

LLM Benchmarks November 2024

Rinat Abdullin

22 August, 2025

Table of Contents

1 Claude 3.5 v2 Update and document extraction in manufacturing	4
2 GPT-4o from November 20 - TOP 3	7
3 Multiple Qwen models from Alibaba.....	9
4 LLM Benchmark v2	12
5 Gemini Experimental 1121 - Good, but “unobtainium”	13
6 Text-to-SQL Benchmark.....	14

November saw a lot of changes in TOP-10 of our LLM Benchmark. It also saw a few changes in how we build LLM-driven products. Let's get started.

- Update: Claude Sonnet 3.5 v2 - Small capability improvement and great PDF capability
- GPT-4o from November 20 - TOP 3!
- Qwen 2.5 Coder 32B Instruct - mediocre but pushes SotA!
- Qwen QwQ 32B Preview - too smart for its own good
- Gemini Experimental 1121 - decent, but hard to get
- Plans for LLM Benchmarks v2 - focus on cases and capabilities
- Text-to-SQL Benchmark

1 Claude 3.5 v2 Update and document extraction in manufacturing

In the previous LLM Benchmark we've written that Anthropic has achieved a small improvement in the capabilities of its Claude 3.5 v2. That improvement is relatively small, but not enough to put it into the TOP-10.

MODEL	CODE+ENG	CRM	DOCS	INTEGRATE	MARKETING	REASON	FINAL 🏆
12. Claude 3.5 Sonnet v2 ☁️	82	97	93	84	71	57	81
17. Claude 3.5 Sonnet v1 ☁️	72	83	89	87	80	58	78
18. Claude 3 Opus ☁️	69	88	100	74	76	58	77
37. Claude 3.5 Haiku ☁️	52	80	72	75	75	68	70
56. Claude 3 Haiku ☁️	64	69	64	75	75	35	64
61. Claude 3 Sonnet ☁️	72	41	74	74	78	28	61
65. Anthropic Claude Instant v1.2 ☁️	58	75	65	77	65	16	59
68. Anthropic Claude v2.0 ☁️	63	52	55	67	84	34	59
75. Anthropic Claude v2.1 ☁️	29	58	59	78	75	32	55

Yet, this **Anthropic Claude 3.5 Sonnet v2** is currently our first choice for data extraction projects (e.g. as a part of business automation in manufacturing industries). Why is it so?

For example, imagine that you need to carefully extract product specification data for 20000 electrical components out of 1000 data sheets. These PDFs could include complex tables and even charts. Extracted data could then be used for comparing company products to the products of competitors, offering equivalent components in inline ads or driving supply chain decisions.

Anthropic Claude 3.5 Sonnet v2 has two nice features that work well together:

- [Native PDF handling](#)¹
- [Prompt Caching](#)²

Native PDF Handling - we can now upload PDF files directly into the API along with the data extraction instructions. Under the hood, the Anthropic API will break the PDF down into pages and upload each

1. <https://docs.anthropic.com/en/docs/build-with-claude/pdf-support>

2. <https://docs.anthropic.com/en/docs/build-with-claude/prompt-caching>

page twice: as image and as text. This solution works well enough out of the box to replace previously complicated setups that used dedicated VLMs (Visual Language Models) running on local GPUs.

PDFs can consume a lot of tokens, especially when accompanied with a large system prompt. To speed up the processing, improve accuracy and lower costs we use two-level Prompt caching from Anthropic. This allows us to pay the full cost of PDF tokenisation only once.

Here is how our prompt can look for the data extraction:

1. System prompt: Your task is to extract product data from the PDF. Here is the schema (large schema) and company context.
2. Document prompt: Here is the PDF to extract the data from. It has multiple products. (large PDF)
3. Task: extract product X from the PDF.

This way we can extract multiple products from the single PDF (following the checklist pattern). System prompt (1) and Document prompt (2) will be cached between all extraction requests to the same PDF. System (1) will be cached between all requests for this type of PDF extraction in general.

