hope it
doesn't break

Results

Aggregate results

- Higher CodeBLEU score is strongly **negative correlated** with presence of code errors (lower likelihood of bugs)
- Judges are **unreliable** on predicting if a code snippet will generate an error or not (47.7% of accurate predictions)
- Judge ability of detecting hallucination is **not necessarily linked** to the **accuracy of error prediction** (because of point previous reason). Negligible correlation (0.01)
- Positive correlation** between hallucination detection and code snippets with errors (0.717) Most of the LLM struggled with **asynchronous code generation**

Table 4.2: Error and Hallucination Statistics (Value range: [0.000,100.000])

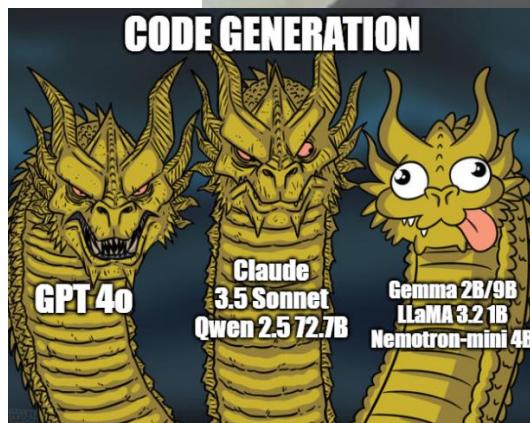
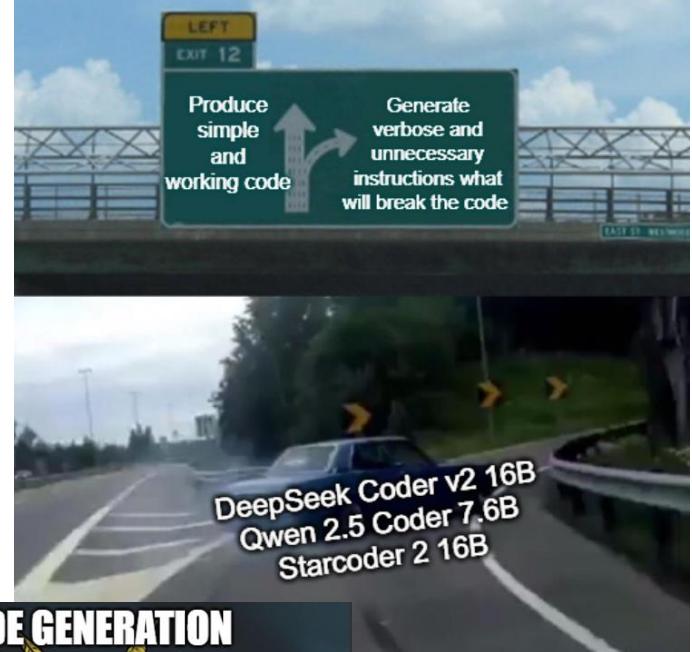
Model	Scripts that raised errors (%)	Judge-detected hallucinations (%)	Judge-validated relevance (%)	Judge correct error predictions (%)	Judge incorrect error predictions (%)
Mean	67.678	20.699	83.064	45.455	54.545
Std	23.483	26.360	24.306	16.427	16.427
Min	22.730	0.000	17.740	18.180	22.730
25%	55.685	1.610	78.228	36.360	45.450
50%	63.640	8.065	95.160	47.725	52.275
75%	84.088	30.240	98.390	54.550	63.640
Max	100.000	87.100	100.000	77.270	81.820



Results

Aggregate results

- Code-specialized LLMs (*DeepSeekCoder v2 16B* and *Starcoder 2 16B*) tended to hallucinate more frequently (excessive “divide-and-conquer” that leads to unexpected output). Verbose and overcomplicated code.
- Most common issue in code generation was incorrect parameters handling in API calls, which led to failures on remote servers.
- *GPT-4o* demonstrated the **lowest error rate** in code generation, at 22.73%, followed by *Claude 3.5 Sonnet* (31.82%) and *Qwen 2.5 72.7B* (40.91%).
- Conversely, models like *Gemma 2B*, *Gemma 9B*, *LLaMA 3.2 1B*, and *Nemotron-mini 4B* exhibited a **100% error rate**.



NOKIA

Limitation and future works

Limitations and future works

Limitations

- Closed source models (GPT 4o, 4o mini, Claude 3.5) had limited token usage, which in some way capped their potential
- Could not use the full context window of each LLM (because of hardware and budget constraints).
For LLAMA just 10% of it has been used (the maximum context window is 132k token)
- The code generation quality could be improved by **adding to the knowledge base** also the NaC SDK codebase instead of using just the Markdown documentation (docstring and async code issues)
- Too few code examples in the official documentation



LLMs



LLMs (capped)

Limitations and future works

Future works

- Involve domain experts in the evaluation (human in the loop) because of their capabilities to detect nuances in the code generation and documentation explanation.
- Create a golden dataset using **human annotators**
- Improve over time LLM answers incorporating **A/B testing** for users in the pipeline
- Explore other **Prompt Engineering** techniques (current system prompt is too verbose)
- Explore **RAG hyperparameter tuning** rather than focusing only on tweaking the final LLM output
- Use LLM as judge output to improve the answer of the LLM (**refine with second iteration**)



Thanks for the attention!



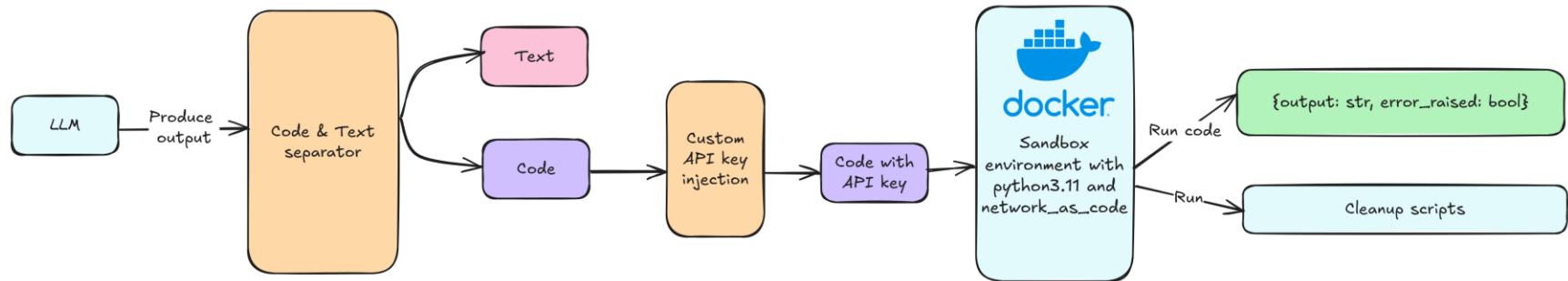
Kiitos!

Extra slides

Evaluation methods

Proposed evaluation tool

Sandbox environment code evaluation



Proposed evaluation tool

Example of Judge prompt

```
judges:
- judge_name: Runnable code evaluator
  instructions: |
    <instructions>
    You are an impartial judge that has to evaluate the correctness of a Large Language Model generated code.
    You have to understand the code and decide if the code will generate an error or not. Consider that the user will have already installed the network_as_code Python library. Please ignore any error that could be related to absence or wrong token.
    You are provided with some snippets of code retrieved from a vector database, the user query and the answer provided by the LLM.
    Please answer only with a Markdown JSON format containing a boolean value that indicated if the code generate errors and the reason why you took that decision.
    Here's an example:
    '''json
    {"error_will_be_raised": true, "reason": "An error will be raised because ..."}
    '''
    or
    '''json
    {"error_will_be_raised": false, "reason": "An error wont' be raised because ..."}
    '''
  </instructions>
  ---
  Here's the context:
  <user_query>
  {user_query}
  </user_query>

  <context_retrieved>
  {context_retrieved}
  </context_retrieved>

  content_to_judge: |
    <llm_answer>
    {llm_answer}
    </llm_answer>
  ---
  Remember to only provide the answer in json format without any preamble or introductory text.
```

Metrics available

- evaluate code
- code bleu
- bert score
- hf bert score
- llm as judge

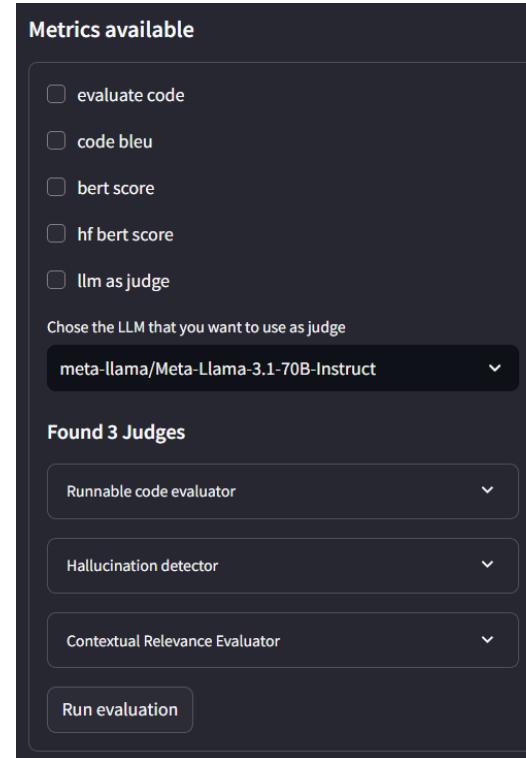
Chose the LLM that you want to use as judge

meta-llama/Meta-Llama-3.1-70B-Instruct

Found 3 Judges

- Runnable code evaluator
- Hallucination detector
- Contextual Relevance Evaluator

Run evaluation

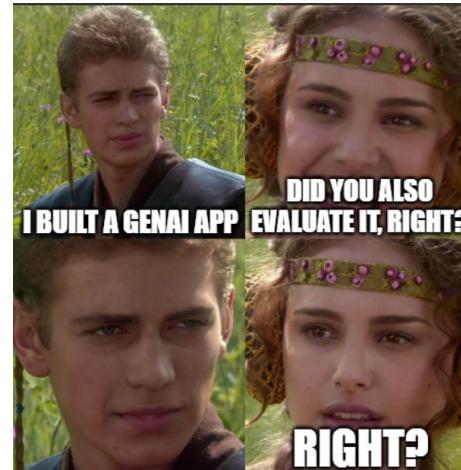
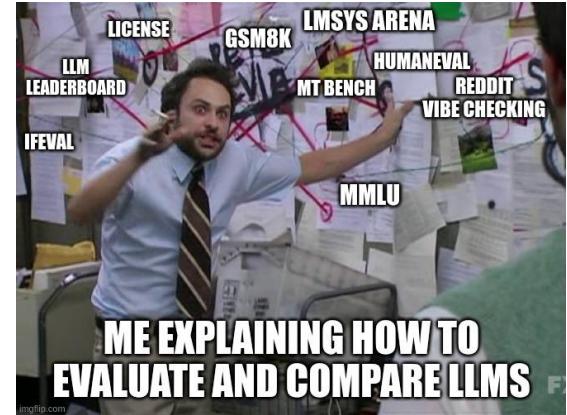
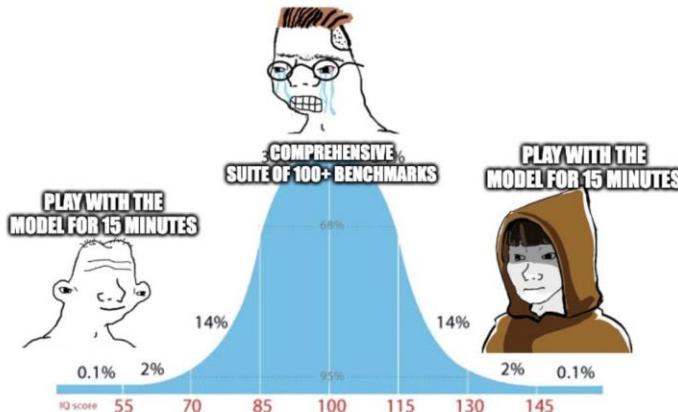


Why having a LLM Benchmark?

Key concepts

Why do you need an evaluation Benchmark?

- Easy and structured way to compare different models on the same task (no vibe checking)
- Easier to spot LLM errors or common pitfalls (some LLMs fail on a specific task in a specific way)
- You need to validate your GenAI app in some way!
- Forces you to define requirements (what is or not a good behavior)



Setup (HW+SW) and Models

Setup

Hardware and Software used

Table 3.3: Key Tools Used for Each Task

Task	Tool
Reference LLM inference for golden dataset generation	vLLM ¹ (hosted on a local server with a dedicated NVIDIA H100 GPU) ²
Inference of open-source LLMs	Ollama ³ (hosted on a local server with a dedicated NVIDIA A40 GPU) ⁴
Inference of closed-source LLMs (e.g., GPT-4o, GPT-4o mini, Claude 3.5 Sonnet)	Azure AI ⁵
Frontend development for the evaluation application	Streamlit ⁶

Internal Copilot

Hyperparameters used

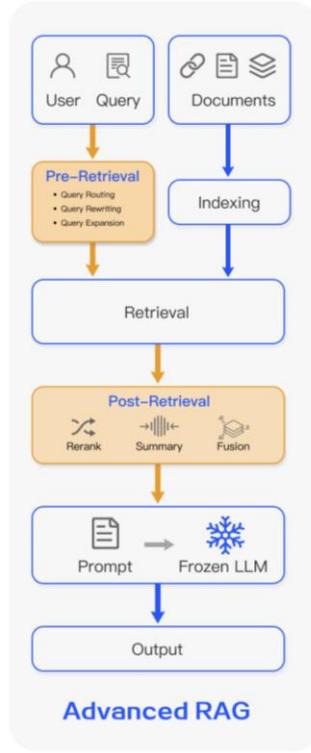


Table 3.1: RAG Pipeline Hyperparameters with choice justification

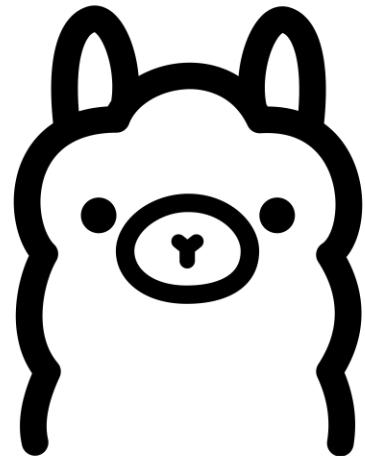
Hyperparameter	Value	Reason
Temperature	0.1	Limits LLM creativity and enforces reproducibility, aiming for predictable answers without making them entirely robotic.
Database choice	Milvus	Supports both sparse and dense vector representation: sparse for lexical (keyword, term) results, and dense for semantic (sentence, meaning) results.
Embedding model	BAAI/bge-m3	High-quality multilingual embedding model that can produce both sparse and dense embeddings, with long context handling for large text chunks.
Reranker model	BAAI/bge-reranker-v2-m3	Corresponding reranker model to sort semantic results by relevance with the user query.
Minimum semantic relevancy	0.01	Experimental cut-off point for low-relevance semantic results. Document chunks with lower scores are not sent to the LLM as knowledge.
Large Language Model	meta-llama/Meta-Llama-3.1-70B-Instruct	Provides the best available cost/quality tradeoff.
LLM output max tokens	2048	A 2048-token limit (approx. 8000 characters) typically fits retrieved Q and A content in RAG; 4096 or 8192 tokens may be better for long-form content generation.
Max chunk size [chars]	6000	Targets chunking one document section (header + content) or table per chunk. Small sections/tables can be concatenated.
Max knowledge size [chars]	60000	Best tradeoff between inference time, performance, and quality, leaving room for conversation history within a 32k or 64k context limit.
Chunks overlap	False	No need for redundant text in chunks, as chunks are split by document sections (header + content or entire table).
Query reformulation (surrogate query)	True	Rewriting the user query enhances search effectiveness by targeting semantic, lexical, or hybrid queries optimized for the database query engine.

