LLM Benchmarks February 2025

Rinat Abdullin

22 August, 2025

Table of Contents

We focused on the Enterprise RAG Challenge in the past months, so this benchmark hasn't seen that much progress as we expected. The work on an interactive table and more test cases will happen in the next releases.

Still, we have quite a few topics to cover:

- AI coding tests imported into benchmark
- OpenAI: o3-mini and GPT-4.5
- Anthropic: Claude 3.7 with reasoning and without
- Qwen: QwQ 32B, Qwen Max, Qwen Plus
- Crisis of OpenAI SDK as a common standard for LLM APIs
- Insights from the Enterprise RAG Challenge

Let's get started!

Here is how the benchmark looks like at this moment. It tests abilities of LLMs to handle business and enterprise tasks that we have extracted from industry cases. It presents models with challenging tasks, but also allows them to benefit from the reasoning (by running custom chain-of-thought routines within the prompt).

At the moment OpenAI is at the top, closely followed up by Anthropic and DeepSeek models. It is amazing to see locally-capable models within the TOP-5.

It is even more **amazing to see a 70B model at the 5th place**. If the rumours of NVIDIA RTX PRO 6000 Blackwell[1]are true, one would need only one GPU card like this to run this model at native (to GPU) fp8 quantisation, while still having some spare space for KV caches and larger parallel contexts.

1. https://videocardz.com/newz/nvidia-rtx-pro-6000-blackwell-leaked-24064-cores-96gb-g7-memory-and-600w-double-flow-through-cooler

Model	bi	compliance	code	reason	Score	Features
openai/o3-mini-2025-01-31	45%	70%	100%	74%	76%	SO, Reason
anthropic/claude-3.7-sonnet:thinking	54%	32%	100%	67%	70%	Reason
openai/o1-2024-12-17	45%	70%	84%	67%	70%	SO, Reason
deepseek/deepseek-r1	27%	64%	100%	63%	66%	SO, Reason, Open
deepseek/deepseek-r1-distill-llama-70b	36%	32%	96%	56%	60%	Open
anthropic/claude-3.7-sonnet	45%	47%	65%	55%	56%	
openai/gpt-4o-2024-11-20	36%	55%	62%	55%	53%	SO
openai/gpt-4.5-preview-2025-02-27	45%	47%	62%	53%	51%	SO
deepseek/deepseek-chat	36%	47%	58%	49%	50%	SO, Open
openai/gpt-4o-2024-08-06	18%	62%	63%	52%	50%	SO
microsoft/phi-4	36%	62%	57%	48%	49%	
qwen/qwen-max	45%	45%	45%	50%	46%	
anthropic/claude-3.5-sonnet	36%	32%	57%	44%	43%	
meta-llama/llama-3.1-70b-instruct	36%	50%	44%	43%	42%	SO, Open
meta-llama/llama-3.3-70b-instruct	27%	50%	48%	41%	40%	SO, Open
google/gemini-2.0-flash-001	27%	24%	57%	38%	40%	
qwen/qwq-32b	36%	52%	41%	37%	40%	SO, Reason, Open
qwen/qwen-2.5-72b-instruct	27%	30%	47%	39%	39%	SO, Open
qwen/qwen2.5-32b-instruct	27%	20%	53%	36%	36%	Open
qwen/qwen-2.5-coder-32b-instruct	18%	35%	54%	39%	36%	SO, Open
meta-llama/llama-3.1-405b-instruct	18%	55%	40%	38%	35%	SO, Open
qwen/qwen-plus	18%	25%	40%	31%	31%	
mistralai/mixtral-8x22b-instruct	9%	27%	47%	28%	29%	SO, Open
openai/gpt-4o-mini-2024-07-18	9%	32%	41%	30%	28%	SO
mistral/mistral-small-24b-instruct-2501	27%	22%	33%	30%	27%	SO, Open
qwen/qwen-turbo	0%	15%	41%	20%	21%	
deepseek/deepseek-r1-distill-qwen-32b	9%	22%	29%	17%	21%	SO, Open
meta-llama/llama-3.2-3b-instruct	0%	17%	16%	11%	10%	SO, Open
mistralai/mistral-large-2411	0%	0%	0%	0%	0%	SO, Open
Averages	27%	39%	54%	42%		

1 New AI Coding Tasks imported into the benchmark

As we've mentioned earlier, this benchmark is not full, yet. We are still in the process of reviewing AI cases in our portfolio and importing them as new tests into the benchmark.