Whenever a portion of the prompt is cached on the server - it costs less and runs faster. For example, 30-70% faster and 50-90% cheaper, as described in the [documentation of Anthropic](#)³. In data extraction cases, cost savings tend to be closer to the upper end of that range.

This is how things look in action: 'Cache creation' indicates when part of the prompt is stored in the cache, and 'Cache read' shows when the cached prompt is reused, saving time and money.

Timestamp	Log Level	Message	Cache Status
2024-11-21 14:19:20	INFO	_extract_2.py - Processing group with 18 items	
2024-11-21 14:19:36	INFO	_extract_2.py - Anthropic 20241022: 16.12s. Input: 606. Output: 562. Cache creation: 13543.	Cache creation
2024-11-21 14:19:48	INFO	_extract_2.py - Anthropic 20241022: 11.18s. Input: 605. Output: 547. Cache read: 13543.	Cache read
2024-11-21 14:19:59	INFO	_extract_2.py - Anthropic 20241022: 11.27s. Input: 605. Output: 548. Cache read: 13543.	Cache read
2024-11-21 14:20:12	INFO	_extract_2.py - Anthropic 20241022: 12.11s. Input: 609. Output: 571. Cache read: 13543.	Cache read
2024-11-21 14:20:25	INFO	_extract_2.py - Anthropic 20241022: 13.60s. Input: 608. Output: 570. Cache read: 13543.	Cache read
2024-11-21 14:20:39	INFO	_extract_2.py - Anthropic 20241022: 13.30s. Input: 608. Output: 593. Cache read: 13543.	Cache read
2024-11-21 14:20:51	INFO	_extract_2.py - Anthropic 20241022: 12.06s. Input: 606. Output: 557. Cache read: 13543.	Cache read
2024-11-21 14:21:05	INFO	_extract_2.py - Anthropic 20241022: 14.16s. Input: 605. Output: 555. Cache read: 13543.	Cache read
2024-11-21 14:21:40	INFO	_extract_2.py - Anthropic 20241022: 34.85s. Input: 605. Output: 554. Cache read: 13543.	Cache read
2024-11-21 14:21:54	INFO	_extract_2.py - Anthropic 20241022: 13.37s. Input: 609. Output: 591. Cache read: 13543.	Cache read
2024-11-21 14:22:06	INFO	_extract_2.py - Anthropic 20241022: 11.75s. Input: 608. Output: 562. Cache read: 13543.	Cache read
2024-11-21 14:22:44	INFO	_extract_2.py - Anthropic 20241022: 38.33s. Input: 608. Output: 577. Cache read: 13543.	Cache read
2024-11-21 14:23:09	INFO	_extract_2.py - Anthropic 20241022: 25.00s. Input: 606. Output: 547. Cache read: 13543.	Cache read
2024-11-21 14:23:25	INFO	_extract_2.py - Anthropic 20241022: 15.43s. Input: 605. Output: 549. Cache read: 13543.	Cache read
2024-11-21 14:23:42	INFO	_extract_2.py - Anthropic 20241022: 17.09s. Input: 605. Output: 574. Cache read: 13543.	Cache read
2024-11-21 14:23:56	INFO	_extract_2.py - Anthropic 20241022: 13.91s. Input: 609. Output: 570. Cache read: 13543.	Cache read
2024-11-21 14:24:44	INFO	_extract_2.py - Anthropic 20241022: 47.59s. Input: 608. Output: 570. Cache read: 13543.	Cache read
2024-11-21 14:24:58	INFO	_extract_2.py - Anthropic 20241022: 14.26s. Input: 608. Output: 570. Cache read: 13543.	Cache read
2024-11-21 14:24:58	INFO	_extract_2.py - Processing group with 6 items	
2024-11-21 14:25:18	INFO	_extract_2.py - Anthropic 20241022: 19.57s. Input: 622. Output: 593. Cache creation: 6729. Cache read: 4608.	Cache creation
2024-11-21 14:25:29	INFO	_extract_2.py - Anthropic 20241022: 11.04s. Input: 623. Output: 604. Cache read: 11337.	Cache read
2024-11-21 14:25:41	INFO	_extract_2.py - Anthropic 20241022: 11.87s. Input: 621. Output: 601. Cache read: 11337.	Cache read
2024-11-21 14:25:54	INFO	_extract_2.py - Anthropic 20241022: 12.96s. Input: 621. Output: 585. Cache read: 11337.	Cache read
2024-11-21 14:26:07	INFO	_extract_2.py - Anthropic 20241022: 12.29s. Input: 621. Output: 607. Cache read: 11337.	Cache read
2024-11-21 14:26:42	INFO	_extract_2.py - Anthropic 20241022: 35.15s. Input: 621. Output: 609. Cache read: 11337.	Cache read
2024-11-21 14:26:42	INFO	_extract_2.py - Processing group with 20 items	
2024-11-21 14:27:15	INFO	_extract_2.py - Anthropic 20241022: 16.35s. Input: 648. Output: 687. Cache creation: 9666. Cache read: 4608.	Cache creation
2024-11-21 14:27:33	INFO	_extract_2.py - Anthropic 20241022: 17.26s. Input: 647. Output: 689. Cache read: 14274.	Cache read
2024-11-21 14:27:47	INFO	_extract_2.py - Anthropic 20241022: 14.52s. Input: 647. Output: 680. Cache read: 14274.	Cache read
2024-11-21 14:28:02	INFO	_extract_2.py - Anthropic 20241022: 14.53s. Input: 650. Output: 709. Cache read: 14274.	Cache read