LLMS

Opensource models (Ollama)

Table 4.1: Specifications of Evaluated Open Source LLMs

Model Name	Number of Parameters	Context Length	Num Ctx	Embedding Length	Quantization Technique
llama3.2	3.0B	131072	16000	3072	Q4_K_M
llama3.2:1b	1.0B	131072	16000	2048	Q8_0
llama3.1	8.0B	131072	16000	4096	Q4_0
qwen2.5	7.6B	32768	16000	3584	Q4_K_M
qwen2.5:72b	72.7B	32768	16000	8192	Q4_K_M
qwen2.5-coder	7.6B	32768	16000	3584	Q4_K_M
mixtral	46.7B	32768	16000	4096	Q4_0
mistral	7.2B	32768	16000	4096	Q4_0
mistral-nemo	12.2B	131072	16000	5120	Q4_0
nemotron-mini	4.2B	4096	16000	3072	Q4_K_M
gemma2	9.2B	8192	8192	3584	Q4_0
gemma2:2b	2.6B	8192	8192	2304	Q4_0
phi3.5	3.8B	131072	16000	3072	Q8_0
starcoder2:instruct	16.0B	16384	16000	6144	Q4_0
deepseek-coder-v2	15.7B	163840	16000	2048	Q4_0



NLP metrics (in depth)

Standard NLP metrics

Not suitable for LLM evaluation

LLM-based NLG Evaluation: Current Status and Challenges

Mingqi Gao*, Xinyu Hu*, Jie Ruan, Xiao Pu, Xiaojun Wan

Peking University

{gaomingqi, huxinyu, wanxiaojun}@pku.edu.cn, {ruanjie, puxiao}@stu.pku.edu.cn

Evaluating natural language generation (NLG) is a key but challenging issue. Traditional evaluation metrics like BLEU Papineni et al. (2002) and ROUGE Lin (2004) rely on the n-gram overlap between model outputs and references to measure its quality. They have been criticized for low correlation with human judgments Sulem et al. (2018), as surface-level matching cannot reliably evaluate text. After the rise of deep learning, model-based evaluation metrics like BERTScore Zhang et al. (2020) and BARTScore Yuan et al. (2021) have been continuously proposed and gradually adopted to evaluate the overall quality or various specific aspects of generated outputs (e.g., fluency, coherence, coverage, faithfulness, etc.). Although better than traditional metrics, their performance is still not satisfactory, and their application scope is very limited. For example,

CodeBLEU

An extension of BLEU (used in machine translation evaluation)

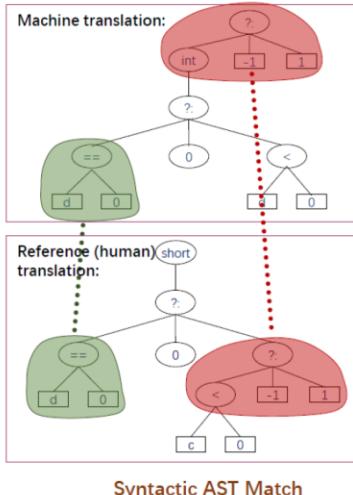
Machine translation:

```
public static int Sign ( double d )
{
    return (int)(( d == 0 )? 0:( d < 0 )));
    -1: 1;
}
```

1.0 1.0 0.7 0.5

Reference (human) translation:

```
public static short Sign ( double d )
{
    return (short)(( d == 0 )? 0:( d < 0 ));
    -1: 1;
}
```



Machine translation:

```
public static int Sign ( double d )
{
    return (int)(( d == 0 )? 0:( d < 0 )));
    -1: 1;
}
```

Reference (human) translation:

```
public static short Sign ( double c )
{
    return (short)(( c == 0 )? 0:( c < 0 ));
    -1: 1;
}
```

Semantic Data-flow Match

[(d', 7, 'comesFrom', [], []),
(d', 16, 'comesFrom', [d'], [7]),
(d', 24, 'comesFrom', [d'], [7])]

Abstract Syntax Tree (AST)
Nodes = construct in the code
Leaves = names of the functions and all the variables

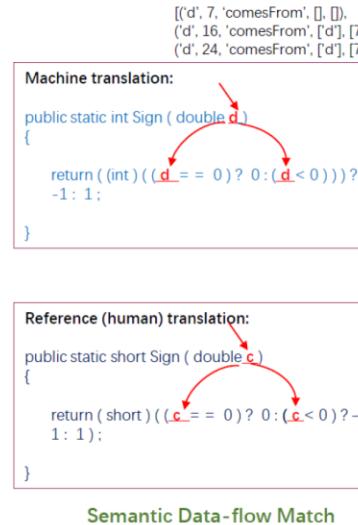
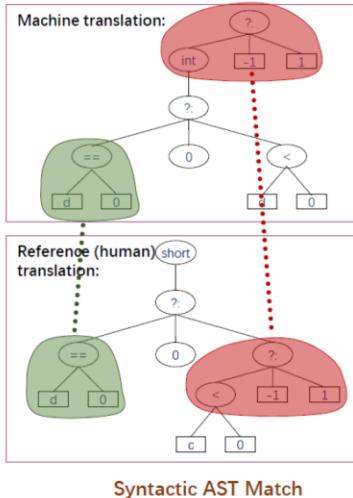
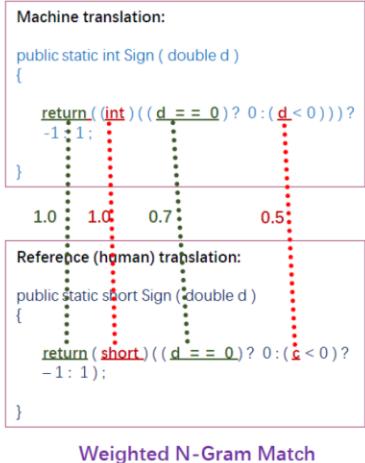
As shown in the middle part of Figure 1, we extract all the sub-trees of the candidate and the reference ASTs respectively. Then we calculate the syntactic AST match score as:

$$\text{Match}_{\text{ast}} = \text{Count}_{\text{clip}}(\text{T}_{\text{cand}}) / \text{Count}(\text{T}_{\text{ref}}) \quad (4)$$

where $\text{Count}(\text{T}_{\text{ref}})$ is the total number of the reference subtrees, and $\text{Count}_{\text{clip}}(\text{T}_{\text{cand}})$ is the number of the candidate subtrees that are matched the reference. This score can evaluate code quality from a syntactic perspective, because grammatical errors such as token missing, data type errors can be captured by the difference between their ASTs.

CodeBLEU

An extension of BLEU (used in machine translation evaluation)



Semantic Data-flow match
Code represented as graph
(nodes = vars, edges = where value comes from)

Based on the above, there are three steps to compute the semantic data-flow match score.

Step 1: Obtain the data-flow graphs for the candidate and the reference. Based on AST, we first utilize the leaves to identify variable sequence, denoted as $V = \{v_0, v_1, \dots, v_m\}$. We then take each variable as a node of the graph and a directed edge $e = \langle v_i, v_j \rangle$ from v_i to v_j refers that the value of j -th variable comes from i -th variable. The graph $\mathcal{G}(C) = (V; E)$ is used to represent relations among variables of the code C , as shown by the red arrows in Figure 1.

Step 2: Normalize data-flow items. For simplicity and unity, we ignore the variable position and normalize their names. We collect all the variables in the data-flow items and rename them var_i , where i is the order of the variables appearing in all data-flow items.

Step 3: Calculate the semantic data-flow match score as:

$$\text{Match}_{df} = \text{Count}_{clip}(\text{DF}_{\text{cand}}) / \text{Count}(\text{DF}_{\text{ref}}) \quad (5)$$

where $\text{Count}(\text{DF}_{\text{ref}})$ is the total number of the reference data-flows, and $\text{Count}_{clip}(\text{DF}_{\text{cand}})$ is the number of

$$\text{CodeBLEU} = \alpha \cdot \text{N-Gram Match (BLEU)} + \beta \cdot \text{Weighted N-Gram Match} + \gamma \cdot \text{Syntactic AST Match} + \delta \cdot \text{Semantic Data-flow Match}$$

Tables with all results

Results

Judges

Table 4.2: Error and Hallucination Statistics (Value range: [0.000,100.000])

Model	Scripts that raised errors (%)	Judge-detected hallucinations (%)	Judge-validated relevance (%)	Judge correct predictions (%)	Judge error predictions (%)
Mean	67.678	20.699	83.064	45.455	54.545
Std	23.483	26.360	24.306	16.427	16.427
Min	22.730	0.000	17.740	18.180	22.730
25%	55.685	1.610	78.228	36.360	45.450
50%	63.640	8.065	95.160	47.725	52.275
75%	84.088	30.240	98.390	54.550	63.640
Max	100.000	87.100	100.000	77.270	81.820

Results

NLP metrics

Table 4.3: Bert Native Statistics (Value range: [0.000,1.000])

Model	Bert Precision	Native Recall	Bert F1 Score
Mean	0.868	0.889	0.878
Std	0.022	0.016	0.017
Min	0.820	0.863	0.841
25%	0.852	0.874	0.866
50%	0.870	0.892	0.883
75%	0.886	0.900	0.889
Max	0.899	0.914	0.904

Table 4.4: Bert Hugging Face Statistics (Value range: [0.000,1.000])

Model	Bert Precision	HF Recall	Bert HF Re- call	Bert HF F1 Score
Mean	0.807	0.852	0.826	0.826
Std	0.034	0.025	0.027	0.027
Min	0.747	0.81	0.78	0.78
25%	0.782	0.829	0.806	0.806
50%	0.81	0.856	0.830	0.830
75%	0.833	0.875	0.840	0.840
Max	0.868	0.885	0.876	0.876

Table 4.5: Code Bleu Statistics (Value range: [0.000,1.000])

Model	Code Bleu	Ngram Match	Weighted Ngram	Syntax Match Score	Dataflow Match Score
Mean	0.429	0.202	0.253	0.735	0.518
Std	0.144	0.121	0.143	0.139	0.178
Min	0.172	0.0045	0.0115	0.47	0.172
25%	0.347	0.122	0.178	0.623	0.448
50%	0.432	0.1965	0.24725	0.7575	0.58
75%	0.552	0.293625	0.336375	0.8615	0.637125
Max	0.648	0.413	0.484	0.9255	0.706

Results

Bert Score

Table 4.7: Bert Native Statistics (Value range: [0.000,1.000])

Model	Bert Precision	Native Recall	Bert Native F1 Score
Claude 3.5 Sonnet	0.868	0.906	0.882
DeepSeek-Coder-V2 16B	0.887	0.896	0.888
Gemma 2 2B	0.820	0.863	0.841
Gemma 2 9B	0.843	0.868	0.855
GPT 4o mini	0.897	0.914	0.904
GPT 4o	0.871	0.908	0.889
LLaMA 3.1 8B	0.899	0.897	0.897
LLaMA 3.2 1B	0.860	0.872	0.870
LLaMA 3.2 3B	0.895	0.892	0.898
Mistral 7B	0.873	0.893	0.883
Mistral Nemo 12B	0.888	0.892	0.884
Mixtral 8x7B	0.881	0.888	0.885
Nemotron-mini 4B	0.851	0.864	0.856
Phi 3.5 3.8B	0.849	0.878	0.864
Qwen2.5 7.6B	0.884	0.900	0.891
Qwen2.5 72.7B	0.859	0.902	0.879
Qwen2.5 Coder 7.6B	0.869	0.901	0.883
Starcoder 2 16B	0.835	0.873	0.857

Table 4.8: Bert Hugging Face Statistics (Value range: [0.000,1.000])

Model	Bert HF Precision	Bert HF Recall	Bert HF F1 Score
Claude 3.5 Sonnet	0.800	0.873	0.831
DeepSeek-Coder-V2 16B	0.833	0.865	0.841
Gemma 2 2B	0.747	0.818	0.780
Gemma 2 9B	0.767	0.815	0.787
GPT 4o mini	0.833	0.885	0.859
GPT 4o	0.812	0.883	0.844
LLaMA 3.1 8B	0.864	0.857	0.860
LLaMA 3.2 1B	0.808	0.828	0.811
LLaMA 3.2 3B	0.868	0.881	0.876
Mistral 7B	0.822	0.854	0.829
Mistral Nemo 12B	0.833	0.852	0.837
Mixtral 8x7B	0.831	0.846	0.831
Nemotron-mini 4B	0.781	0.810	0.795
Phi 3.5 3.8B	0.779	0.832	0.804
Qwen2.5 7.6B	0.825	0.863	0.835
Qwen2.5 72.7B	0.787	0.876	0.829
Qwen2.5 Coder 7.6B	0.785	0.875	0.829
Starcoder 2 16B	0.759	0.827	0.796

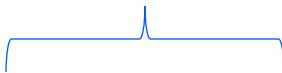
Results

Code BLEU

Code BLEU (aggregate)

Best: GPT 4o

Worst: Nemotron-mini 4B



Ngram match

Best: Phi 3.5 3.8B

Worst: Gemma 2 2B

Syntax match

Best: GPT 4o

Worst: Nemotron-mini 4B

Weighted Ngram match

Best: GPT 4o

Worst: Gemma 2 2B

Dataflow match

Best: GPT 4o, Qwen2.6 72.7B

Worst: Gemma 2 2B

Table 4.9: Code Bleu Statistics (Value range: [0.000,1.000])

Model	Code Bleu	Ngram Match	Weighted Ngram	Syntax Match Score	Dataflow Match Score
Claude 3.5 Sonnet	0.552	0.312	0.339	0.889	0.649
DeepSeek-Coder-V2 16B	0.551	0.294	0.406	0.867	0.638
Gemma 2 2B	0.190	0.012	0.020	0.534	0.172
Gemma 2 9B	0.201	0.005	0.012	0.595	0.179
GPT 4o mini	0.553	0.293	0.323	0.897	0.619
GPT 4o	0.648	0.413	0.484	0.926	0.706
LLaMA 3.1 8B	0.559	0.316	0.403	0.845	0.635
LLaMA 3.2 1B	0.333	0.132	0.175	0.579	0.426
LLaMA 3.2 3B	0.402	0.119	0.176	0.708	0.513
Mistral 7B	0.409	0.184	0.242	0.647	0.536
Mistral Nemo 12B	0.444	0.197	0.233	0.772	0.565
Mixtral 8x7B	0.326	0.150	0.195	0.615	0.396
Nemotron-mini 4B	0.172	0.015	0.026	0.470	0.175
Phi 3.5 3.8B	0.804	0.389	0.109	0.748	0.551
Qwen2.5 7.6B	0.493	0.291	0.329	0.768	0.647
Qwen2.5 72.7B	0.623	0.338	0.459	0.904	0.706
Qwen2.5 Coder 7.6B	0.467	0.266	0.292	0.767	0.615
Starcoder 2 16B	0.420	0.196	0.253	0.696	0.595