Here is an example of a fintech code comprehension benchmark. This test (one of many other tests) sounds like this:

```
Which specs would fail, if I add a new feature: after authorizing any transaction
larger than 3000, the system automatically blocks the card due to “Large Transaction
Risk.” We do not add new event type, just extend the existing command handler.

Source code: %SOURCE CODE%
```

By the way, take a note that we put the question before the contextual data in the prompt. This completely spoils the caching semantics, but makes the job easier for the models and improves the overall accuracy.

Source code for this specific task is a class implementing credit card state management. It comes with a list of executable event-driven specifications that verify all the behaviours. Entire file is 1100 lines of dense Python code.

All models are given a custom chain of thought routine that they can leverage within the prompt response. Here is, for example, how o1 correctly handles this tricky question:

```
{
  "short_thought_steps": [
    "Check specs that authorizeTransaction with amounts > 3000",
    "No spec has an authorization > 3000",
    "Hence no test scenario triggers the new block logic",
    "No spec fails"
  ],
  "failing_spec_names": []
}
```

And here is how the request is handled by Sonnet 3.7 (without reasoning):

```
{
    "short_thought_steps": [
        "1. The new feature will automatically block a card when a transaction over
3000 is authorized",
        "2. This means the authorize_transaction method will now have a side effect of
blocking the card",
        "3. Need to check all specs that authorize transactions over 3000",
        "4. Also need to check specs that expect certain events after authorization",
        "5. The spec 'authorize_transaction_success' authorizes a transaction of 2000,
which is below the threshold",
        "6. No other specs explicitly authorize transactions over 3000",
```

```
        "7. However, any spec that expects only a TransactionAuthorized event after
authorizing a large transaction would fail",
        "8. Let me create a test case in my mind: if we authorize a 4000 transaction,
we'd expect both TransactionAuthorized and CardBlocked events",
        "9. No existing specs test this scenario directly"
    ],
    "failing_spec_names": []
}
```

Coding tasks like that are a good proxy for understanding capabilities of models in software engineering. We will continue importing our AI cases into this benchmark to make it more relevant and representative.

Now, let's get to benchmarking actual models.

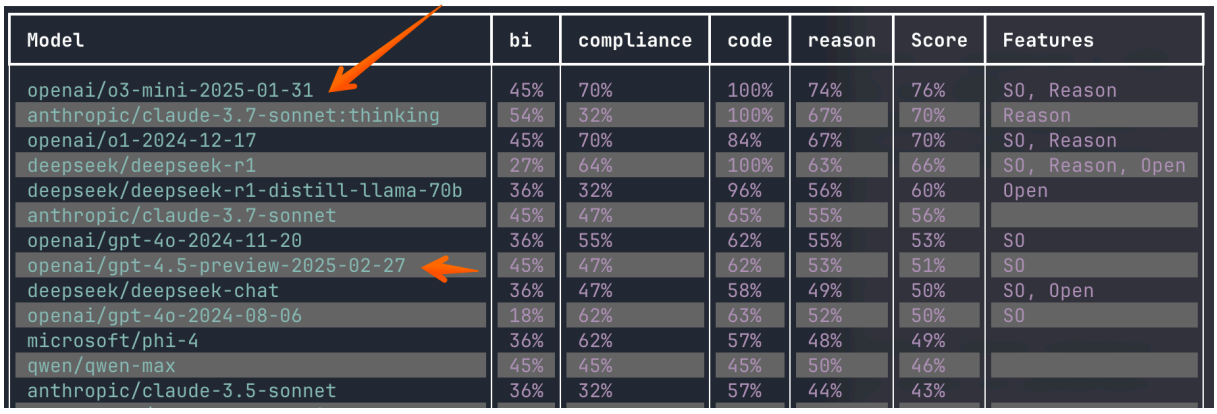
2 OpenAI: o3-mini and GPT-4.5

o3-mini model from OpenAI took the first place of our Benchmark (reasoning effort: medium). It currently is the best model available. It scored a perfect 100% in our current coding benchmark, 45% on BI benchmark and 70% on compliance benchmark.

Recently released GPT-4.5 model doesn't have reasoning. It is very thoughtful and capable of understanding nuances, but it didn't score that high, ending with 51% aggregate score.