There is a small caveat. Anthropic models don't have Structured Output capability of OpenAI. So you would think that we can lose two amazing features:

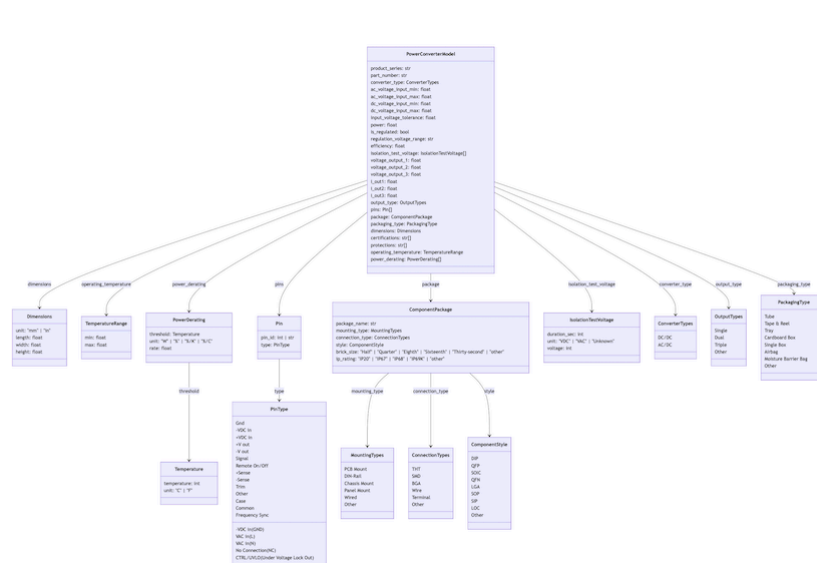
- Precise schema following
- Ability to hardcode custom chain-of-thought process that will drive LLM through the data extraction process.

3. <https://www.anthropic.com/news/prompt-caching>

However, this is not the case! Structured Output is just an inference capability that drives constrained decoding (token selection) to follow the schema precisely. A capable LLM will be able to extract even a complex structure without it. And while doing so, it will follow chain-of-thought process encoded in the schema definition.

Anthropic Claude 3.5 Sonnet v2 certainly can perform that. And in 5-7% of cases that return slightly invalid schema, we can pass results to GPT-4o for the schema repair.

