

SPECIAL ANNIVERSARY EDITION
LET'S MEET FOR THE 10TH TIME!



bigdata
TECHNOLOGY WARSAW SUMMIT



APRIL 10TH-11TH
2024

Your personal LLM and RAG-backed Data Copilot – lessons learned

Marek Wiewiórka, Phd

GetInData | Part of Xebia

About me



Marek Wiewiórka
PhD | Chief Data Architect at
GetInData | Cloud and Big Data A...



- Chief Data Architect @**GetInData | Part of Xebia**
- PhD(2023), Research Assistant at the Warsaw University of Technology
- Personally, a keen long-distance runner and gravel bike enthusiast

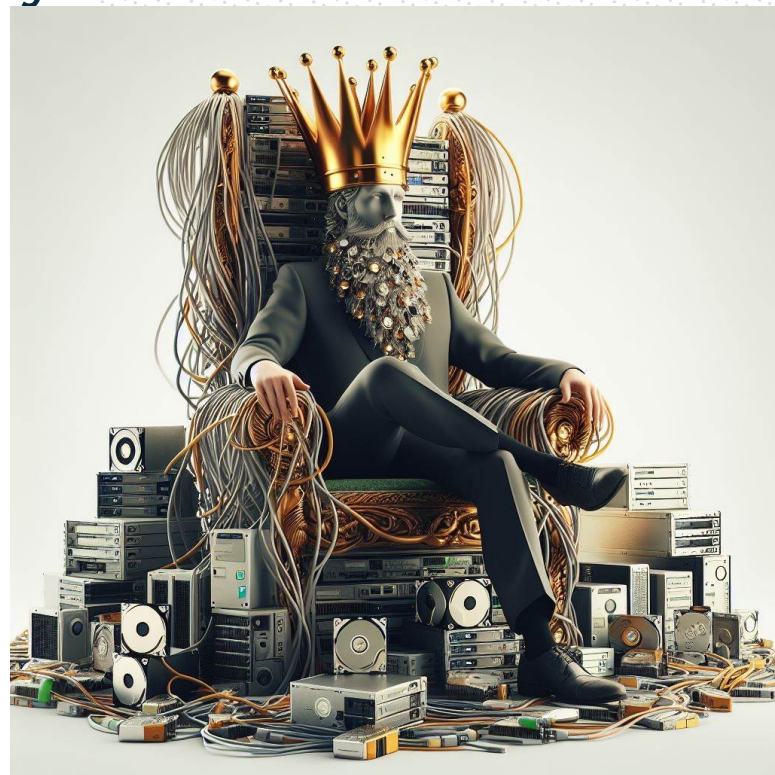
...how to turn best practices into AI coding assistant

1. Why do we need yet another (open-source) Copilot?
2. How can we build one?
3. Architecture and evaluation
4. DEMO



(Data) Context is king!

- Explicit and precise data context of your whole data platform
- Data transformation codebase
- Data models with comments and table relationships
- Other user queries
- Lineage and human curated dataset descriptions from data catalogs



Customized and specialized models are the future.



We believe that in the future, the vast majority of organizations will develop customized models that are personalized to their industry, business, or use case.

With a variety of techniques available to build a custom model, organizations of all sizes can develop personalized models to realize more meaningful, specific impact from their AI implementations. The key is to clearly scope the use case, design and implement evaluation systems, choose the right techniques, and be prepared to iterate over time for the model to reach optimal performance.

With OpenAI, most organizations can see meaningful results quickly with the self-serve fine-tuning API. For any organizations that need to more deeply fine-tune their models or imbue new, domain-specific knowledge into the model, our Custom Model programs can help.

Build high-quality generative AI applications with DBRX customized for your unique data

by [Jonathan Frankle](#), [Ali Ghodsi](#), [Naveen Rao](#), [Hanlin Tang](#), [Abhinav Venigalla](#) and [Matei Zaharia](#)

March 27, 2024 in [Company Blog](#)