Results

Bert Score

Table 4.7: Bert Native Statistics (Value range: [0.000,1.000])

Model	Bert Precision	Native Recall	Bert Native F1 Score
Claude 3.5 Sonnet	0.868	0.906	0.882
DeepSeek-Coder-V2 16B	0.887	0.896	0.888
Gemma 2 2B	0.820	0.863	0.841
Gemma 2 9B	0.843	0.868	0.855
GPT 4o mini	0.897	0.914	0.904
GPT 4o	0.871	0.908	0.889
LLaMA 3.1 8B	0.899	0.897	0.897
LLaMA 3.2 1B	0.860	0.872	0.870
LLaMA 3.2 3B	0.895	0.892	0.898
Mistral 7B	0.873	0.893	0.883
Mistral Nemo 12B	0.888	0.892	0.884
Mixtral 8x7B	0.881	0.888	0.885
Nemotron-mini 4B	0.851	0.864	0.856
Phi 3.5 3.8B	0.849	0.878	0.864
Qwen2.5 7.6B	0.884	0.900	0.891
Qwen2.5 72.7B	0.859	0.902	0.879
Qwen2.5 Coder 7.6B	0.869	0.901	0.883
Starcoder 2 16B	0.835	0.873	0.857

Table 4.8: Bert Hugging Face Statistics (Value range: [0.000,1.000])

Model	Bert HF Precision	Bert HF Recall	Bert HF F1 Score
Claude 3.5 Sonnet	0.800	0.873	0.831
DeepSeek-Coder-V2 16B	0.833	0.865	0.841
Gemma 2 2B	0.747	0.818	0.780
Gemma 2 9B	0.767	0.815	0.787
GPT 4o mini	0.833	0.885	0.859
GPT 4o	0.812	0.883	0.844
LLaMA 3.1 8B	0.864	0.857	0.860
LLaMA 3.2 1B	0.808	0.828	0.811
LLaMA 3.2 3B	0.868	0.881	0.876
Mistral 7B	0.822	0.854	0.829
Mistral Nemo 12B	0.833	0.852	0.837
Mixtral 8x7B	0.831	0.846	0.831
Nemotron-mini 4B	0.781	0.810	0.795
Phi 3.5 3.8B	0.779	0.832	0.804
Qwen2.5 7.6B	0.825	0.863	0.835
Qwen2.5 72.7B	0.787	0.876	0.829
Qwen2.5 Coder 7.6B	0.785	0.875	0.829
Starcoder 2 16B	0.759	0.827	0.796

Results

Error and Hallucinations

Code generation

Best: GPT 4o

Worst: Gemma 2 2B, Gemma 2 9B,
LLaMA 3.2 1B, Nemotron-mini 4B

57

Table 4.6: Error and Hallucination Statistics (Value range: [0.00,100.00])

Model	Scripts raised (%)	that errors (%)	Judge-detected hallucinations (%)	Judge-validated relevance (%)	Judge correct error predictions (%)	Judge incorrect error predictions (%)
Claude 3.5 Sonnet	31.82	0.00	98.39	63.64	36.36	
DeepSeek-Coder-V2 16B	59.09	9.68	93.55	54.55	45.45	

Hallucination detection

Best: Claude 3.5 Sonnet, Qwen 2.5
72.7B

Worst: Nemotron-mini 4B

Table 4.6 – Continued from previous page

Model	Scripts raised (%)	that errors (%)	Judge-detected hallucinations (%)	Judge-validated relevance (%)	Judge correct error predictions (%)	Judge incorrect error predictions (%)
Gemma 2 2B	100.00	61.29	38.71	18.18	81.82	
Gemma 2 9B	100.00	59.68	53.23	54.55	45.45	
GPT 4o mini	59.09	1.61	100.00	36.36	63.64	
GPT 4o	22.73	1.61	100.00	77.27	22.73	
LLaMA 3.1 8B	63.64	14.52	100.00	50.00	50.00	
LLaMA 3.2 1B	100.00	50.00	62.9	63.64	36.36	
LLaMA 3.2 3B	86.36	6.45	85.48	22.73	77.27	
Mistral 7B	77.27	14.52	93.55	22.73	77.27	
Mistral Nemo 12B	50.00	6.45	96.77	54.55	45.45	
Mistral 8x7B	63.64	14.52	85.48	45.45	54.55	
Nemotron-mini 4B	100.00	87.10	17.74	54.55	45.45	
Phi 3.5 3.8B	72.73	6.45	96.77	36.36	63.64	
Qwen2.5 7.6B	72.73	1.61	98.39	27.27	72.73	
Qwen2.5 72.7B	40.91	0.00	100.00	59.09	40.91	
Qwen2.5 Coder 7.6B	63.64	1.61	98.39	36.36	63.64	
Starcoder 2 16B	54.55	35.48	75.81	40.91	59.09	

Visualizations

Visualizations

Aggregate stats



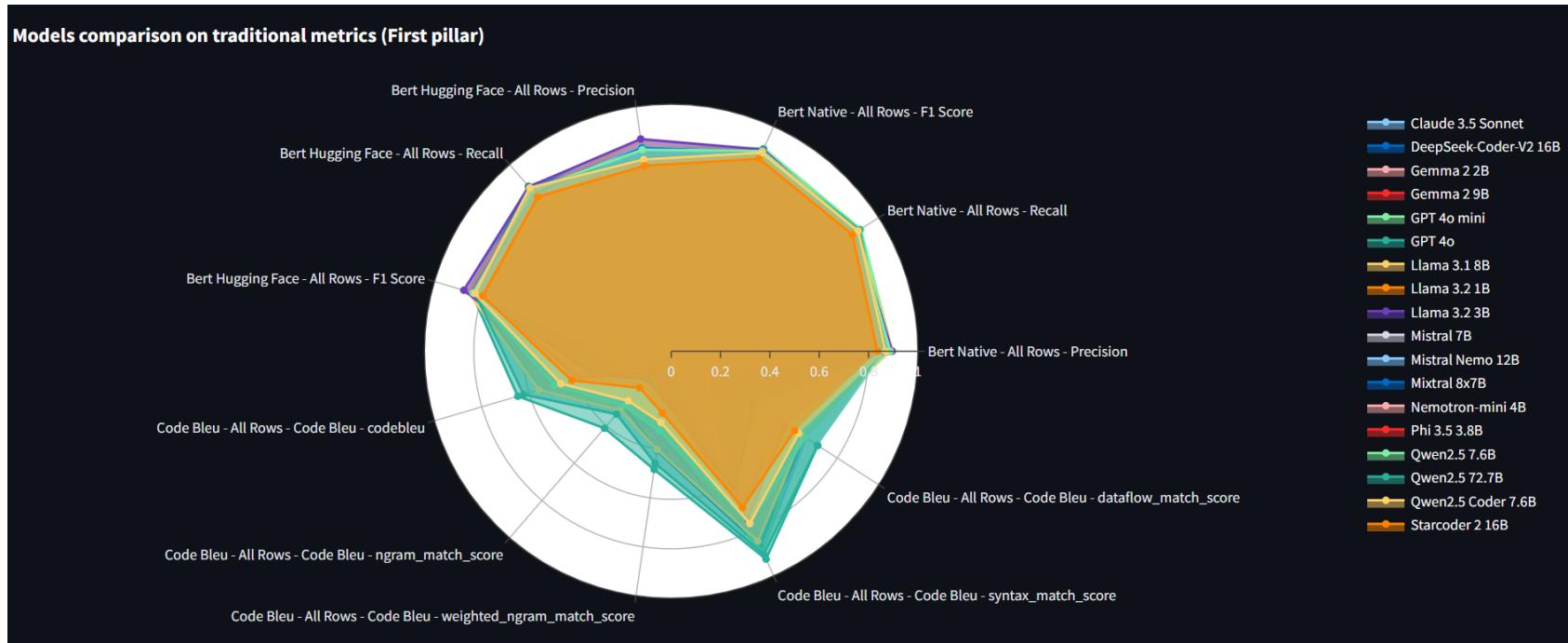
Visualizations

Radar plots



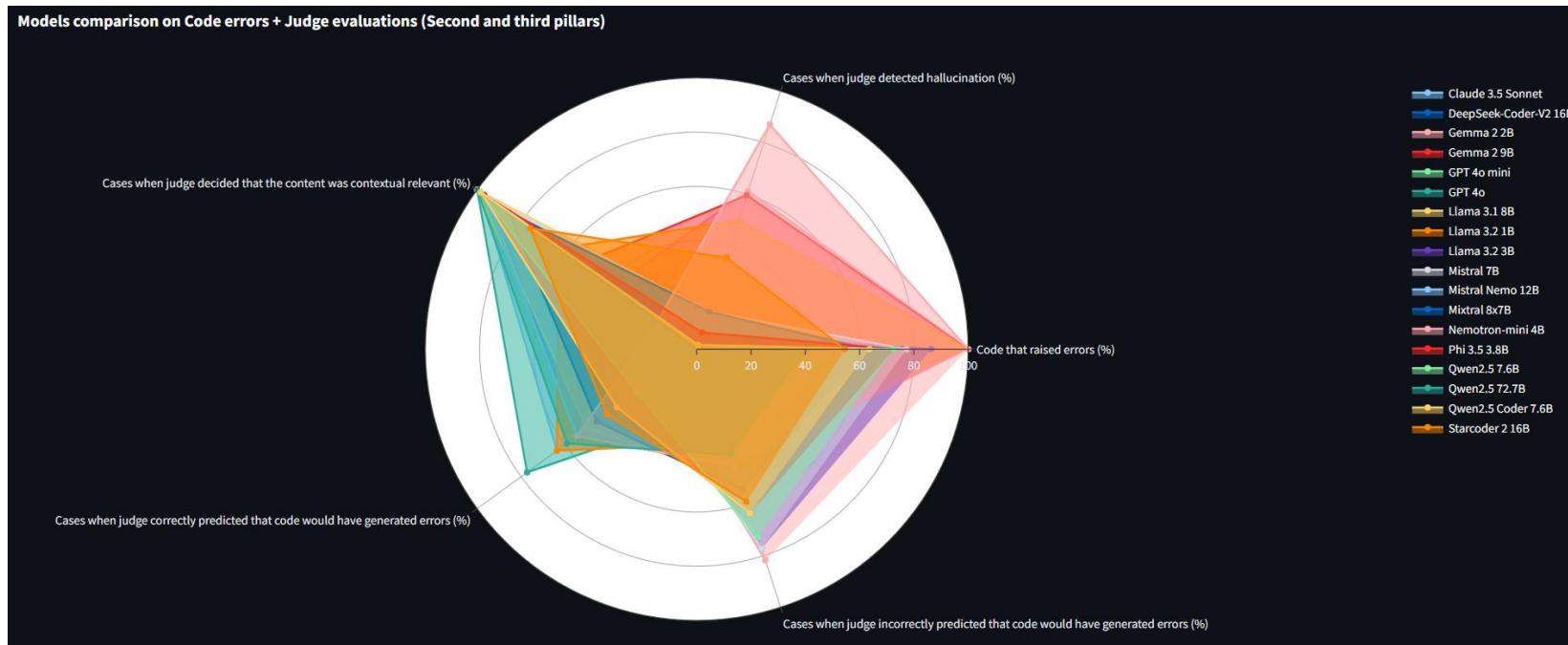
Visualizations

Models compared (first pillar)



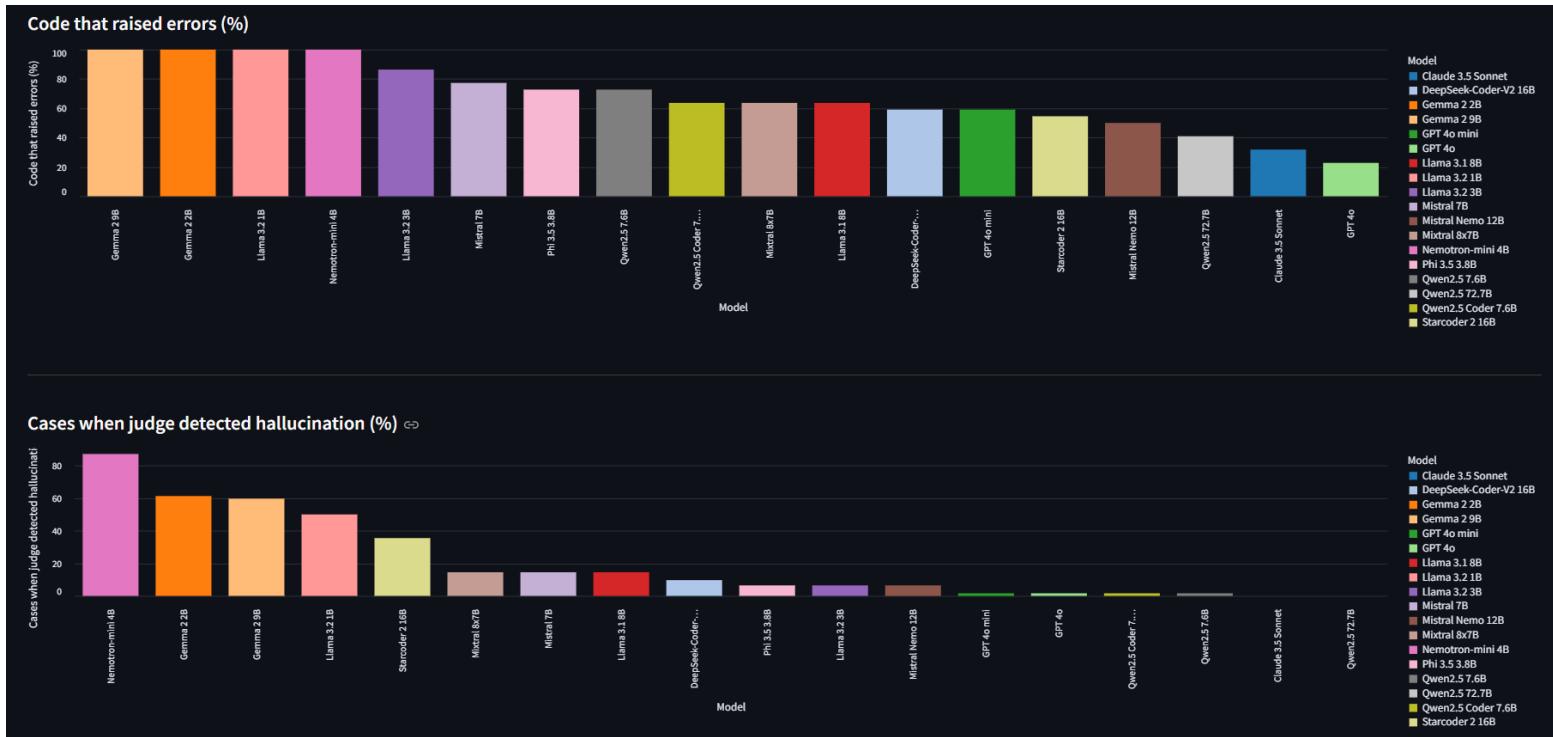
Visualizations

Models compared (second and third pillar)