For reference, here is an example of Structured Output definition from one of the projects (image quality was lowered intentionally).



2 GPT-4o from November 20 - TOP 3

OpenAI didn't bother to publish a proper announcement for this model (`gpt-4o-2024-11-20` in the API). They have just tweeted the update:



GPT-4o got an update 🎉

The model's creative writing ability has leveled up—more natural, engaging, and tailored writing to improve relevance & readability.

It's also better at working with uploaded files, providing deeper insights & more thorough responses.

7:01 PM · Nov 20, 2024 · **3.3M** Views

The model deserves a special mention in our benchmarks. Compared to the previous GPT-4o v2/2024-08-06, the model shows noticeable improvement, especially in the Reason category.

MODEL	CODE+ENG	CRM	DOCS	INTEGRATE	MARKETING	REASON	FINAL 🏆	COST
1. GPT o1-preview v1/2024-09-12 ☁	95	92	94	95	88	87	92	52.32 €
2. GPT o1-mini v1/2024-09-12 ☁	93	96	94	83	82	87	89	8.15 €
3. GPT-4o v3/2024-11-20 ☁	86	97	94	95	88	72	89	0.63 €
4. GPT-4o v1/2024-05-13 ☁	90	96	100	92	78	74	88	1.21 €
5. Google Gemini 1.5 Pro v2 ☁	86	97	94	99	78	74	88	1.00 €
6. GPT-4 Turbo v5/2024-04-09 ☁	86	99	98	96	88	43	85	1.45 €
7. Google Gemini Exp 1121 ☁	70	97	97	95	72	72	84	1.89 €
8. GPT-4o v2/2024-08-06 ☁	90	84	97	86	82	59	83	0.63 €
9. Google Gemini 1.5 Pro 0801 ☁	84	92	79	100	70	74	83	0.90 €
10. Qwen 2.5 72B Instruct 🚩	79	92	94	97	71	59	82	0.10 €
11. Llama 3.1 405B Hermes 3 🐼	68	93	89	98	88	53	81	0.54 €
12. Claude 3.5 Sonnet v2 ☁	82	97	93	84	71	57	81	0.95 €
13. GPT-4 v1/0314 ☁	90	88	98	73	88	45	80	7.04 €
14. X-AI Grok 2 🚩	63	93	87	90	88	58	80	1.03 €

You can also note the usual pattern of OpenAI with the models:

1. First, they release a new powerful model (GPT-4o v1 in this case)
2. Then they release the next model in the same family that is much cheaper to run
3. Finally, they improve the model to be better, while still running at lower costs.

3 Multiple Qwen models from Alibaba

Qwen 2.5 Coder 32B Instruct is a new model in Qwen family. It will first make you sad and then—glad.

The model itself can be [downloaded from HuggingFace](#)⁴ and run locally on your hardware.