Here is, for example, how this model handled (incorrectly) the coding task from above. It just didn't think things through to the end:

```
{
  "short_thought_steps": [
    "Identify specs involving transaction authorization.",
    "Check if transaction amounts exceed 3000.",
    "List specs that would fail due to the new blocking rule."
  ],
  "failing_spec_names": [
    "authorize_transaction_success"
  ]
}
```

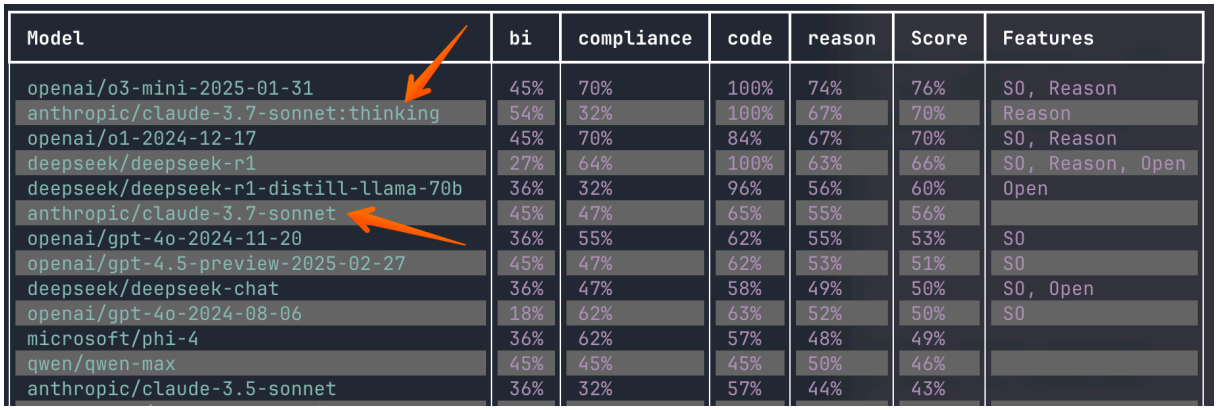


Model	bi	compliance	code	reason	Score	Features
openai/o3-mini-2025-01-31	45%	70%	100%	74%	76%	SO, Reason
anthropic/claude-3.7-sonnet:thinking	54%	32%	100%	67%	70%	Reason
openai/o1-2024-12-17	45%	70%	84%	67%	70%	SO, Reason
deepseek/deepseek-r1	27%	64%	100%	63%	66%	SO, Reason, Open
deepseek/deepseek-r1-distill-llama-70b	36%	32%	96%	56%	60%	Open
anthropic/claude-3.7-sonnet	45%	47%	65%	55%	56%	
openai/gpt-4o-2024-11-20	36%	55%	62%	55%	53%	SO
openai/gpt-4.5-preview-2025-02-27	45%	47%	62%	53%	51%	SO
deepseek/deepseek-chat	36%	47%	58%	49%	50%	SO, Open
openai/gpt-4o-2024-08-06	18%	62%	63%	52%	50%	SO
microsoft/phi-4	36%	62%	57%	48%	49%	
qwen/qwen-max	45%	45%	45%	50%	46%	
anthropic/claude-3.5-sonnet	36%	32%	57%	44%	43%	

3 Anthropic: Claude 3.7 Sonnet with reasoning and without

Anthropic models are usually praised by the developers and users for their pleasant and productive conversation format. However Anthropic wasn't really successful with advanced models recently (anybody remembers Opus?) until recently.

New Claude 3.7 Sonnet with thinking scored 70% on our benchmark (and a perfect 100% coding score), taking 2nd place. Its sibling model (without reasoning) took 8th place.



Model	bi	compliance	code	reason	Score	Features
openai/o3-mini-2025-01-31	45%	70%	100%	74%	76%	SO, Reason
anthropic/claude-3.7-sonnet:thinking	54%	32%	100%	67%	70%	Reason
openai/o1-2024-12-17	45%	70%	84%	67%	70%	SO, Reason
deepseek/deepseek-r1	27%	64%	100%	63%	66%	SO, Reason, Open
deepseek/deepseek-r1-distill-llama-70b	36%	32%	96%	56%	60%	Open
anthropic/claude-3.7-sonnet	45%	47%	65%	55%	56%	
openai/gpt-4o-2024-11-20	36%	55%	62%	55%	53%	SO
openai/gpt-4.5-preview-2025-02-27	45%	47%	62%	53%	51%	SO
deepseek/deepseek-chat	36%	47%	58%	49%	50%	SO, Open
openai/gpt-4o-2024-08-06	18%	62%	63%	52%	50%	SO
microsoft/phi-4	36%	62%	57%	48%	49%	
qwen/qwen-max	45%	45%	45%	50%	46%	
anthropic/claude-3.5-sonnet	36%	32%	57%	44%	43%	

We were running Claude 3.7 Sonnet in reasoning mode with a budget of 25k tokens and 80% of that dedicated to reasoning. It also had a few reasoning slots within the answer budget.