Share this post

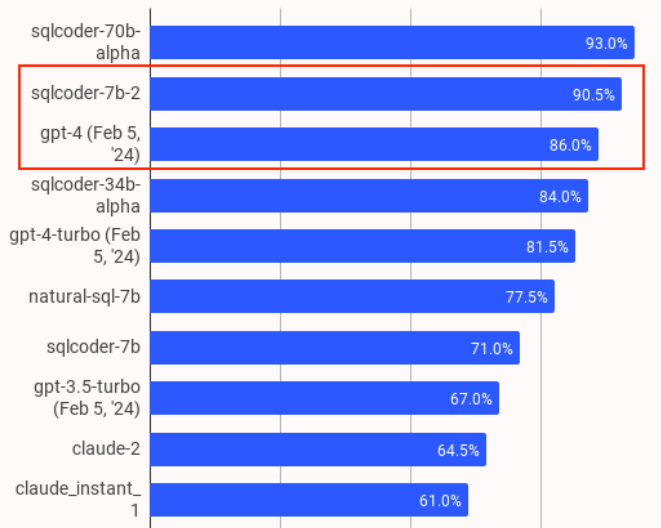


Databricks' mission is to deliver data intelligence to every enterprise by allowing organizations to understand and use their unique data to build their own AI systems. Today, we are excited to advance our mission by open sourcing DBRX, a general purpose large language model (LLM) built by our [Mosaic Research](#) team that outperforms all established open source models on standard benchmarks. We believe that pushing the boundary of open source models enables generative AI for all enterprises that is customizable and transparent.

sqlcoder-7b and others

- Many other small (7-34b) models licensed for commercial use, e.g. :
 - ✓ starcoder2
 - ✓ dolphincoder
 - ✓ deepseek-coder
 - ✓ opencodeinterpreter

Percentage of correctly generated SQL queries on novel schemas not seen in training (n = 200), with 4 beams



When quantized can be even run locally!

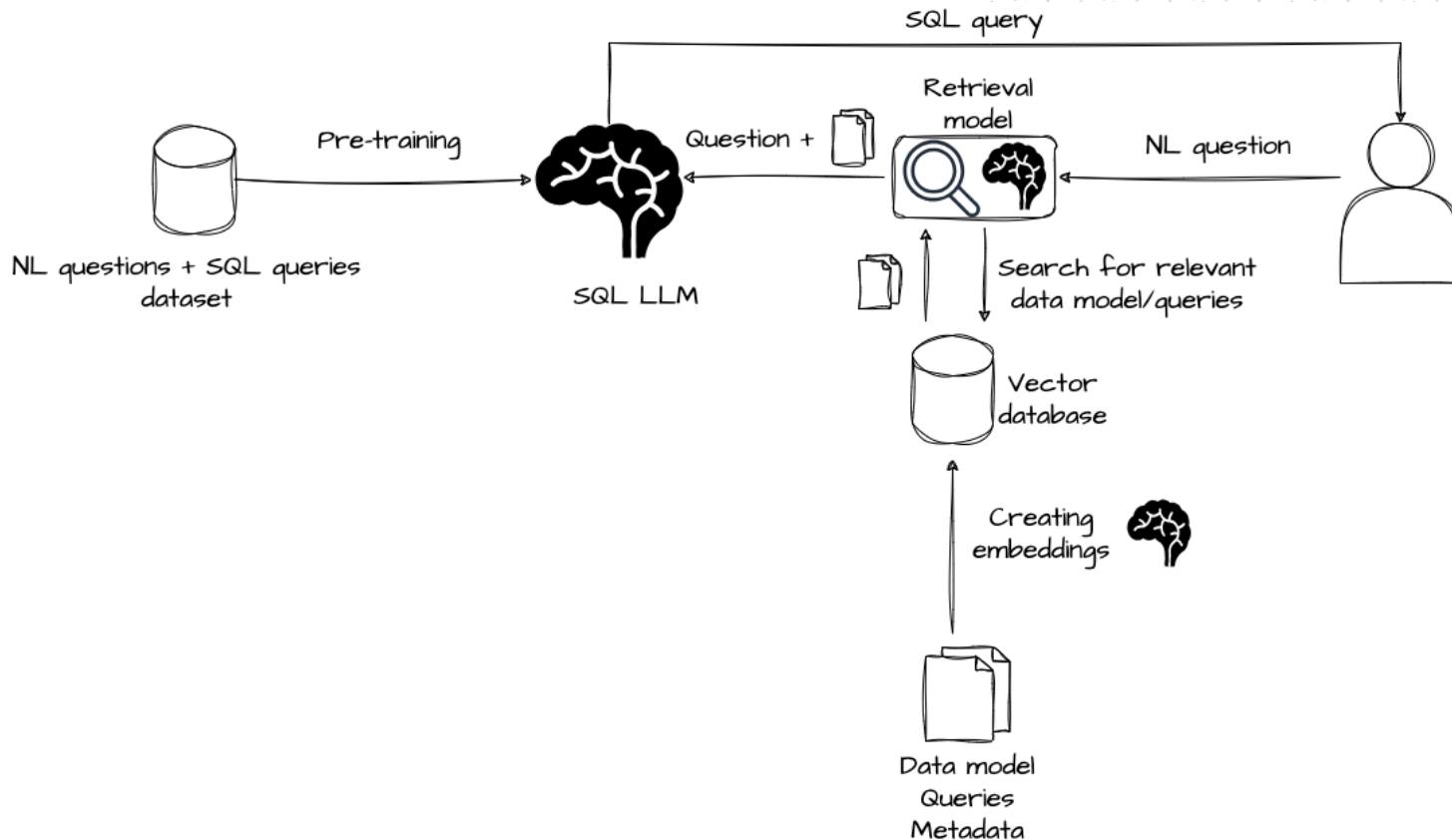
How turn your best practices into Copilots ?