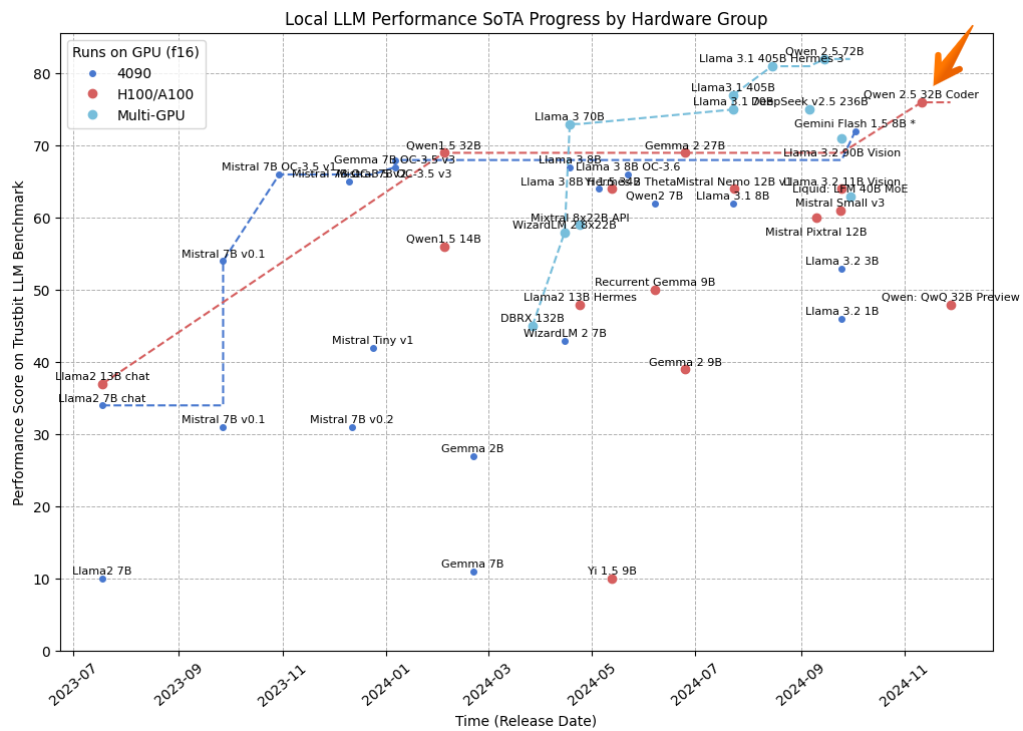
The sad part is that this coding model performed poorly on our Code+Eng category of tasks. It was able to handle coding tasks, but failed to deal with more complex code review and analysis challenges. Besides, its reasoning is generally quite low - 46.

MODEL	CODE+ENG	CRM	DOCS	INTEGRATE	MARKETING	REASON	FINAL 🏆	COST
10. Qwen 2.5 72B Instruct ⚠️	79	92	94	97	71	59	82	0.10 €
23. Qwen 2.5 32B Coder Instruct ⚠️	43	94	98	98	76	46	76	0.05 €
41. Qwen1.5 32B Chat f16 ⚠️	70	90	82	76	78	20	69	0.97 €
51. Qwen 2.5 7B Instruct ⚠️	48	77	80	68	69	47	65	0.07 €
60. Qwen2 7B Instruct f32 ⚠️	50	81	81	61	66	31	62	0.46 €
70. Qwen1.5 7B Chat f16 ⚠️	56	81	60	56	60	36	58	0.29 €
73. Qwen1.5 14B Chat f16 ⚠️	50	58	51	72	84	22	56	0.36 €
83. Qwen: QwQ 32B Preview ⚠️	43	32	74	52	48	40	48	0.05 €

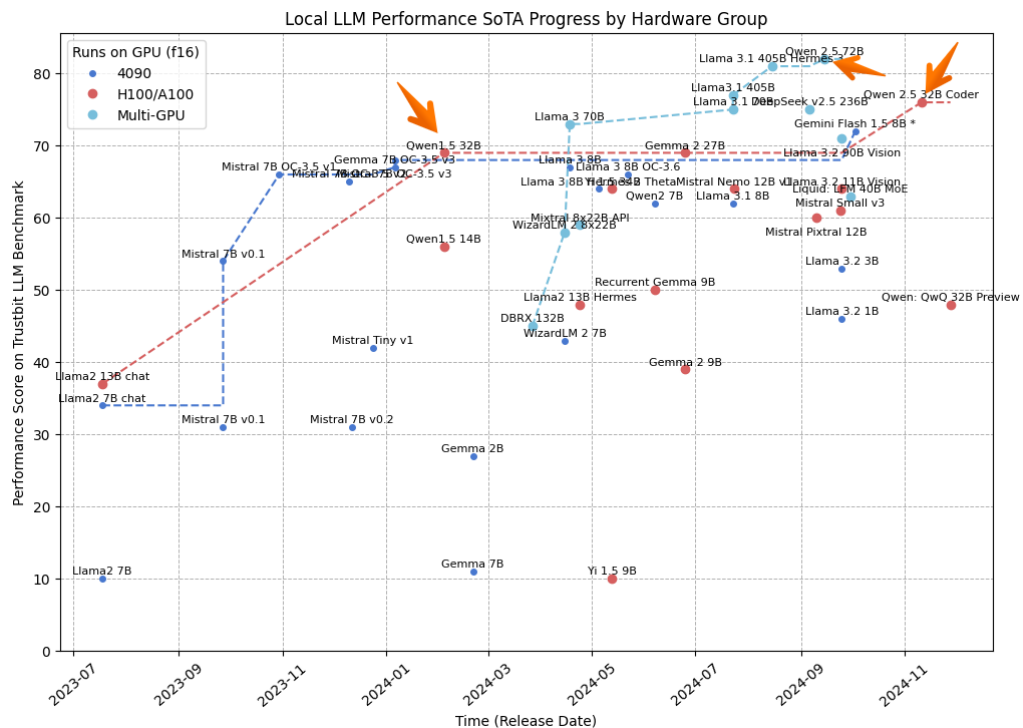
What would you expect from a model that is called “Coder”, right? And actually in coding this model is quite good. This model performed as well as Sonnet 3.5 in coding-only benchmark for complex text-to-SQL tasks (more about that later).