What makes the victory sweeter for Anthropic - their models still don't have a proper Structured Output mode (where schema is guaranteed in 100% of the cases by constrained decoding), yet the manage to respond with a correctly parseable JSON schema in every single case of this benchmark.

> Note, that your mileage with Anthropic may vary. For example, in some production cases, we are seeing JSON Schema error rates around 5-10%, which usually requires running schema repair routine.

4 Qwen Models

We have a bunch of new Qwen models in our benchmark, but will focus only on the two outstanding ones: Qwen Max and QwQ 32B.

Qwen Max is large-scale MoE model. It has decent scores, placing it on the 12th place. It is comparable with Microsoft Phi-4 and Claude 3.5 Sonnet.

Qwen QwQ-32B is a **locally capable model that pushes the state of the art in our benchmark**. This model raises the bar of the accuracy achievable by 32B models. The previous record was also held by a Qwen model (qwen-2.5-32B instruct), but the new model is a proper reasoning model.

Model	bi	compliance	code	reason	Score	Features
openai/o3-mini-2025-01-31	45%	70%	100%	74%	76%	SO, Reason
anthropic/claude-3.7-sonnet:thinking	54%	32%	100%	67%	70%	Reason
openai/o1-2024-12-17	45%	70%	84%	67%	70%	SO, Reason
deepseek/deepseek-r1	27%	64%	100%	63%	66%	SO, Reason, Open
deepseek/deepseek-r1-distill-llama-70b	36%	32%	96%	56%	60%	Open
anthropic/claude-3.7-sonnet	45%	47%	65%	55%	56%	
openai/gpt-4o-2024-11-20	36%	55%	62%	55%	53%	SO
openai/gpt-4.5-preview-2025-02-27	45%	47%	62%	53%	51%	SO
deepseek/deepseek-chat	36%	47%	58%	49%	50%	SO, Open
openai/gpt-4o-2024-08-06	18%	62%	63%	52%	50%	SO
microsoft/phi-4	36%	62%	57%	48%	49%	
qwen/qwen-max	45%	45%	45%	50%	46%	
anthropic/claude-3.5-sonnet	36%	32%	57%	44%	43%	
meta-llama/llama-3.1-70b-instruct	36%	50%	44%	43%	42%	SO, Open
meta-llama/llama-3.3-70b-instruct	27%	50%	48%	41%	40%	SO, Open
google/gemini-2.0-flash-001	27%	24%	57%	38%	40%	
qwen/qwq-32b	36%	52%	41%	37%	40%	SO, Reason, Open
qwen/qwen-2.5-72b-instruct	27%	30%	47%	39%	39%	SO, Open
qwen/qwen2.5-32b-instruct	27%	20%	53%	36%	36%	Open
qwen/qwen-2.5-coder-32b-instruct	18%	35%	54%	39%	36%	SO, Open
meta-llama/llama-3.1-405b-instruct	18%	55%	40%	38%	35%	SO, Open
qwen/qwen-plus	18%	25%	40%	31%	31%	
mistralai/mixtral-8x22b-instruct	9%	27%	47%	28%	29%	SO, Open
openai/gpt-4o-mini-2024-07-18	9%	32%	41%	30%	28%	SO
mistral/mistral-small-24b-instruct-2501	27%	22%	33%	30%	27%	SO, Open
qwen/qwen-turbo	0%	15%	41%	20%	21%	
deepseek/deepseek-r1-distill-qwen-32b	9%	22%	29%	17%	21%	SO, Open
meta-llama/llama-3.2-3b-instruct	0%	17%	16%	11%	10%	SO, Open
mistralai/mistral-large-2411	0%	0%	0%	0%	0%	SO, Open
google/gemma-3-27b-it:free	0%	0%	0%	0%	0%	SO

We tested **QwQ-32B** model by running it via OpenRouter from the Fireworks provider (selected, because they support StructuredOutputs). This experience cost us a few days of frustration and highlighted a general problem with the ecosystem of reasoning models.

5 Crisis of OpenAI SDK as a common standard for LLM APIs

At the moment of writing OpenAI API and Python SDK are a de-facto standard for interfacing with different models. It isn't the best standard, but nothing else did stick. Within the ecosystem you can use one library to access various servers: vLLM, ollama. Nvidia makes OpenAI compatible frontend for the Triton Inference server[2]. Even Google, after a long deliberation said that they are going to be OpenAI compatible[3].

Things work most of the time, until they don't.

For instance, when calling QwQ-32B model through OpenAI SDK library (which is guaranteed to be working by OpenRouter), we were getting `Error: 'NoneType' object is not iterable` error from deep inside the OpenAI SDK.