- Vector database as a knowledge base - what ?
- Prompts as instructions following best practices - how ?
- LLM to combine both...

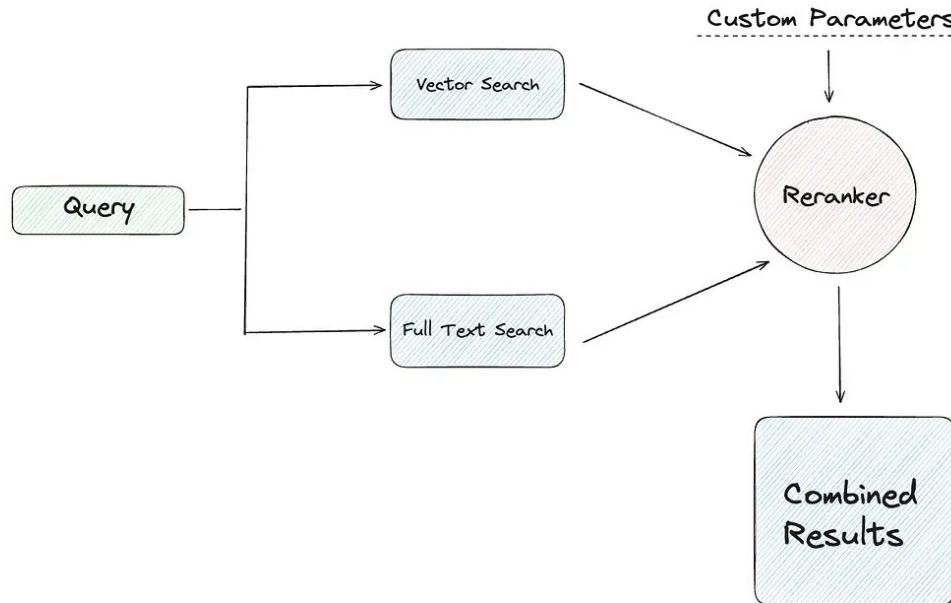
Retrieval-Augmented Generation(RAG)