What is so good about this model, then? **This coding-oriented model represents a new quality improvement for local models in category “It can run on A100/H100 GPU”!**

4. <https://huggingface.co/Qwen/Qwen2.5-Coder-32B-Instruct>



By the way, it is interesting to note that a few other big quality improvements that pushed State of the Art for local models were also driven by Qwen.



It is also interesting that “o1-killer” from Qwen didn’t score that high on our benchmark. [Qwen: QwQ 32B Preview](#)⁵ was designed to push state of the art in reasoning capabilities. According to some benchmarks it did succeed. However, **it doesn’t look like a fit for product tasks and business automation**. Why? It talks too much and doesn’t follow the instructions.

For example, given the prompt below, that is also reinforced by a couple of samples:

You extract product properties from provided text. Respond in format: "number unit" or "N/A" if can't determine. Strip quotes, thousands separators and comments.

The model will tend to start the response this way:

Alright, I've got this text about an electric screwdriver,...

Even the tiny `mistral-7b-instruct-f16` would answer precisely something like `1300 rpm`.

This might seem like an unfair comparison for QwQ against a top model [o1-preview](#)⁶. o1 has a chance to reason in private before providing its response (it uses reasoning tokens for that).

To make things more fair for the new generations of reasoning models, we will change things a bit in the **next major update of our benchmark - models will be allowed to reason before providing an answer**. Models that think too much will be natively penalised by their cost and huge latency.

5. <https://huggingface.co/Qwen/QwQ-32B-Preview>

6. <https://openai.com/index/introducing-openai-o1-preview/>

4 LLM Benchmark v2

We've been running current version of the benchmark without major changes for almost a year and a half. Changes were avoided to keep benchmark results comparable between models and test runs.

However, a lot has changed in the landscape since July 2023:

- Structured Outputs - allow us to define precise response format and even drive custom chain-of-thought for the complex tasks.
- Multi-modal language models can handle images and audio in addition to text input. Image inputs are used heavily in document extraction.
- Prompt caching shifts perspective for building RAG systems, running complex checklists or extracting data from a lot of documents.
- New reasoning models allow us to push model performance forward by breaking down complex tasks into small steps and then investing (paid) time to think through them.


In addition to that, we've gained a lot more insights in building LLM-driven systems and added more cases to our AI portfolio.

It is time for a big refresh. The work on the TimeToAct LLM Benchmark v2 has already started. We are expecting to publish the first draft report early next year.

The V2 benchmark will keep the foundations from v1 but will focus more on concrete AI Cases and new model capabilities. More charts are to be expected, too.