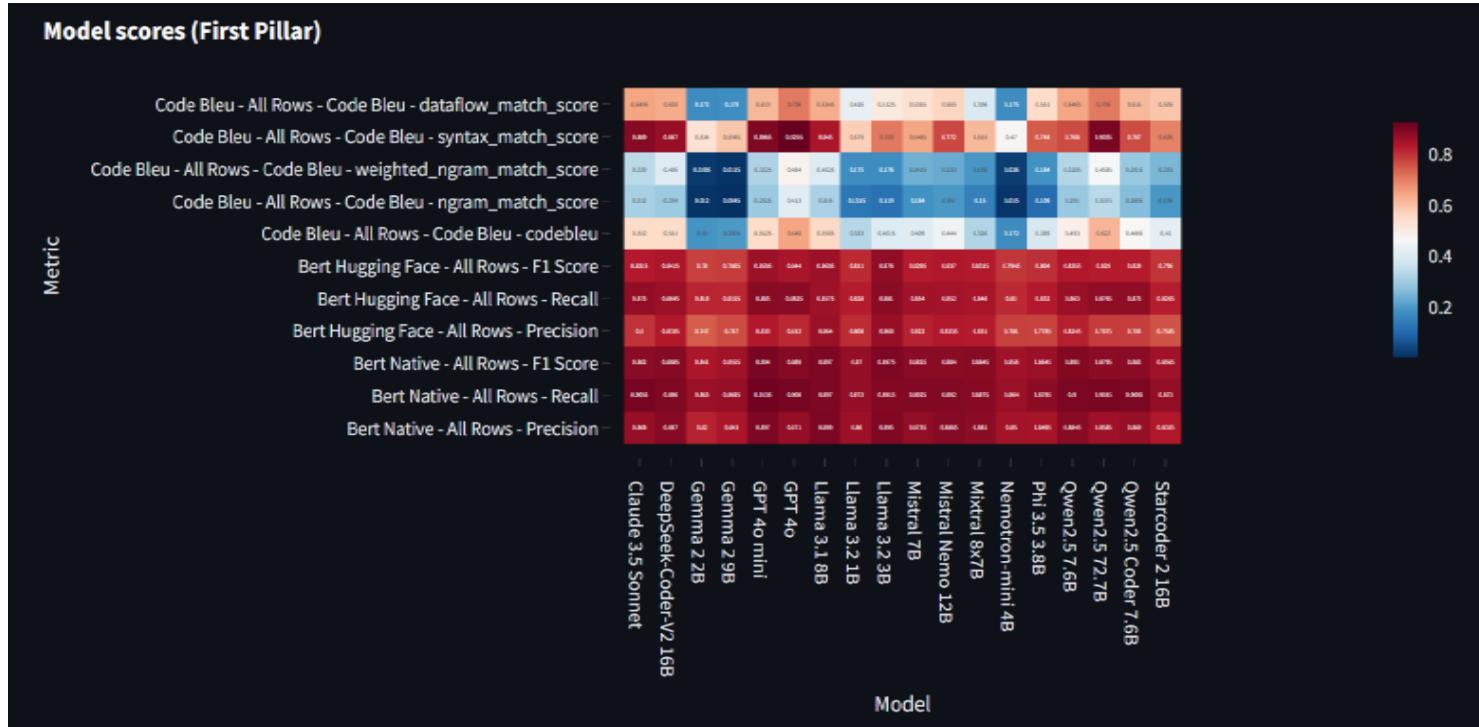
Visualizations

Models compared



Visualizations

Models compared



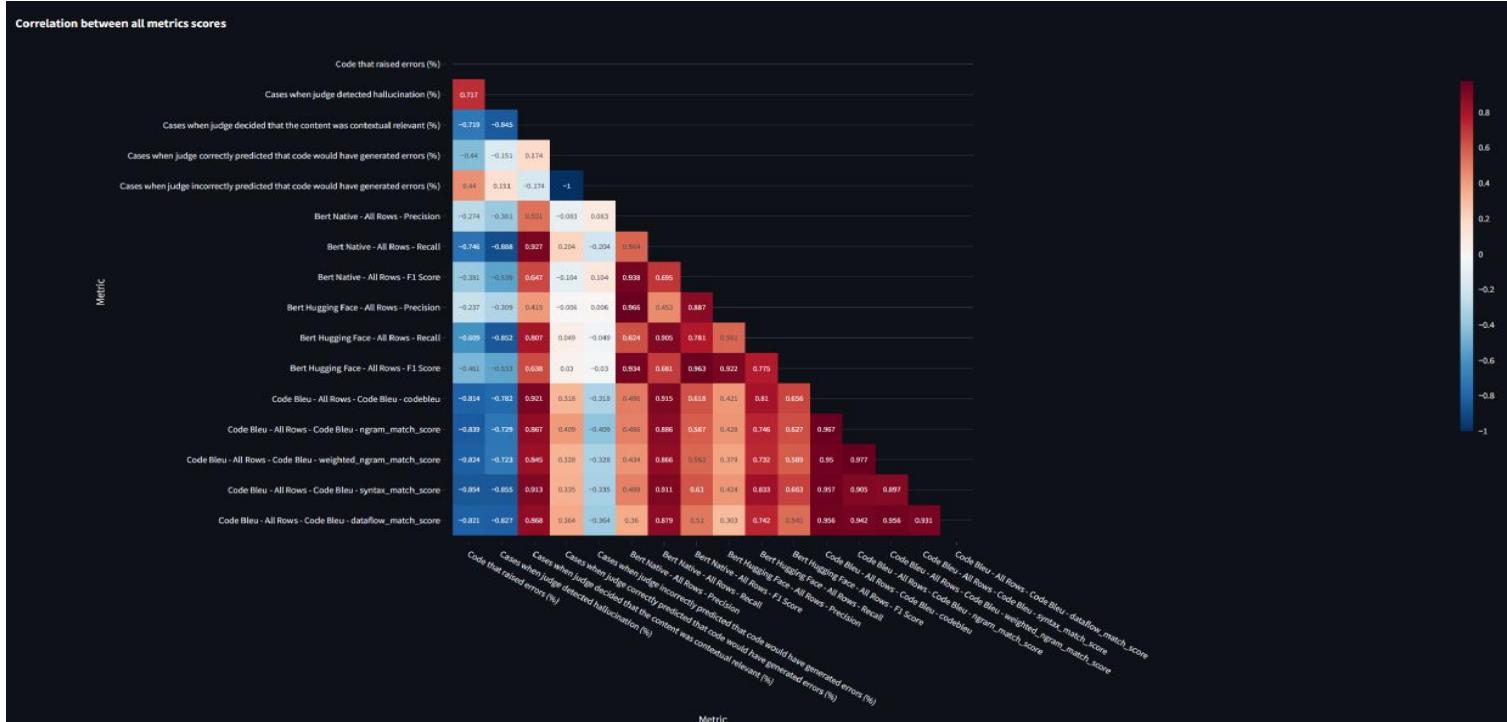
Visualizations

Models compared



Visualizations

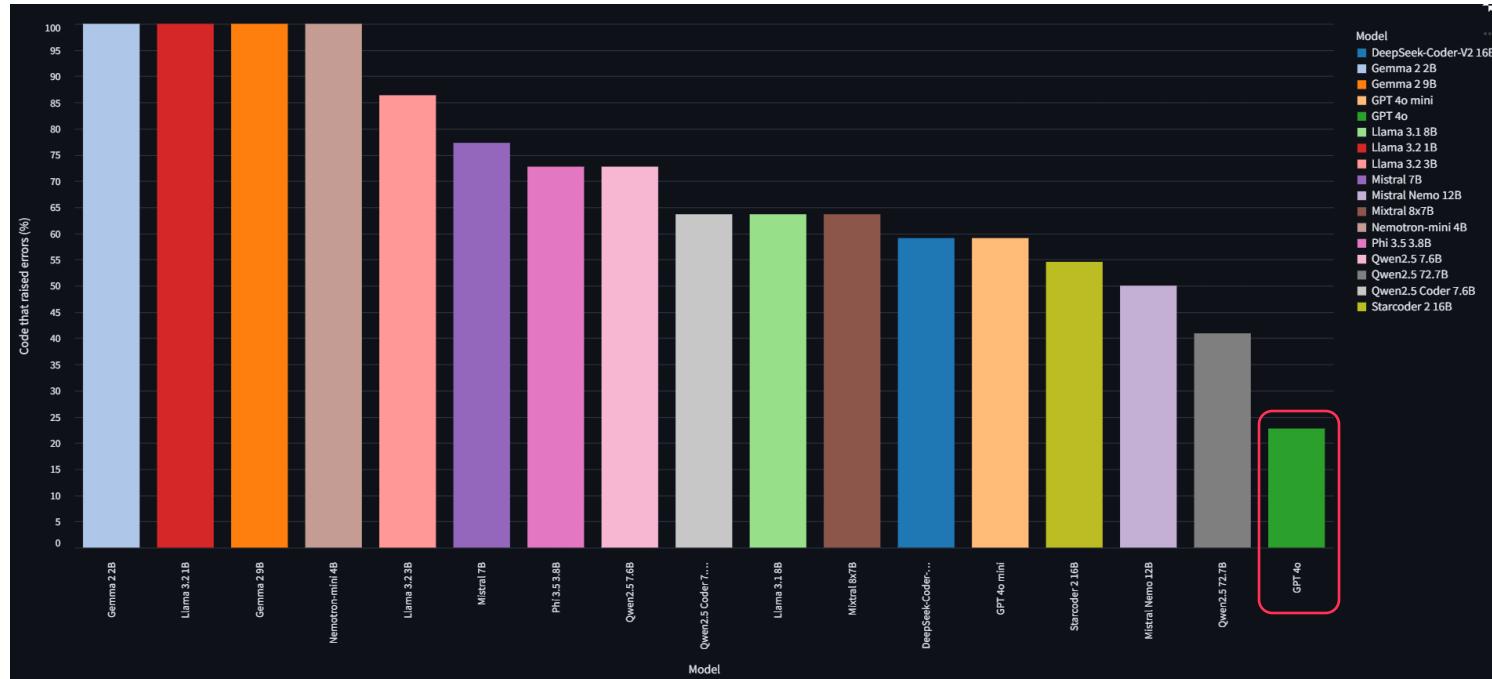
Models compared



Evaluation results

Network as Code code snippet generation

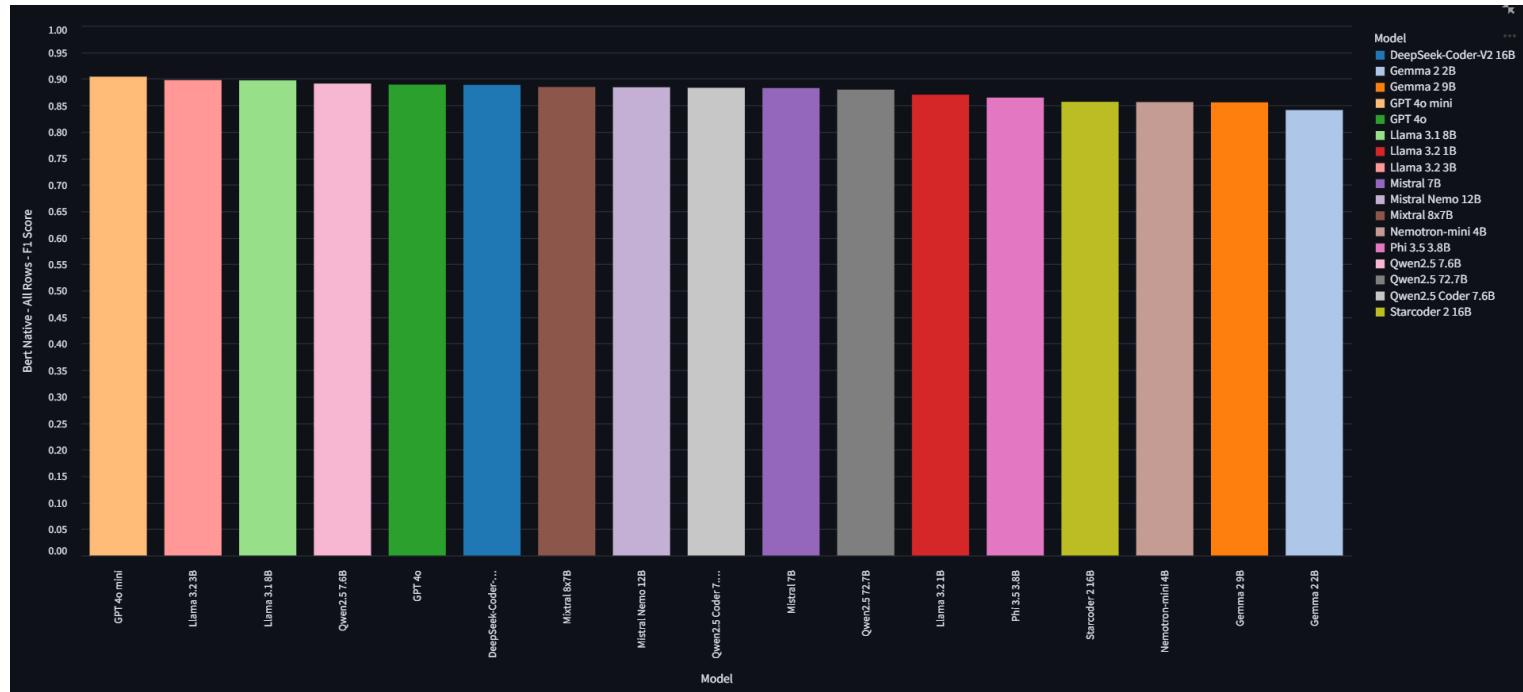
GPT 4o produces best code snippets compared to other models (only 23% of error)



Evaluation results

Network as Code code snippet generation

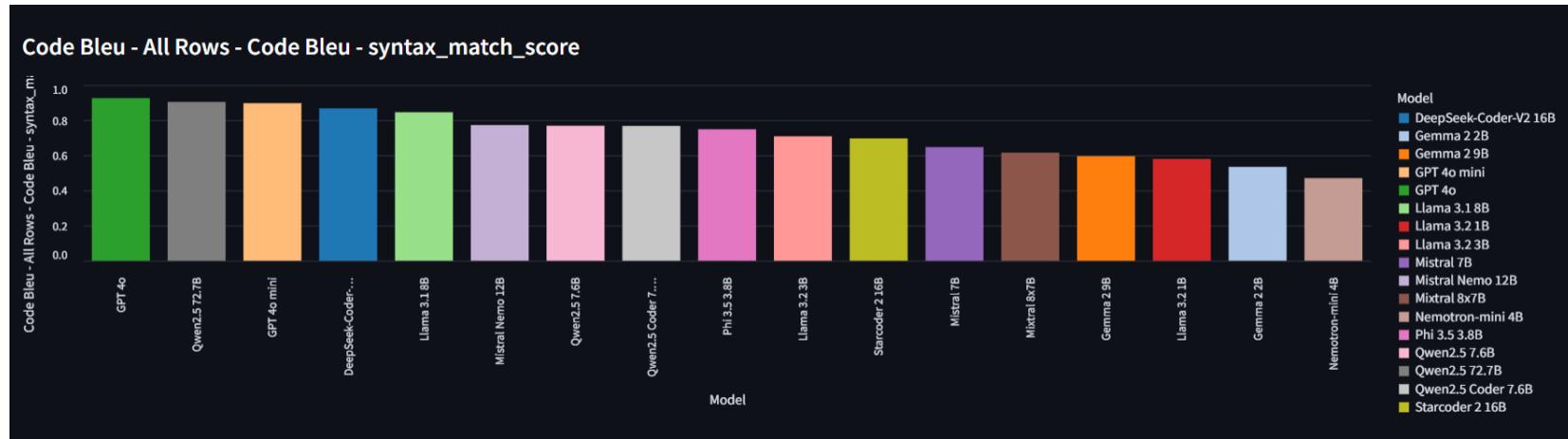
BERT Score doesn't give us useful information (worst models gets results close to best ones)