RAG for Text-to-SQL

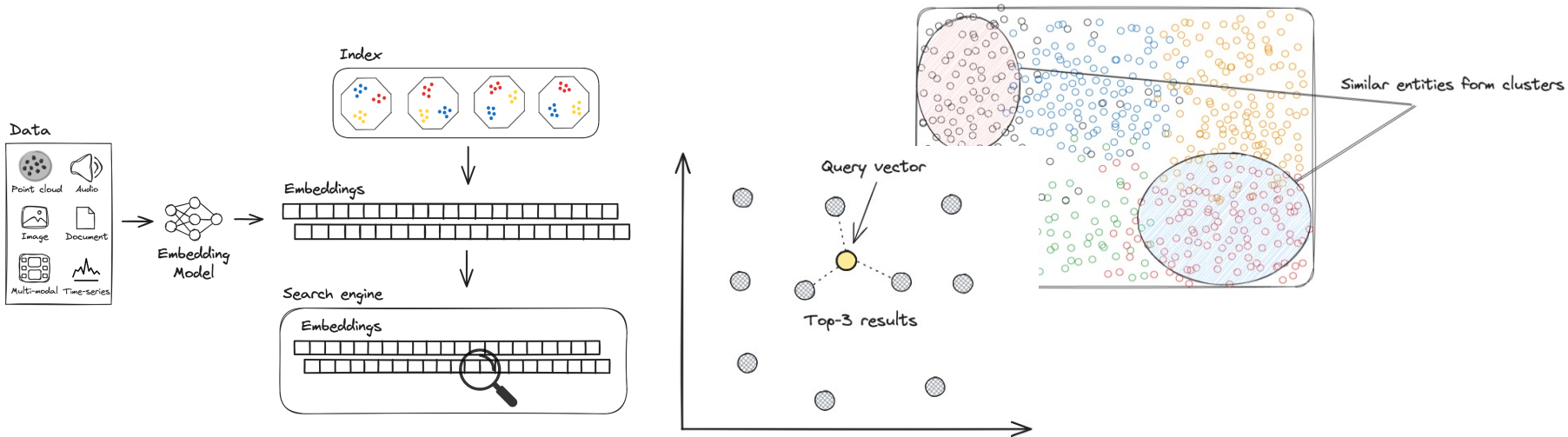


- combination of keyword and vector search



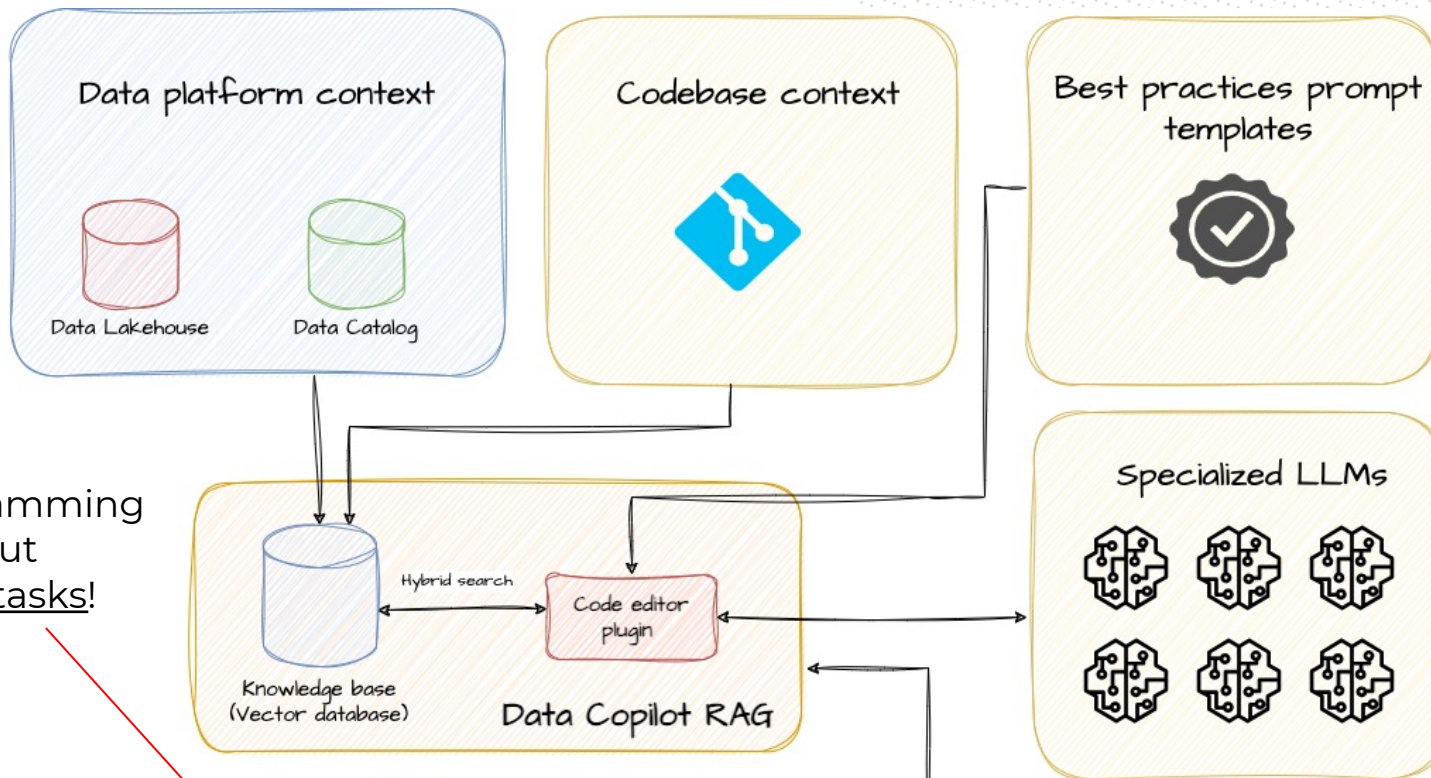
Vector search

- a technique used to search for similar items based on their vector representations, called embeddings
- Approximate Nearest Neighbours algorithms



Data Copilot RAG architecture

Data programming
is more about
repeatable tasks!



```
/dbtSQL jaffle How many products by type and  
supplier that can decay and sort by product count  
desc?  
+ Add Context Enter  
phind-codellama:34b + Loading (25%)
```

GID Data Copilot (GDC)

- An extensible AI programming assistant for **SQL** and **dbt** code
- Powered by:
 - **Large Language Models** (SOTA LLMs)
 - Robust **Retrieval Augmented Generation** (RAG) architecture
 - **Hybrid search** techniques
 - **Fast Vector Database**
 - **Curated Prompts**
 - **Builtin Data commands**