5 Gemini Experimental 1121 - Good, but “unobtainium”

Gemini Experimental 1121 is a new prototype model from Google. It is currently available in test environments like AI Studio or OpenRouter. This model doesn’t push state of the art for Gemini, but **proves that the presence of Google in TOP-10 is not a lucky coincidence**. It is the third Gemini model to be in TOP-10.

MODEL	CODE+ENG	CRM	DOCS	INTEGRATE	MARKETING	REASON	FINAL 🏆	COST
1. GPT o1-preview v1/2024-09-12 ☁	95	92	94	95	88	87	92	52.32 €
2. GPT o1-mini v1/2024-09-12 ☁	93	96	94	83	82	87	89	8.15 €
3. GPT-4o v3/2024-11-20 ☁	86	97	94	95	88	72	89	0.63 €
4. GPT-4o v1/2024-05-13 ☁	90	96	100	92	78	74	88	1.21 €
5. Google Gemini 1.5 Pro v2 ☁	86	97	94	99	78	74	88	1.00 €
6. GPT-4 Turbo v5/2024-04-09 ☁	86	99	98	96	88	43	85	2.45 €
7. Google Gemini Exp 1121 ☁ 	70	97	97	95	72	72	84	0.89 €
8. GPT-4o v2/2024-08-06 ☁	90	84	97	86	82	59	83	0.63 €
9. Google Gemini 1.5 Pro 0801 ☁	84	92	79	100	70	74	83	0.90 €
10. Qwen 2.5 72B Instruct ⚠	79	92	94	97	71	59	82	0.10 €
11. Llama 3.1 405B Hermes 3 🐼	68	93	89	98	88	53	81	0.54 €
12. Claude 3.5 Sonnet v2 ☁	82	97	93	84	71	57	81	0.95 €
13. GPT-4 v1/0314 ☁	90	88	98	73	88	45	80	7.04 €
14. X-AI Grok 2 ⚠	63	93	87	90	88	58	80	1.03 €

However, this model is currently impossible to use. It is provided for free but is **heavily rate limited**. It took 3 days and multiple API keys just to run a few hundred evals from our benchmark.

6 Text-to-SQL Benchmark

Neo4j has published a video from its NODES24 conference about benchmarking different LLMs in text-to-SQL and text-to-Cypher tasks.

An example of a Text-to-SQL task is when an LLM is used to translate human request into a complex query against company SQL database. It is used for self-service reporting. Text-to-Cypher is the same, but runs queries against graph databases like Neo4j.

The research and presentation was done in partnership with two companies from the TIMETOACT GROUP: [X-Integrate](https://www.x-integrate.com/en)⁷ and [TimeToAct Austria](https://www.timetoact-group.at)⁸.

The most important slide of the presentation is the one below. It shows accuracy with which different LLMs have generated queries for a complex database. This DB held information about technical and organisational dependencies in the company for the purposes of risk management.

LLM	Cypher Basic	Cypher Full	SQL Basic	SQL Full
GPT 4-0613	0.48	0.48	0.24	0.71
GPT 4 Turbo 2024-04-09	0.24	0.61	0.19	0.80
GPT 4o 2024-08-06	0.56	0.86	0.24	1.00
Mistral Large	0.33	0.71	0.14	0.86
Qwen 2.5-72b-instruct	0.29	0.56	0.24	0.86
Anthropic Claude 3.5 Sonnet	0.71	1.00	0.14	0.90
Hermes 3 Llama 3.1 405b	0.48	0.62	0.24	0.86
Gemini Pro 1.5 0409	0.33	0.57	0.24	0.91

“Basic” scores are the scores without any performance optimisations, while “Full” scores employ a range of performance optimisations to boost the accuracy of query generation.

You can learn more about these optimisations (and about the benchmark) by watching the presentation online on YouTube: <https://youtu.be/YbJVq8ZOsaM?si=r8AjLduNtXcdfq7L>

Some of these text-to-query tasks will even be included in our upcoming LLM v2 benchmark.

7. <https://www.x-integrate.com/en>

8. <https://www.timetoact-group.at>