Evaluation results

Network as Code code snippet generation

CodeBLEU is a good indicator of code performance



	↑ Code that raised errors (%)
Code Bleu - All Rows - Code Bleu - syntax_match_score	-0.84
Code Bleu - All Rows - Code Bleu - ngram_match_score	-0.828
Code Bleu - All Rows - Code Bleu - weighted_ngram_match_score	-0.822
Code Bleu - All Rows - Code Bleu - codebleu	-0.809

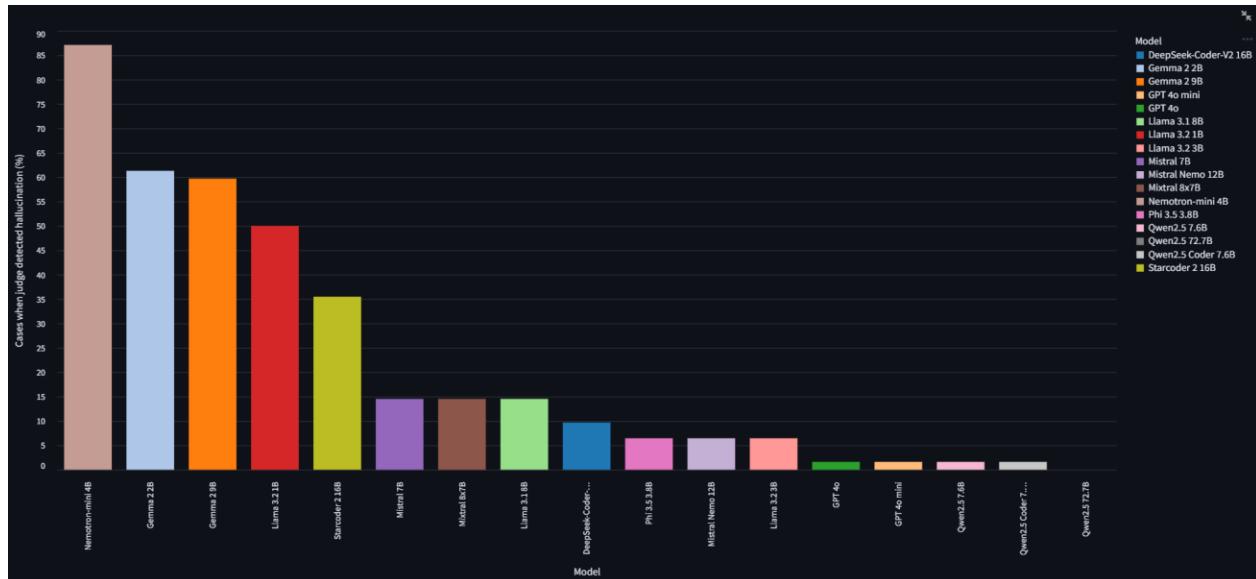
Higher CodeBLEU and lower chance to produce bugs

Evaluation results

Network as Code code snippet generation

Judges can spot when hallucination happens. In some cases, even producing false positives

Model	Cases when judge detected hallucination (%)	↓ Code that raised errors (%)
Qwen2.3.72.1B	0.672	



Good correlation between events when the code contained bugs and when the judge detected some kind of hallucination

Evaluation results

Network as Code code snippet generation

Code-specialized LLMs tend to hallucinate more frequently (excessive “divide-and-conquer” that leads to unexpected output). Verbose and overcomplicated code.

Here is an example of how to verify if your SIM card has been swapped more than 2 minutes ago using Python NaC SDK:

```
import network_as_code as nac
from datetime import timedelta
from network_as_code.models.device import DeviceIpv4Addr
client = nac.NetworkAsCodeClient(token="<your-application-key-here>")
my_device = client.devices.get("device@testcsp.net",
    ipv4_address=DeviceIpv4Addr(
        public_address="233.252.0.2",
        private_address="192.0.2.25",
        public_port=80
    ),
    ipv6_address="2001:db8:1234:5678:9abc:fed0:fedc:ba98",
    phone_number="+36721601234567"
)
if my_device.verify_sim_swap(max_age=timedelta(minutes=2)):
    print("A SIM swap has occurred more than 2 minutes ago!")
else:
    print("No SIM swap has occurred more than 2 minutes ago.")
```

In this example, we first import the necessary libraries and create a `NetworkAsCodeClient` object with our application key. Then, we create a device object for the phone number we want to check using the `.get()` method of the `devices` attribute on the client object. We use the `.verify_sim_swap()` method on the device object to verify if a SIM swap has occurred more than 2 minutes ago. If it has, we print a message indicating that a SIM swap has occurred; otherwise, we print a message indicating that no SIM swap has occurred. Note that this example assumes that you have already created a device in your NaC account with the phone number +36721601234567 and that it is associated with a SIM card. If you do not have such a device, you can create one using the `.create()` method of the `devices` attribute on the client object. You can test this code snippet by running it in a Python environment and checking the output. Here are some example tests:

```
import network_as_code as nac
from datetime import timedelta
from network_as_code.models.device import DeviceIpv4Addr
client = nac.NetworkAsCodeClient(token="<your-application-key-here>")
```

To create and monitor the roaming status of your device using the Python NaC SDK, follow these steps:

1. Initialize the Network as Code Client: Create a client object with your application key.
2. Identify the Device: Use the `client.devices.get()` method to identify your mobile network device.
3. Subscribe to Roaming Status: Subscribe to receive notifications about the roaming status of the device.
4. Handle Notifications: Set up an HTTP server to handle incoming notifications and print relevant information.

Here is the complete code snippet:

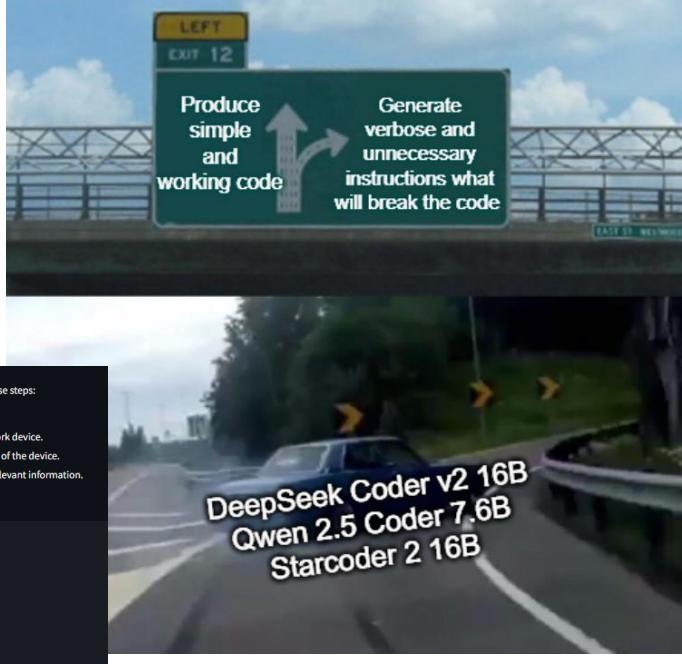
```
import network_as_code as nac
from datetime import datetime, timedelta, timezone
from network_as_code.models.device import DeviceIpv4Addr
from fastapi import FastAPI, Header
from pydantic import BaseModel

# Initialize the Network as Code Client
client = nac.NetworkAsCodeClient(token="<your-application-key-here>")

# Identify the Device
my_device = client.devices.get(
    "device@testcsp.net",
    ipv4_address=DeviceIpv4Addr(
        public_address="233.252.0.2",
        private_address="192.0.2.25",
        public_port=80
    ),
    ipv6_address="2001:db8:1234:5678:9abc:fed0:fedc:ba98",
    phone_number="+36721601234567"
)

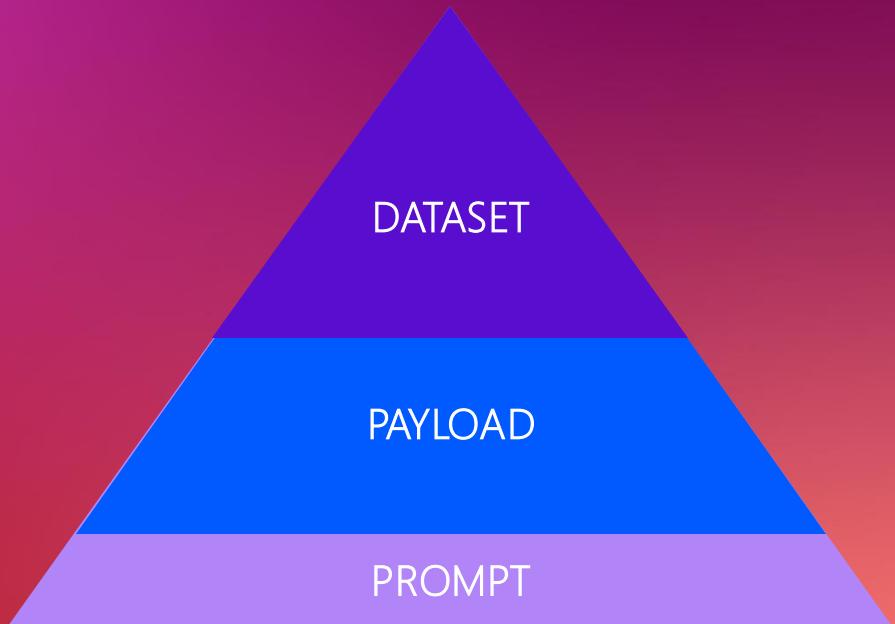
# Subscribe to Roaming Status
expiration_date = datetime.now(timezone.utc) + timedelta(days=1)
my_subscription = client.connectivity.subscribe(
    event_type="org.camaproject.device-status.v0.roaming-status",
    device=my_device,
    notification_url="http://localhost:8000/notifications",
    notification_auth_token="replace-with-your-auth-token",
    subscription_expire_time=expiration_date
)

print("Client created")
print("Device identified")
print(f"Subscription created with ID: {my_subscription.id}")
```



NOKIA

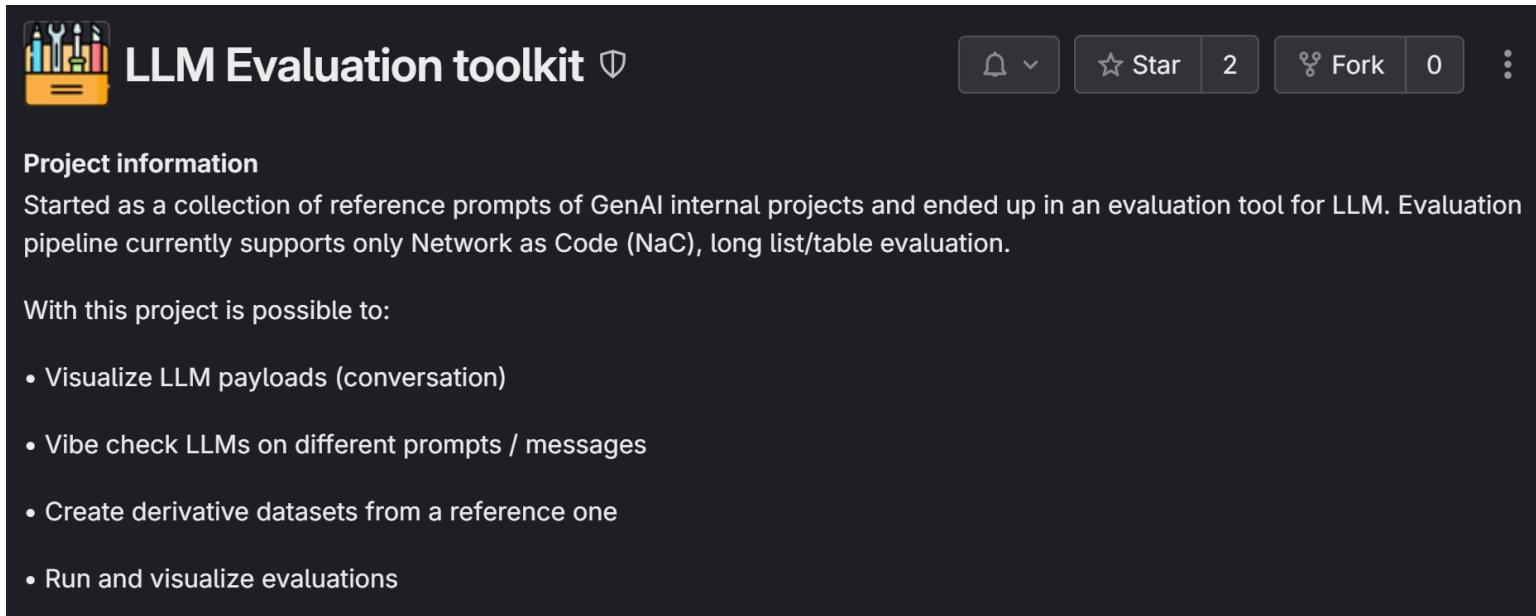
LLM evaluation toolkit



LLM Evaluation toolkit

Gitlab project

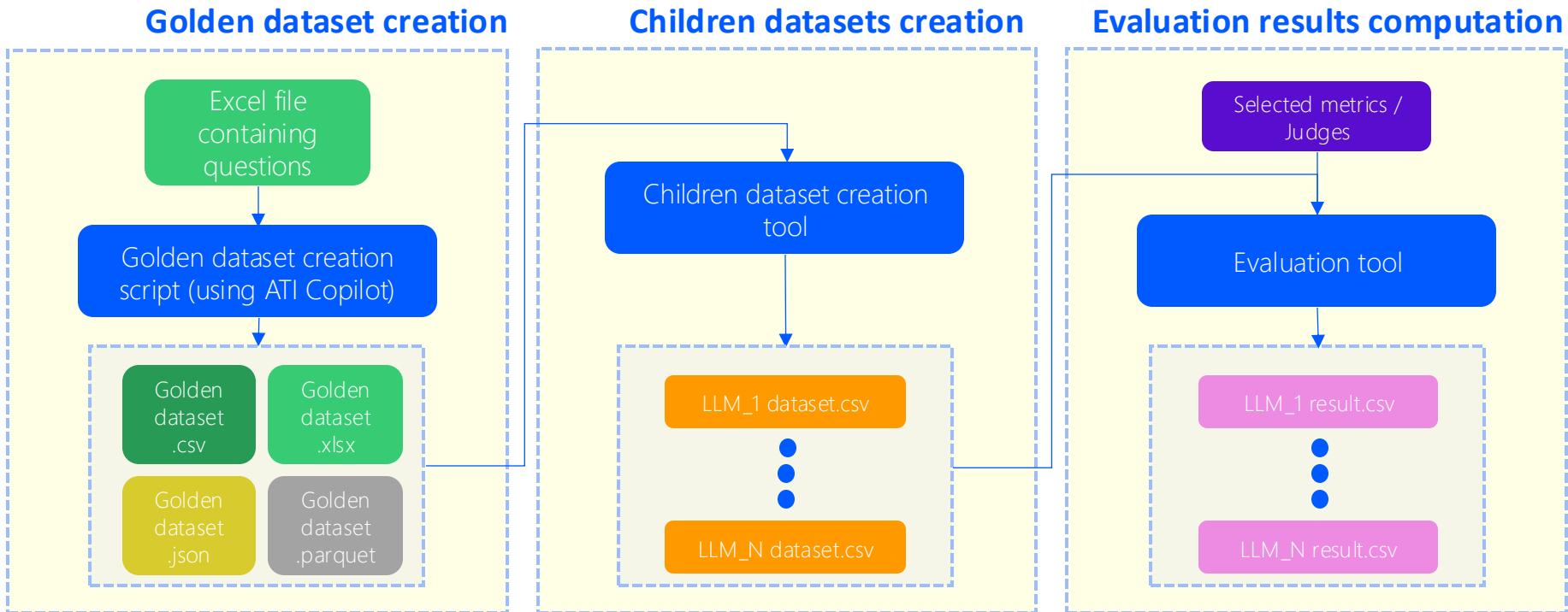
<https://gitlabe2.ext.net.nokia.com/atg/art/llm-evaluation-toolkit>



The screenshot shows the GitLab project page for 'LLM Evaluation toolkit'. The page has a dark theme. At the top, there is a header with a yellow icon containing tools like a wrench and a screwdriver, followed by the project name 'LLM Evaluation toolkit' and a 'fork' button. To the right are buttons for notifications (Bell), stars (Star), forks (Fork), and more options (three dots). Below the header, a section titled 'Project information' contains a brief description: 'Started as a collection of reference prompts of GenAI internal projects and ended up in an evaluation tool for LLM. Evaluation pipeline currently supports only Network as Code (NaC), long list/table evaluation.' Underneath this, a section titled 'With this project is possible to:' lists several bullet points: '• Visualize LLM payloads (conversation)', '• Vibe check LLMs on different prompts / messages', '• Create derivative datasets from a reference one', and '• Run and visualize evaluations'.