The screenshot displays the GID Data Copilot interface. The main window shows a SQL query editor with the following code:

```
1 with perishable_products as (  
2  
3  
4 SELECT  
5   p.product_type,  
6   s.supply_name,  
7   COUNT(p.product_id) as product_count  
8  
9 FROM  
10  {{ ref('products') }} as p  
11 JOIN  
12  {{ ref('supplies') }} as s  
13 ON  
14  p.product_id = s.product_id  
15 WHERE  
16  s.is_perishable_supply = TRUE  
17 GROUP BY  
18  p.product_type,  
19  s.supply_name  
20 )  
21  
22 SELECT * FROM perishable_products  
23 ORDER BY product_count DESC
```

Annotations in the image include:

- Data context with RAG**: Points to the SQL query in the editor.
- Hybrid search**: Points to the chat window.
- Data "slash" commands**: Points to the terminal window.

The chat window contains the following prompt and response:

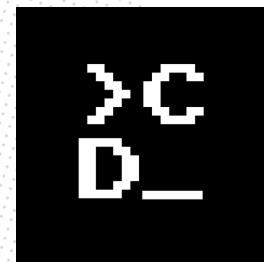
@RAGDB jaffiedb what column and table to use if I need to find products that can decay?

The information about whether a product can decay or not (i.e., if it is perishable) can be found in the 'supplies' table under the 'is_perishable_supply' column. The

The terminal window shows the following execution logs:

```
(base) = lakehouse-copilot-demo git:(duckdb-setup) x dbt run -s perishable_products  
12:32:23 Running with dbt=1.7.8  
12:32:24 Registered adapters: duckdb=1.7.2  
12:32:24 Found 14 models, 6 seeds, 25 tests, 0 exposures, 13 metrics, 773 macros, 0 groups, 6 semantic models  
12:32:24 Concurrency: 1 threads (target='dev')  
12:32:24  
12:32:24 1 of 1 START sql table model main.perishable_products ..... [RUN]  
12:32:24 1 of 1 OK created sql table model main.perishable_products ..... [OK in 0.14s]  
12:32:24  
12:32:24 Finished running 1 table model in 0 hours 0 minutes and 0.44 seconds (0.44s).  
12:32:24  
12:32:24 Completed successfully  
12:32:24
```

Continue - an open-source autopilot



- support for both tasks and tab autocompletion
- highly extensible
 - use any LLM model you wish - also **multiple, specialized models** for different languages or tasks
 - support for many **model providers**, such as Ollama, vLLM, LM Studio
 - custom **context providers** for more control over LLMs augmentation
 - custom **slash commands** that can combine own prompts, contexts and models for specialized, reusable tasks
- support for VSCode and JetBrains
- secure (i.e. can be run locally, on-premise or cloud VPC)
- translate your best practices into "slash data commands"

Continue - a custom context provider

```
const RagContextProvider: CustomContextProvider = {
  title: "ragdb",
  displayTitle: "RAGDB",
  description:
    "Retrieve DB schema from our vector database of internal documents",
  type: "normal",
  getContextItems: async (
    query: string,
    extras: ContextProviderExtras
  ): Promise<ContextItem[]> => {
    console.info(extras.fullInput)
    const inputArray = extras.fullInput.split(' ');
    const db = inputArray[0];
    const userQuestion = inputArray.slice(2).join(' ');
    const response = await fetch("http://localhost:8000/retrieve" {
      method: "POST",
      headers: {
        'content-type': 'application/json;charset=UTF-8',
      },
      body: JSON.stringify({ query: userQuestion }),
    });

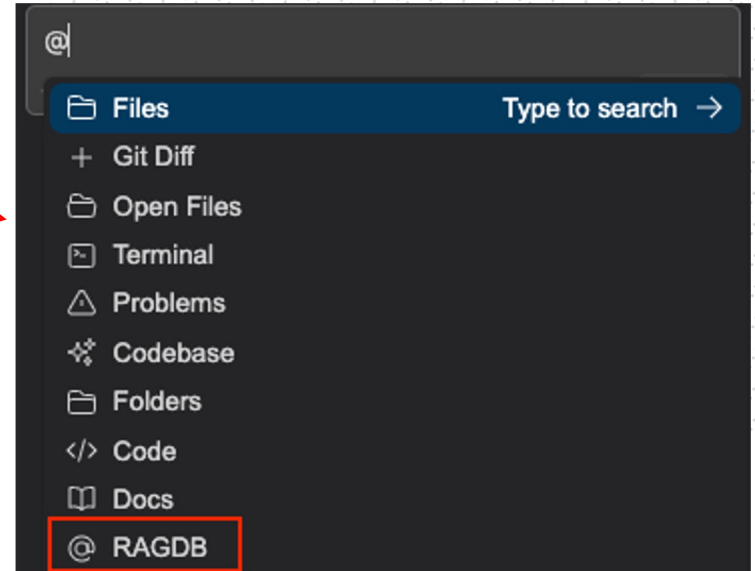
    const results = await response.json();

    return results.map((result: { title: any; contents: any; }) => ({
      name: result.title,
      description: result.title,
      content: result.contents,
    }));
  },
};
```