Root cause was a bit complex.

Modern reasoning models spend more tokens to think through the problem before providing answers. OpenAI reasoning models hide this thinking context, while open-source models make this explicit, by annotating with `<think>` tags in the response.

Now, OpenRouter probably tried to be smart and convenient. They are automatically putting think tokens into a special "reasoning" field of the response, while putting actual answer into "content". That is alone enough to mess things up in cases, where the model runs with StructuredOutput constrained decoding. Note, that the reasoning includes a proper JSON answer, while the content is empty:


```
{
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "logprobs": null,
      "message": {
        "content": "",
        "reasoning": "{ \"chain_of_thought\": [ \"To determine how many ...",
        "refusal": null,
        "role": "assistant"
      },
      "native_finish_reason": "stop"
    }
  ],
```


It took some time to identify the root cause, implement a custom client (fixing a problem like that on the fly), only to hit another edge case.

2. https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/client_guide/openai_readme.html
3. https://ai.google.dev/gemini-api/docs/openai

In a few percent of the cases, Qwen Qwq-32B, when queried through OpenRouter from Fireworks, would return a response like this one, where thinking output is text and the final content is also text, without any Structured Output in sight:


```
{
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "logprobs": null,
      "message": {
        "content": "```json\n{\n  \"short_thought_steps...```",
        "reasoning": "Okay, let me figure...",
        "refusal": null,
        "role": "assistant"
      },
      "native_finish_reason": "stop"
    }
  ],
```


Long story short, the ecosystem around standardised querying of reasoning models is currently a mess. There are too many gaps that leave interpretation open for the implementing parties. This leads to weird edge cases.

Coincidentally, OpenAI has just introduced its new Response API[4]. It is designed to replace Completions API and perhaps could bring in a better standard for working with reasoning LLMs.

4. https://platform.openai.com/docs/guides/responses-vs-chat-completions

6 Insights from the Enterprise RAG Challenge round 2 (ERCr2)

As you may have heard, we have just finished running a round 2 of our Enterprise RAG Challenge. We had more than 350 registrations before the competition. The challenge involved building a RAG system to answer 100 questions about 100 different annual reports of companies. The largest report had more than 1000 pages!

We have received a lot of submissions and more than a 100 filled in AI surveys, where teams explained in greater detail their aproaches and architectures. This made ERCr2 a big crowd-sourced research experiment on building precise Enterprise RAGs.

Here are some of the early insights that we have discovered together with the community.

First of all, **quality of data extraction is important for the overall accuracy**. Winning RAG solutions used the Docling document extraction library[5].

Second insight was: **if you have a good architecture, you can get high quality results even when using a tiny local LLM**.

Take a look at the winning architecture: *PDF parsing with heavily modified Docling library + Dense retrieval + Router + Parent Document Retrieval + SO CoT + SO reparser*

When evaluated with different LLM models, it obtained the following scores:

```
o3-mini         R: 83.8 | G: 81.8
llama3.3-70b    R: 83.9 | G: 72.8
llama-3.1 8b    R: 81.1 | G: 68.7

R - Retrieval score
G - Generation score
```

As you can see, less capable is the model - lower are the scores. But given the high quality of the Retrieval, the generation score doesn't drop as fast as we would've expected!

In other words, answer accuracy would be 68.7% using a good overall architecture and llama-3.1 8B. If we replaced it with 70B model, it would increase only by 4%. And if we replaced the local 70B model with the best reasoning model, Enterprise RAG Accuracy improves only by 9%.

This scoring was extracted from our benchmarks and applied to the Enterprise RAG Challenge. It turned out to be a good proxy for the potential quality that could be achieved with a given RAG Architecture. RAG accuracy depends on the quality of the retrieval component. But if the final RAG score is way lower than the quality of the retrieval - there are some low-hanging fruits that could be picked to improve the precision.

Our third insight: **reasoning patterns on top of Structured Outputs and Custom Chains of Thought were used by all top solutions**. They work even in cases, where Structured Outputs are not supported by the LLM serving (via constrained decoding) and schema must be repaired first.

5. https://ds4sd.github.io/docling/

Our fourth insight: **vector search is not completely dead, yet**. A high-quality dense retrieval mechanism was used to power the best solution. Although the second best solution (in prize leaderboard) didn't use vectors.

But in general, one thing that we've learned - small locally-capable models are capable of achieving a lot more than we initially thought. Open source LLMs are already a good competition to proprietary models, especially if you are using SO/CoT patterns to push them to their limits.

Perhaps, in year 2025 we will see a wider adoption of AI business systems that run locally and no longer compromise quality for the safety.