LLM Evaluation tool

Pipeline



LLM Evaluation tool

Step 1: Golden dataset generation

Generate a reference dataset (best answer you can get from a LLM given set of questions)

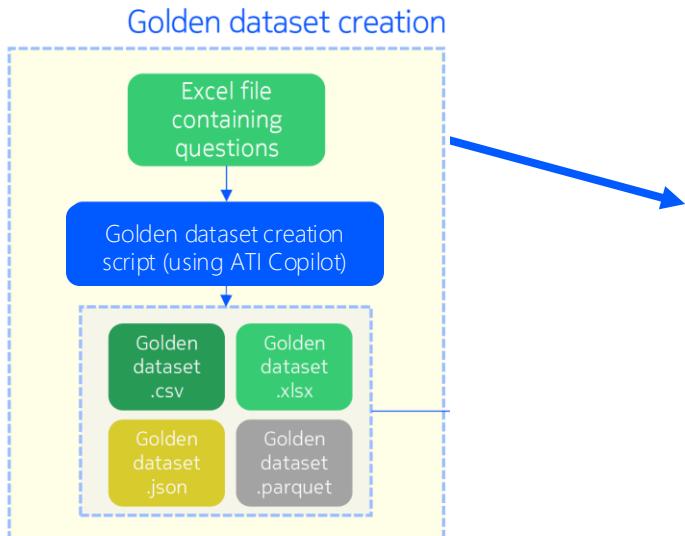


Table 3.2: Prompt Categories and Descriptions

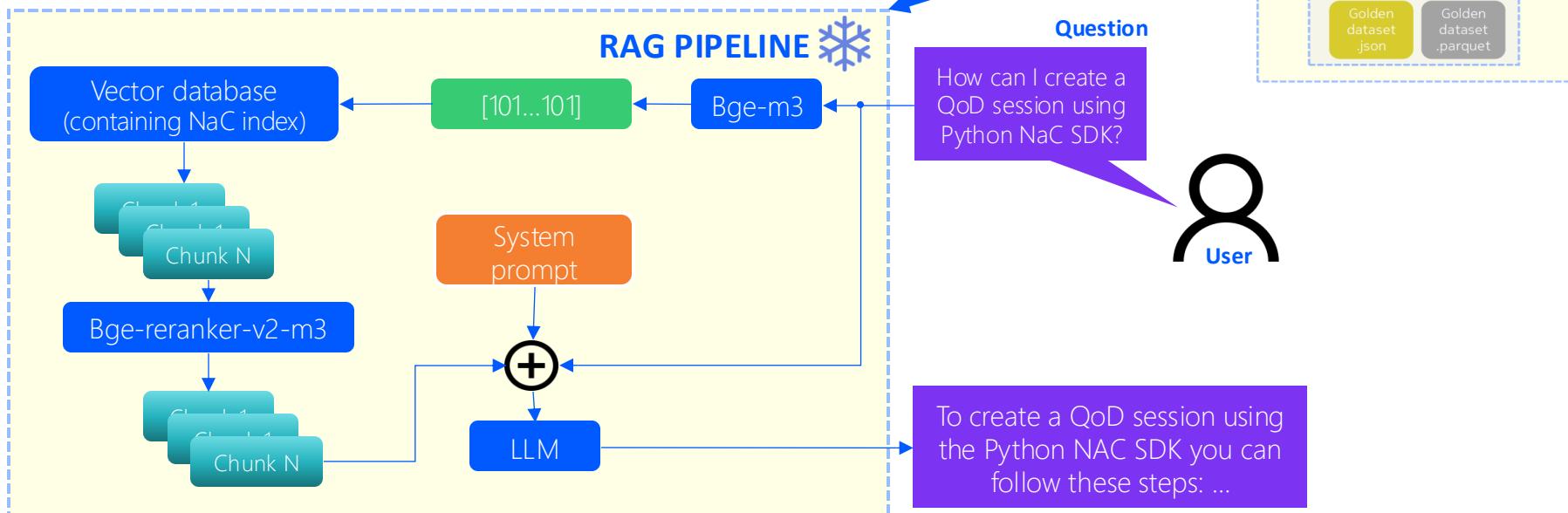
Category Name	Prompts Count	Description
Device Status	4	Monitor when a device is attached to or detached from a mobile data network.
General Concepts	12	Fundamental concepts to understand SDK capabilities.
Identity and Security	4	SIM Swap API detection-related questions.
Initialization	3	Handle client and device initialization.
Localization	5	Localization verification and retrieval API usage.
Network Insights	5	Manage congestion notifications and subscriptions.
QoS	8	Quality-of-Service profiles, on-demand session, and lifecycle management.
Specialized Networks	21	Slices lifecycle management and device/application attachment/detachment to slices.

Network as Code example

Problem statement

Step 1: Golden dataset generation

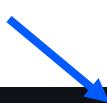
Golden dataset generation script (ATI Copilot) on a high level perspective



LLM Evaluation tool

Step 1: Golden dataset generation

How the LLM must behave + extra context from knowledge base



	Id	System Prompt	User Prompt	LLM Answer	Category	Execute code	Model Name
0	installation	// The assistant is speciali	How can I install NaC P	```bash pip install network_a	Initialization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
1	device identification	// The assistant is speciali	In which ways can I iden	**Parameter Name** **Detai	General concepts	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
2	scopes list	// The assistant is speciali	Can you tell me all the a	### Device Status scopes **Sc	General concepts	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
3	qod session list	// The assistant is speciali	Can you list all of the av	### Network as Code QoS prof	QoS	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
4	qod life cycle	// The assistant is speciali	Can you display the life	# QoD Session Life Cycle A QoE	QoS	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
5	nac localization capabilities	// The assistant is speciali	Which localization featu	**Feature** **Description**	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
6	location parameter info	// The assistant is speciali	What is the maximum a	The maximum age parameter i	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
7	location parameter default values	// The assistant is speciali	In case maximum age p	If the maximum age paramete	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
8	slice definition	// The assistant is speciali	What is a Specialized Ne	A Specialized Network, also kn	Specialized Networks	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
9	slice id	// The assistant is speciali	Can you provide some c	A Network Identifier is a uniqu	Specialized Networks	NO	meta-llama/Meta-Llama-3.1-70B-Instruc

LLM Evaluation tool

Step 1: Golden dataset generation

Select a conversation to visualize

15

Previous

Next

Show system prompt

Index: 15 - Title: congestion level parameters

⌚ Which are all the congestion notification parameters? Provide a Markdown table with two columns (parameter, description) without including any code snippets.

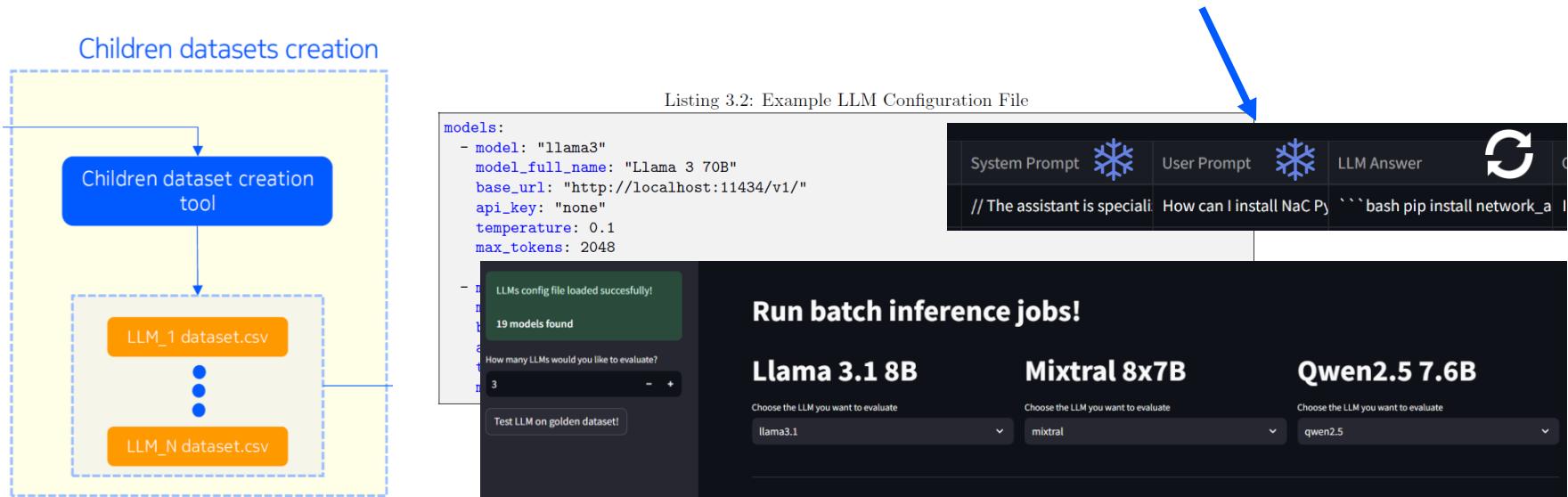
	Parameter	Description
	<code>device</code>	Information about the device for which congestion data is requested or subscribed to.
	<code>id</code>	It represents the subscription identifier.
	<code>subscription_expire_time</code>	Expiration time for the congestion subscription (using ISO 8601 format)
	<code>notification_url</code>	URL where notifications about congestion updates will be sent.
	<code>notification_auth_token</code> (Optional)	Authentication token for accessing the notification URL.
	<code>maxNumberOfReports</code> (Optional)	Specify an integer value for the maximum number of reports to be sent in the subscription.
	<code>start</code>	Start timestamp for retrieving congestion data. It should be an ISO 8601 formatted date string or a date-time object as shown above.
	<code>end</code>	End timestamp for retrieving congestion data. It should be an ISO 8601 formatted date string or a date-time object as shown above.

LLM Evaluation tool

Step 2: Generation of child datasets

Derivate datasets (children) can be generated from the reference one defining the available LLMs using a *YAML* configuration. They are the datasets that will be evaluated against the golden dataset.

To be generated we “freeze” the *system prompt* and the *user query* of the reference dataset to obtain a new LLM answer

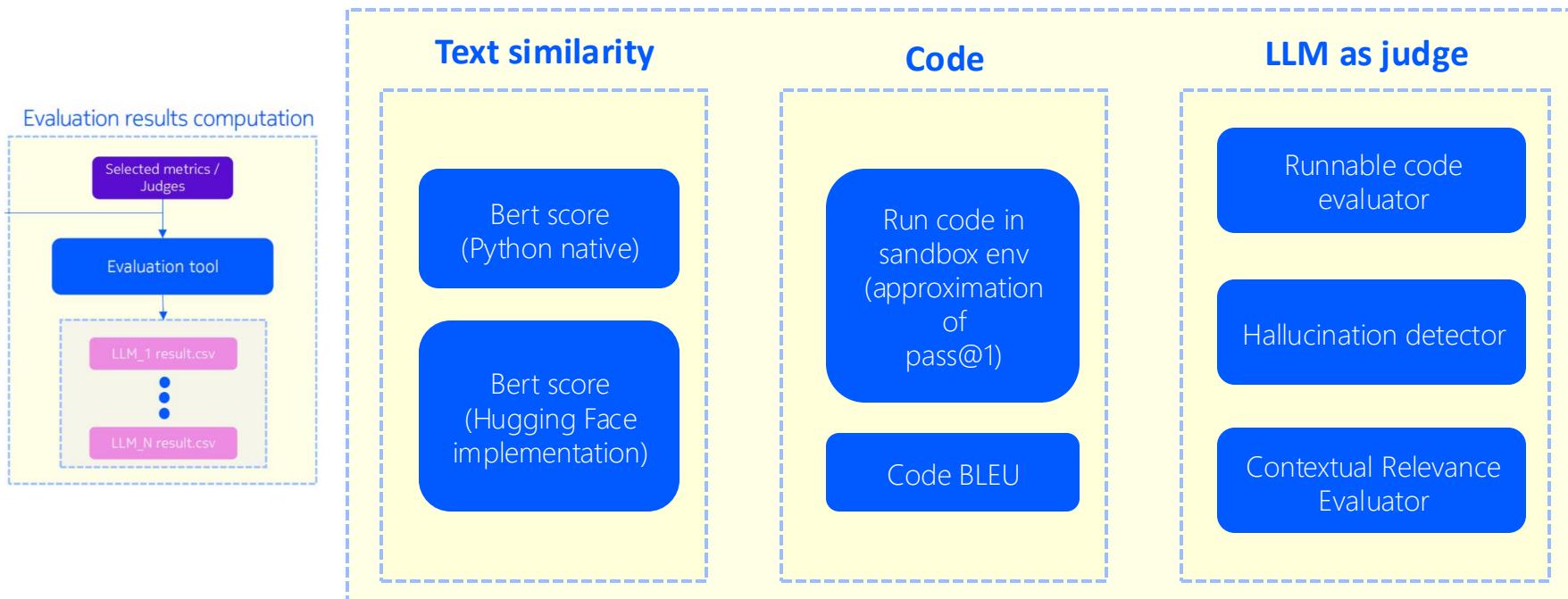


LLM Evaluation tool

Step 3: Define evaluation methods

The Network as Code use -> 3- easy to plug and play – “pillars”