generate context

call retriever

user question



dbtSQL task = custom(context + prompt + model)

```
export function modifyConfig(config: Config): Config {
  config.slashCommands?.push({
    name: "dbtSQL",
    description: "Write a SQL code",
    run: async function* (sdk) {

      const inputArray = sdk.input.split(' ');
      const db = inputArray[0];
      const userQuestion = inputArray.slice(2).join(' ');
      const response = await fetch("http://localhost:8000/retrieve", {
        method: "POST",
        headers: {
          'content-type': 'application/json;charset=UTF-8',
        },
        body: JSON.stringify({ query: userQuestion }),
      });

      const results = await response.json();

      const ragResponse = results.map((result: { title: any; contents: any; }) => ({
        name: result.title,
        description: result.title,
        content: result.contents,
      }))[0];

      const PROMPT = `
      ## Task
      Generate a SQL query to answer: ${userQuestion}

      ## Instructions
      - If you cannot answer the question with the available database schema, return 'I
      - Format answer as a code using markdown in the chat.

      ## Database Schema
      The query will run on a database with the following schema:
      ${ragResponse.content}
    `;

      const model = "pxlksr/defog_sqlcoder-7b-2:Q4_K";

      const sql = await generateSQL(model, db, userQuestion, PROMPT);

      return sql;
    }
  });
}
```

/dbtSQL jaffle How many products by type and supplier that can decay and sort by product count desc?

+ Add Context Enter

task = context + prompt + model

- GPT-4 (Free Trial)
- GPT-4 Vision (Free Trial)
- Gemini Pro (Free Trial)
- Codellama 70b (Free Trial)
- GPT-3.5-Turbo
- gpt-4-turbo-preview
- Ollama-llama2
- sqlcoder-7b
- pxlksr/defog_sqlcoder-7b-2:Q4_K
- pxlksr/defog_sqlcoder-7b-2:F16
- sqlcoder-15b
- starcoder2-15b
- deepseek-coder-6.7b

- fast and easy self-hosting of LLMs almost everywhere
- hybrid CPU+GPU inference
- powered by [llama.cpp](https://github.com/jmorganca/llama.cpp)
- rich [library](#) of existing LLMs in different flavours
- [GGUF](#) - fast and memory efficient quantization for GPU
- Serve model with one-liner:

```
ollama run starcoder2:7b
```
- [vLLM](#) for production deployments

([Our video tutorial](#))

starcoder2

StarCoder2 is the next generation of transparently trained open code LLMs that comes in three sizes: 3B, 7B and 15B parameters.

↓ 21.2K Pulls ⌚ Updated 4 weeks ago

The screenshot shows the Ollama library interface for the 'starcoder2' model. At the top, there is a dropdown menu set to 'latest' and a search bar containing 'ollama run starcoder2'. Below this is a table of model versions with columns for the version name and its size. The 'latest' version is highlighted. To the right of the table, there is a list of tags, including '49 Tags', '167ae0f64584 · 1.7GB', 'starcoder2 · parameters 3B · quantization 4-bit · 1.7GB', and 'Open RAIL-M v1 License Agreement Section I: Preamb... · 13KB'. A 'View all tags' link is located at the bottom of the table.

Version	Size
latest	1.7GB
3b	1.7GB
7b	4.0GB
15b	9.1GB
3b-q4_0	1.7GB
3b-q4_1	1.9GB

Ollama - custom model in 4 steps

Hugging Face Search models, datasets, users...

defog / **sqlcoder-7b-2** like 194

Text Generation Transformers Safetensors GGUF llama

Model card Files and versions Community 30

main sqlcoder-7b-2

rishdotblog Update README.md 7e5b6f7 VERIFIED
.gitattributes 1.58 kB
README.md 3.54 kB
config.json 691 Bytes
generation_config.json 111 Bytes
label_mask.npy 458 kB LFS
labeled_matches.npy 458 kB LFS
labels.npy 3.66 MB LFS
model-00001-of-00003.safetens... 4.94 GB LFS
model-00002-of-00003.safetens... 4.95 GB LFS
model-00003-of-00003.safet... 3.59 GB LFS
model.safetensors.index.json 24 kB
predictions.npy 3.6 MB LFS
special_tokens_map.json 515 Bytes
sqlcoder-7b-q5_k_m.gguf 4.78 GB LFS

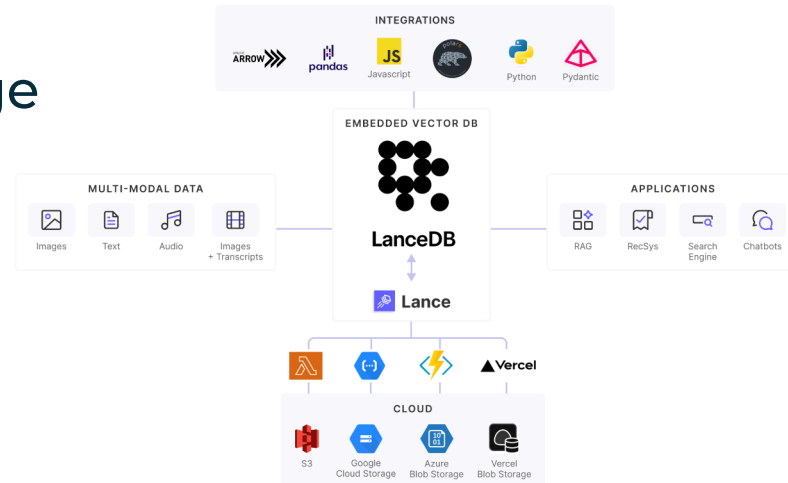
1. Download a model in the GGUF format
2. Create a Modelfile, e.g.:

```
FROM ./sqlcoder-7b-q5_k_m.gguf
TEMPLATE """{{ .Prompt }}"""
PARAMETER stop "<|endoftext|>"
```

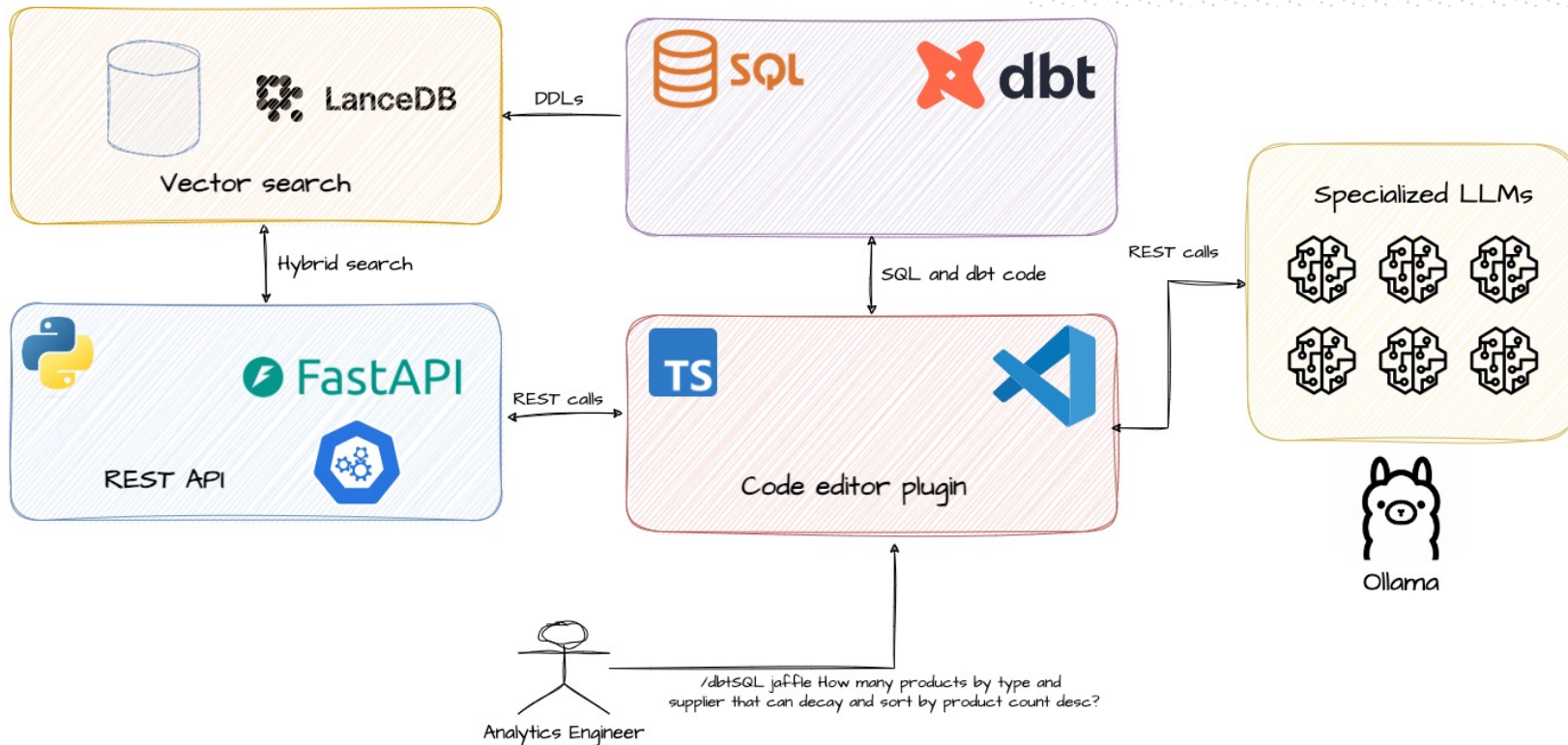
3. Create a model with Ollama
ollama create sqlcoder-7b-2 -f Modelfile