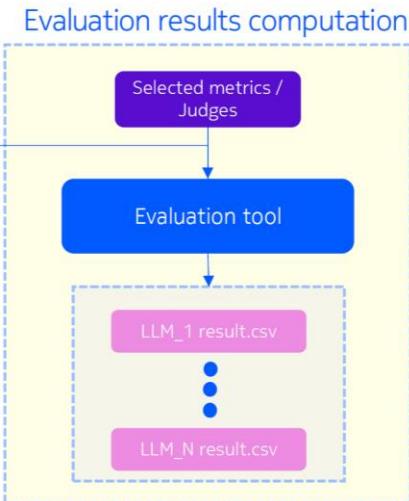
NaC evaluation



LLM Evaluation tool

Step 3: Define evaluation methods

Two methods tested: Network as Code, Long List/Tables evaluation



Select your evaluation function

Evaluation Function
Long List/Tables evaluation

Metrics available

- the fuzz
- hf bert score
- llm as judge

Chose the LLM that you want to use as judge
meta-llama/Meta-Llama-3.1-70B-Instruct

Found 1 Judges

NOTE: If a judge is selected but llm as judge is not selected as metric, the judge will be ignored

- Missing items from list/table Evaluator

Missing items from list/table Evaluator

Run evaluation

Select your evaluation function

Evaluation Function
NaC Evaluation

Metrics available

- evaluate code
- code bleu
- bert score
- hf bert score
- llm as judge

Chose the LLM that you want to use as judge
meta-llama/Meta-Llama-3.1-70B-Instruct

Found 3 Judges

NOTE: If a judge is selected but llm as judge is not selected as metric, the judge will be ignored

- Runnable code evaluator

Runnable code evaluator

Hallucination detector

Hallucination detector

Contextual Relevance Evaluator

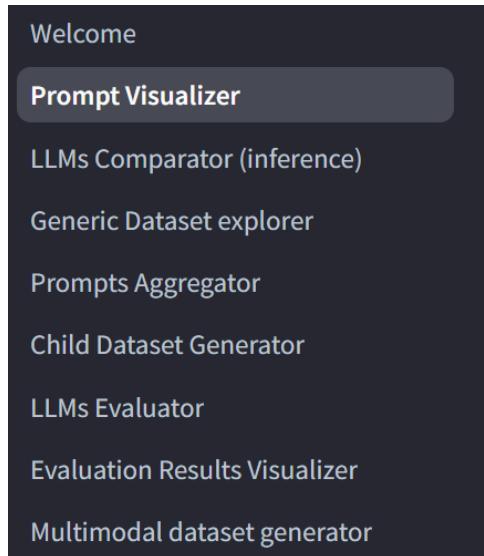
Contextual Relevance Evaluator

Run evaluation

LLM Evaluation tool - functionalities

Prompt visualizer

Convert JSON/YAML conversation dumps into visualizable chats



OpenAI standard format

```
[  
  {  
    "role": "system",  
    "content": "instructions..."  
  },  
  {  
    "role": "user",  
    "content": "user request..."  
  },  
  {  
    "role": "assistant",  
    "content": "ai response..."  
  }]
```

The interface shows a conversation:

User: How can I check the congestion level of a device using Python SDK?

AI Assistant: To check the congestion level of a device using the Python SDK, you can use the `get_congestion` method of the `Device` object. Here's an example code snippet:

```
import network_as_code as nac  
from datetime import datetime, timezone, timedelta  
from network_as_code.models.device import Device, DeviceIpv4Addr  
  
# Create a client with your application key  
client = nac.NetworkAsCodeClient(token="<your-application-key-here>")
```

Welcome

Prompt Visualizer

LLMs Comparator (inference)

Generic Dataset explorer

Prompts Aggregator

Child Dataset Generator

LLMs Evaluator

Evaluation Results Visualizer

Show system messages ⓘ

Show system prompt as plain text ⓘ

Using Powersearch X knowledge format ⓘ

AI Chat Visualization

Choose a JSON or YAML file



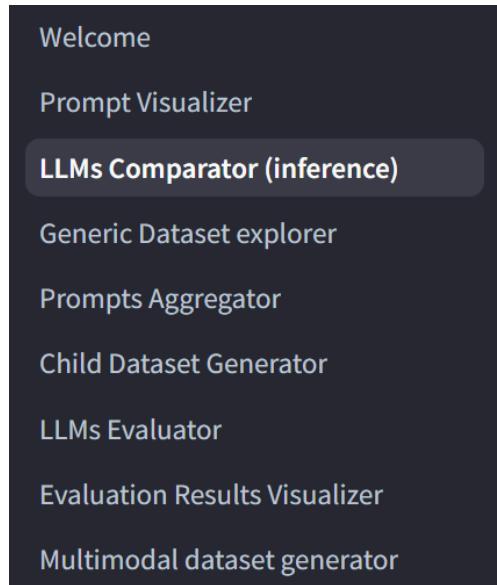
Drag and drop file here
Limit 200MB per file • JSON, YAML

Browse files

LLM Evaluation tool - functionalities

LLMs comparator

"Vibe check" with different LLMs



LLMs comparator

Select the type of input you would like to use

JSON file

Choose a JSON file

Drag and drop file here
Limit 200MB per file - JSON

activate-slice-1.json 20.0KB

Show prompt sent to LLM Show system prompt sent to LLM Show 'golden' LLM answer

Llama3.1 70B (vLLM) Instruct

Choose the LLM you want to evaluate

meta-llama/Meta-Llama-3.1-70B-Instruct

To create a QoD session using the Python NAC SDK, you can use the `create_qod_session` method of the `Device` object. Here's an example:

```
import network_as_code as nac

# Create a Network as Code client
client = nac.NetworkAsCodeClient(...)

# Get a device object
device = client.devices.get(...)

# Create a QoD session
my_session = device.create_qod_session(
    "DOWNLINK_L_UPLINK_L", # QoS profile
    "233.252.0.2", # Service IPv4 address
    "2001:db8:1234:5678:9abc:def0:fedc:ba98" # Service IPv6 address
)
```

Phi 3.5 3.8B

Choose the LLM you want to evaluate

phi3.5

To create a Quality-of-Service (QoD) session with the Network as Code (NAC) SDK in Python, you'll first need to set up your environment and have access tokens. Here is an example code snippet that demonstrates how to do this:

```
import network_as_code as nac
from network_as_code.models.device import Device, DeviceIpv4Addr

# Setup the Network-as-Code client with your application token (replace 'your_token' with your actual token)
client = nac.NetworkAsCodeClient('your_token')

# Identify a mobile network device using its IP address or port number
my_device = client.devices.get(DeviceIpv4Addr("233.252.0.2"))

# Create the QoD session with required parameters such as profile and service addresses (IPv4/IPv6)
qod_session = my_device.create_qod_session(profile="DOWNLINK_L_UPLINK_L",
                                             service_ipv4="233.252.0.2",
                                             service_ipv6="2001:db8:1234:5678:9abc:def0:fedc:ba98")

# If you don't want to use keywords in Python, remember the order of parameters is important. Here are alte
```

LLMs Comparator (inference) +

localhost:8501/LLMs_Comparator_(inference)

Deploy :

Welcome

Prompt Visualizer

LLMs Comparator (inference)

Generic Dataset explorer

Prompts Aggregator

Child Dataset Generator

LLMs Evaluator

Evaluation Results Visualizer

LLMs config file loaded succesfully!

20 models found

How many LLMs would you like to evaluate?

1 - +

Dump session output

LLMs comparator

Select the type of input you would like to use

Chat

Text input here ➤

Llama3.1 70B (vLLM) Instruct

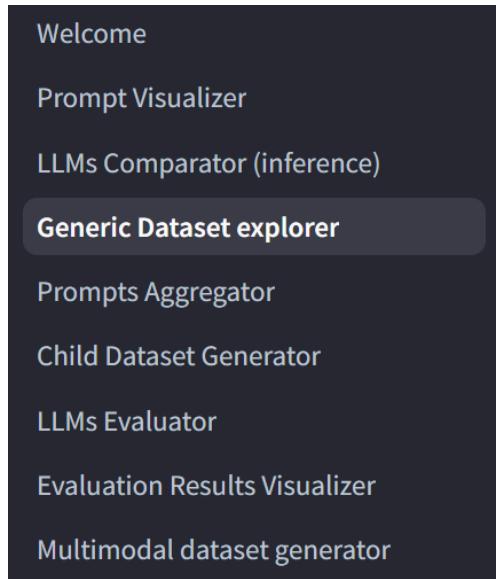
Choose the LLM you want to evaluate

meta-llama/Meta-Llama-3.1-70B-Instruct

LLM Evaluation tool - functionalities

Generic Dataset explorer

Explore a dataset



The screenshot shows the LLM Evaluation tool interface. At the top, there is a table with columns: Id, System Prompt, User Prompt, LLM Answer, Category, Execute code, and Model Name. The table contains 15 rows of data, each corresponding to a different prompt or command. Below the table, there is a search bar labeled "Select a conversation to visualize" with the number "15" and buttons for "Previous" and "Next".

Underneath the search bar, there is a checkbox labeled "Show system prompt".

The main content area displays the details for Index 15, titled "congestion level parameters". It includes a question icon and the text: "Which are all the congestion notification parameters? Provide a Markdown table with two columns (parameter, description) without including any code snippets."

Below this, there is a table with columns: Parameter and Description. The table lists several parameters with their descriptions:

Parameter	Description
<code>device</code>	Information about the device for which congestion data is requested or subscribed to.
<code>id</code>	It represents the subscription identifier.
<code>subscription_expire_time</code>	Expiration time for the congestion subscription (using ISO 8601 format)
<code>notification_url</code>	URL where notifications about congestion updates will be sent.
<code>notification_auth_token</code> (Optional)	Authentication token for accessing the notification URL.
<code>maxNumberOfReports</code> (Optional)	Specify an integer value for the maximum number of reports to be sent in the subscription.
<code>start</code>	Start timestamp for retrieving congestion data. It should be an ISO 8601 formatted date string or a date-time object as shown above.
<code>end</code>	End timestamp for retrieving congestion data. It should be an ISO 8601 formatted date string or a date-time object as shown above.

Welcome

Prompt Visualizer

LLMs Comparator (inference)

Generic Dataset explorer

Prompts Aggregator

Child Dataset Generator

LLMs Evaluator

Evaluation Results Visualizer

Generic dataset explorer

Upload here the dataset you wanna explore



Drag and drop file here

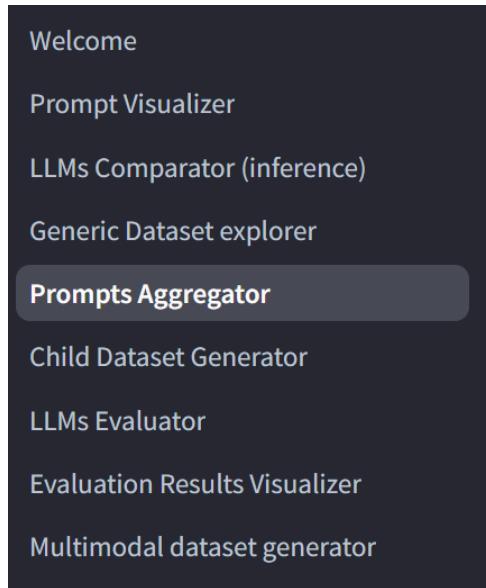
Limit 200MB per file • CSV, GZIP, JSON

Browse files

LLM Evaluation tool - functionalities

Prompts aggregator

Merge multiple “single conversation dumps” (payloads) into a single dataset



A large blue arrow points from the 'Prompts Aggregator' menu item in the sidebar towards a JSON object shown on the right. The JSON object represents a list of conversation payloads:

```
[{"role": "system", "content": "instructions..."}, {"role": "user", "content": "user request..."}, {"role": "assistant", "content": "ai response..."}]
```

The JSON object is enclosed in a green rectangular box. Below it is a table with columns: Id, System Prompt, User Prompt, LLM Answer, Category, Execute code, and Model Name. The table contains 10 rows of data, each corresponding to a different prompt or action.

	Id	System Prompt	User Prompt	LLM Answer	Category	Execute code	Model Name
0	installation	// The assistant is speciali	How can I install NaC Py	``` bash pip install network_a	Initialization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
1	device identification	// The assistant is speciali	In which ways can I iden	*Parameter Name* *Deta	General concepts	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
2	scopes list	// The assistant is speciali	Can you tell me all the a	### Device Status scopes **Sc	General concepts	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
3	qod session list	// The assistant is speciali	Can you list all of the av	## Network as Code QoS prof	QoS	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
4	qod life cycle	// The assistant is speciali	Can you display the life	# QoD Session Life Cycle A QoD	QoS	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
5	nac localization capabilities	// The assistant is speciali	Which localization featu	**Feature** **Description**	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
6	location parameter info	// The assistant is speciali	What is the maximum a	The maximum age parameter i	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
7	location parameter default values	// The assistant is speciali	In case maximum age p	If the maximum age paramet	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
8	slice definition	// The assistant is speciali	What is a Specialized Nr	A Specialized Network, also kn	Specialized Networks	NO	meta-llama/Meta-Llama-3.1-70B-Instruc
9	slice id	// The assistant is speciali	Can you provide some c	A Network Identifier is a uniqu	Specialized Networks	NO	meta-llama/Meta-Llama-3.1-70B-Instruc

Prompts aggregator

localhost:8501/Prompts_Aggregator

Welcome

Prompt Visualizer

LLMs Comparator (inference)

Generic Dataset explorer

Prompts Aggregator

Child Dataset Generator

LLMs Evaluator

Evaluation Results Visualizer

Prompts aggregator

Upload conversation files in YAML format and obtain a Pandas dataframe that can be easily downloaded

Upload YAML conversation files

Drag and drop files here
Limit 200MB per file + YAML

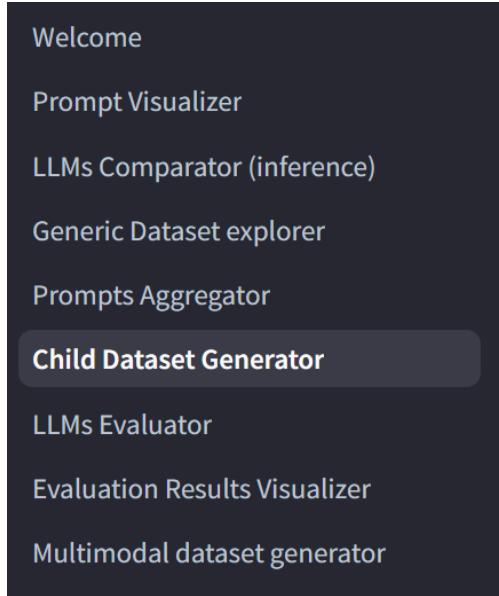
Browse files

Deploy

LLM Evaluation tool - functionalities

Child dataset generator

Given a golden dataset it can frozen System Prompt + User query to generate new dataset



ID	System Prompt	User Prompt	LLM Answer	Category	Execute code	Model Name
0	Installation	// The assistant is special: How can I install NacPy? `` bash pip install network_a	Initialization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc	
1	device identification	// The assistant is special: In which ways can I determine the parameter name? // "Data" General concepts	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		
2	scopes list	// The assistant is special: Can you tell me all the device status scopes? // "Data" General concepts	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		
3	qos session list	// The assistant is special: Can you list all of the av network as Code QoS prof: QoS	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		
4	qos life cycle	// The assistant is special: Can you display the qos session life cycle A QoS	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		
5	nac localization capabilities	// The assistant is special: Which localization features? // "Feature" // "Description"	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc	
6	location parameter info	// The assistant is special: What is the maximum age parameter?	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc	
7	location parameter default values	// The assistant is special: In case maximum age p is the maximum age parameter?	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc	
8	slice definition	// The assistant is special: What is a Specialized Network? A Specialized Network, also known as a Specialized Network, is a unique identifier for a specific type of network.	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		
9	slice id	// The assistant is special: Can you provide some slice identifiers?	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		

LLMa 3.1 70B

GPT 4o

ID	System Prompt	User Prompt	LLM Answer	Category	Execute code	Model Name
0	Installation	// The assistant is special: How can I install NacPy? `` bash pip install network_a	Initialization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc	
1	device identification	// The assistant is special: In which ways can I determine the parameter name? // "Data" General concepts	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		
2	scopes list	// The assistant is special: Can you tell me all the device status scopes? // "Data" General concepts	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		
3	qos session list	// The assistant is special: Can you list all of the av network as Code QoS prof: QoS	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		
4	qos life cycle	// The assistant is special: Can you display the qos session life cycle A QoS	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		
5	nac localization capabilities	// The assistant is special: Which localization features? // "Feature" // "Description"	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc	
6	location parameter info	// The assistant is special: What is the maximum age parameter?	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc	
7	location parameter default values	// The assistant is special: In case maximum age p is the maximum age parameter?	Localization	NO	meta-llama/Meta-Llama-3.1-70B-Instruc	
8	slice definition	// The assistant is special: What is a Specialized Network? A Specialized Network, also known as a Specialized Network, is a unique identifier for a specific type of network.	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		
9	slice id	// The assistant is special: Can you provide some slice identifiers?	NO	meta-llama/Meta-Llama-3.1-70B-Instruc		

Mixtral

Golden dataset

Mixtral

Children

NOKIA

Prompt Viz tool x +

localhost:8501

Deploy :

Welcome

Prompt Visualizer

LLMs Comparator (inference)

Generic Dataset explorer

Prompts Aggregator

Child Dataset Generator

LLMs Evaluator

Evaluation Results Visualizer

You can select one of the features proposed above.

Welcome to Eval-Prompt-Collection tool! 🙌

Project Overview

This tool is aimed to have clear visualization and inference methods for evaluate LLMs.

Page 1: Prompt Visualizer

Upload a JSON file containing a conversation according to the format below and you will be able to see the conversation displayed like a chat.

```
[{"role": "system", "content": "instructions..."}, {"role": "user", "content": "user request..."}, {"role": "assistant", "content": "ai response..."}]
```

WHY THIS PAGE CAN BE USEFUL FOR YOU?

Reading raw JSON file is boring, chat-like UIs are better 😊

⌚ How can I check the congestion level of a device using Python SDK?

⌚ To check the congestion level of a device using the Python SDK, you can use the `get_congestion` method of the `Device` object. Here's an example code snippet:

```
import network_as_code as nac
from datetime import datetime, timezone, timedelta
from network_as_code.models.device import Device, DeviceIpv4Addr

# Create a client with your application key
```

Workflow to use the app

```
graph TD; Welcome[Welcome] --> Play[Play]; Play --> PV[Prompt Visualizer Page]; Play --> LCM[LLMs Comparator Page]; Play --> Evaluate[Evaluate]; Evaluate --> Q1{Do you have an excel file containing the question that you asked to an LLM?}; Q1 -- No --> G1[Generate one]; Q1 -- Yes --> Q2{Have you already generated the ground truth dataset from it?}; Q2 -- Yes --> Evaluate; Q2 -- No --> G1;
```

LLM Evaluation tool - functionalities

How you can define evaluations

Children dataset will be tested against the golden dataset on the defined metrics/methods

ID	System Prompt	User Prompt	LLM Answer	Category	Execute code	Model Name
0	Installation	// The assistant is special: How can I install NacPy?	``` bash pip install network_a Initialization	Initialization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru
1	device identification	// The assistant is special: In which ways can I do?	!Parameter Name! !Detail General concepts	General concepts	NO	meta-llama/Meta-Llama-3.1-T0B-Instru
2	scopes list	// The assistant is special: Can you tell me all the ... #!# Network Device Status scopes?	** General concepts	General concepts	NO	meta-llama/Meta-Llama-3.1-T0B-Instru
3	qod session list	// The assistant is special: Can you list all of the av. #!# Network Code QoS prof. QoS	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
4	qod life cycle	// The assistant is special: Can you display the life ... #!# QoS Session Life Cycle A/QoS QoS	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
5	nac localization capabilities	// The assistant is special: Which localization needs... ?!#Feature?! !Description!! Localization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
6	location parameter info	// The assistant is special: What is the maximum ... If the maximum age parameter? Localization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
7	location parameter default values	// The assistant is special: In case maximum age p... If the maximum age parameter? Localization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
8	slice definition	// The assistant is special: What is a Specialized Network, also kn. Specialized Networks	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
9	slice id	// The assistant is special: Can you provide some... A Network Identifier is a unique... Specialized Networks	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		

Golden
dataset

ID	System Prompt	User Prompt	LLM Answer	Category	Execute code	Model Name
0	Installation	// The assistant is special: How can I install NacPy?	``` bash pip install network_a Initialization	Initialization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru
1	device identification	// The assistant is special: In which ways can I do?	!Parameter Name! !Detail General concepts	General concepts	NO	meta-llama/Meta-Llama-3.1-T0B-Instru
2	scopes list	// The assistant is special: Can you tell me all the ... #!# Network Device Status scopes?	** General concepts	General concepts	NO	meta-llama/Meta-Llama-3.1-T0B-Instru
3	qod session list	// The assistant is special: Can you list all of the av. #!# Network Code QoS prof. QoS	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
4	qod life cycle	// The assistant is special: Can you display the life ... #!# QoS Session Life Cycle A/QoS QoS	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
5	nac localization capabilities	// The assistant is special: Which localization needs... ?!#Feature?! !Description!! Localization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
6	location parameter info	// The assistant is special: What is the maximum ... If the maximum age parameter? Localization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
7	location parameter default values	// The assistant is special: In case maximum age p... If the maximum age parameter? Localization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
8	slice definition	// The assistant is special: What is a Specialized Network, also kn. Specialized Networks	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
9	slice id	// The assistant is special: Can you provide some... A Network Identifier is a unique... Specialized Networks	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		

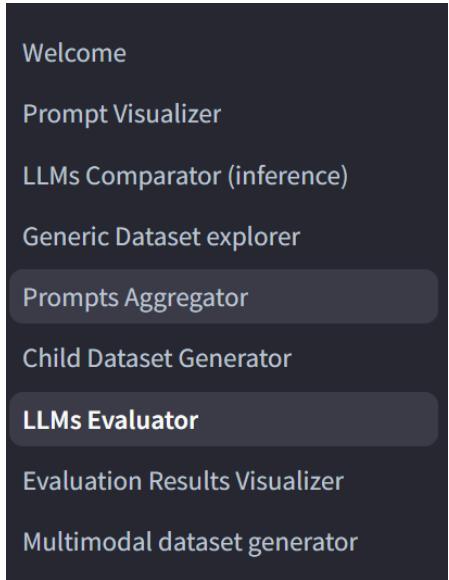
VS
Children

ID	System Prompt	User Prompt	LLM Answer	Category	Execute code	Model Name
0	Installation	// The assistant is special: Can you provide some... A Network Identifier is a unique... Specialized Networks	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
1	device identification	// The assistant is special: What is a Specialized Network, also kn. Specialized Networks	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
2	scopes list	// The assistant is special: Can you tell me all the ... #!# Network Device Status scopes?	** General concepts	General concepts	NO	meta-llama/Meta-Llama-3.1-T0B-Instru
3	qod session list	// The assistant is special: Can you list all of the av. #!# Network Code QoS prof. QoS	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
4	qod life cycle	// The assistant is special: Can you display the life ... #!# QoS Session Life Cycle A/QoS QoS	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
5	nac localization capabilities	// The assistant is special: Which localization needs... ?!#Feature?! !Description!! Localization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
6	location parameter info	// The assistant is special: What is the maximum ... If the maximum age parameter? Localization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
7	location parameter default values	// The assistant is special: In case maximum age p... If the maximum age parameter? Localization	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
8	slice definition	// The assistant is special: What is a Specialized Network, also kn. Specialized Networks	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		
9	slice id	// The assistant is special: Can you provide some... A Network Identifier is a unique... Specialized Networks	NO	meta-llama/Meta-Llama-3.1-T0B-Instru		

Evaluation dataset for each LLM

LLM Evaluation tool - functionalities

How you can define evaluations



```
EVAL_OPTIONS = [
    {
        'name': "NaC Evaluation",
        'metrics': ["evaluate_code", "code_bleu", "bert_score", "hf_bert_score", "llm_as_judge"],
        'function': run_nac_eval,
        'supported_judges': ['Runnable code evaluator', 'Hallucination detector', 'Contextual Relevance Evaluator']
    },
    {
        'name': "Long List/Tables evaluation",
        'metrics': ["the_fuzz", "hf_bert_score", "llm_as_judge"],
        'function': run_long_content_eval,
        'supported_judges': ['Missing items from list/table Evaluator']
    }
]
```

Define metrics with few lines of code

Metrics are implemented in a Python class called [Metric](#).

The developer has just to plug these functions in the evaluation function

```
self.methods = {
    'nltk_sentence_bleu': self.nltk_sentence_bleu,
    'nltk_meteor_score': self.nltk_meteor_score,
    'blanc_blanc_help_score': self.blanc_blanc_help_score,
    'blanc_blanc_tune_score': self.blanc_blanc_tune_score,
    'google_rouge_score': self.google_rouge_score,
    'bert_score': self.bert_score,
    'hf_bert_score': self.hf_bert_score,
    'hf_google_bleu_score': self.hf_google_bleu_score,
    'hf_exact_match_score': self.hf_exact_match_score,
    'hf_rouge_score': self.hf_rouge_score,
    'code_bleu': self.codebleu_calc_codebleu_score,
    'llm_as_judge': self.llm_as_judge,
    'evaluate_code': None,
    'the_fuzz': self.the_fuzz_score,
}
```

LLM Evaluation tool - functionalities

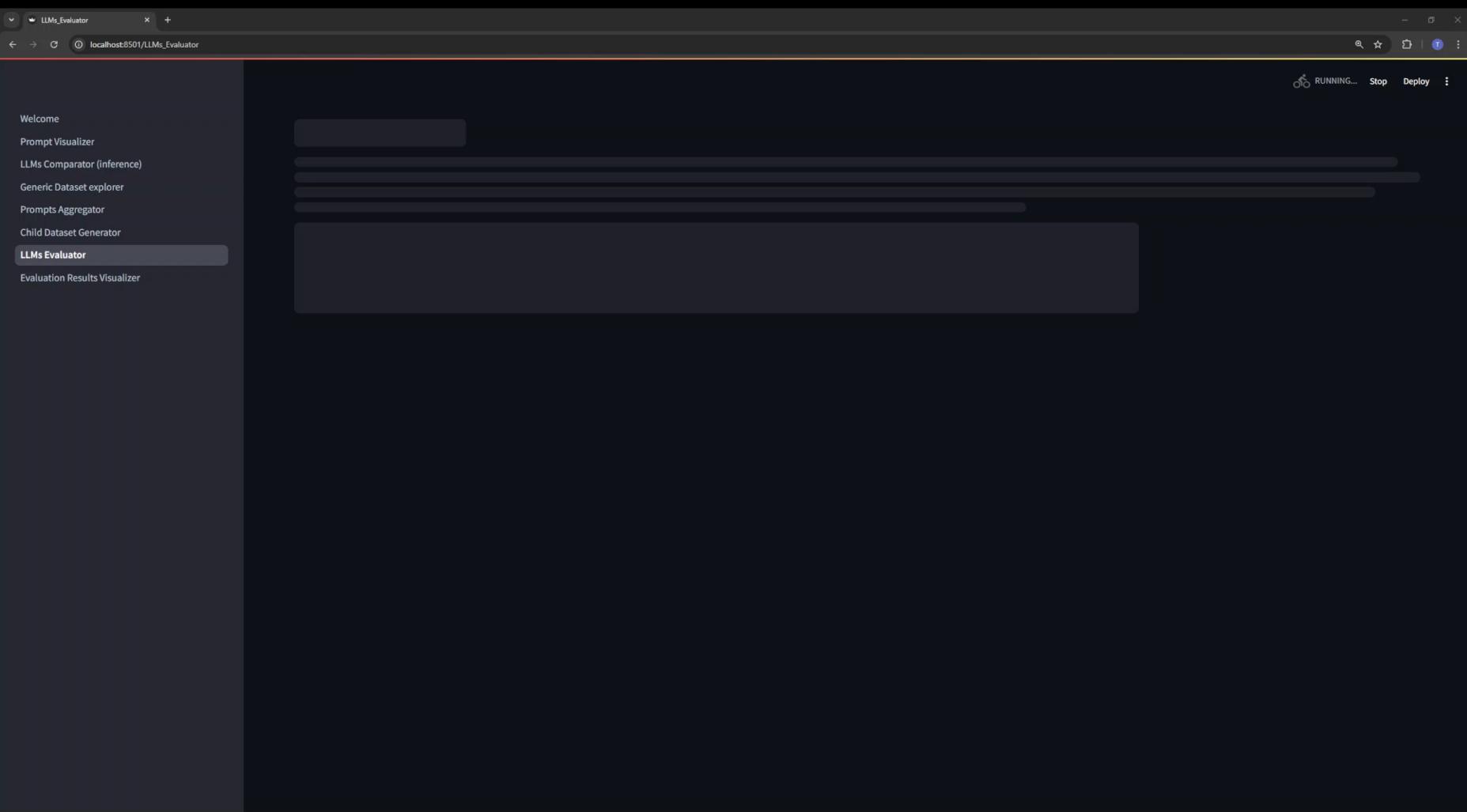
How you can define evaluations

Judges are defined with a YAML configuration

```
{  
    'name': "Long List/Tables evaluation",  
    'metrics': ["the_fuzz", "hf_bert_score", "llm_as_judge"],  
    'function': 'run_long_content_eval',  
    'supported_judges': ['Missing items from list/table Evaluator']  
}
```

Define metrics with few lines of code

```
- judge_name: Missing items from list/table Evaluator  
instructions: |  
<instructions>  
You are an impartial judge tasked with evaluating two lists or two tables: a reference list/table and a proposed list/table.  
Every list or table is obtained by asking to an LLM some information about a topic, which usually results in a long answer in form of either table or list.  
You are provided with a reference answer, provided by the best LLM, and a proposed answer provided by the LLM under evaluation.  
Your goal is to identify if the proposed answer is missing in any items presents in the reference answer.  
  
Your judgment must be provided with the following valid JSON schema:  
```json  
{"reason": str, "missing_item_count": int}
```  
  
</instructions>  
---  
Here is the user query:  
<user_query>  
{user_query}  
</user_query>  
  
Here is the reference example:  
<reference>  
{context_retrieved}  
</reference>  
content_to_judge: |  
Here is the content that you have to evaluate  
  
<llm_answer>  
{llm_answer}  
</llm_answer>  
---  
Remember to only provide the answer in json format without any preamble or introductory text.
```



NOKIA