4. Serve it
ollama run sqlcoder-7b-2

- fast (Rust❤️), serverless and embeddable - DuckDB for ML
- powered by [Lance](#) file format for ML (versioning, zero-copy)
- multi-modal
- support for hybrid (semantic + keyword) search
- [Llamaindex](#) integration
- Python API and fast data exchange with [polars](#) and [Arrow](#)



Technical architecture



Question representation¹

```
Table continents, columns = [ContId, Continent]
Table countries, columns = [CountryId, CountryName,
↳ Continent]
Q: How many continents are there?
A: SELECT
```

Listing 1: Example of Basic Prompt

```
1 ### Complete sqlite SQL query only and with no
↳ explanation
2 ### SQLite SQL tables, with their properties:
3 #
4 # continents(ContId, Continent)
5 # countries(CountryId, CountryName, Continent)
6 #
7 ### How many continents are there?
8 SELECT
```

Listing 3: Example of OpenAI Demonstration Prompt

```
1 Given the following database schema:
2 continents: ContId, Continent
3 countries: CountryId, CountryName, Continent
4
5 Answer the following: How many continents are there?
6 SELECT
```

Listing 2: Example of Text Representation Prompt

```
1 /* Given the following database schema: */
2 CREATE TABLE continents(
3     ContId int primary key,
4     Continent text,
5     foreign key(ContId) references countries(Continent)
6 );
7
8 CREATE TABLE countries(
9     CountryId int primary key,
10    CountryName text,
11    Continent int,
12    foreign key(Continent) references continents(ContId)
13 );
14
15 /* Answer the following: How many continents are there?
↳ */
16 SELECT
```

Listing 4: Example of Code Representation Prompt

- Not meant to be yet another benchmark, such as: [Spider](#) or [Bird-SQL](#)
- [Jaffle Shop](#) example - simple but not trivial
- 4 typical data tasks
 - Data model exploration/discovery
 - SQL: an easy one (single table) and more complex (joins with sorting and aggregations)
 - dbt model generation
 - dbt tests generation based on rules

LLMs evaluation 2/2

Model	Licence	size [b]	Data discovery	SQL - simple	SQL - complex	dbt - model	dbt - tests
deepseek-coder	deepseek	33	+	+	+/-	+/-	-
deepseek-coder	deepseek	6.7	+/-	+/-	+/-	-/+	-
codellama	Llama2	70	+	+	-/+	-	-
starcode2	bigcode-openrail-m	15	-	-	-	-	-
sqlcoder	CC BY-SA 4.0	7	-	+/-	+/-	N/A	N/A
phind-codellama	Llama2	34	+	+	+	+	+/-
wizardcoder	Llama2	33	+	+	+/-	+/-	-/+
gpt-3.5-turbo	Commercial	N/A	+	+	+	+	+/-
gpt-4	Commercial	N/A	+	+	+	+	+
gpt-4-turbo-preview	Commercial	N/A	+	+	+	+	+/-
Gemini Pro	Commercial	N/A	+	+/-	+	+/-	-
OpenCodeInterpreter	Apache-2.0	33	+	+	-/+	-	-

+ - perfect or almost perfect

+/- - not optimal or some minor tweaks needed

-/+ - not very helpful, serious hallucinations

- - totally useless

A handful of conclusions...with a grain of salt

- NL-to-SQL and dbt code generation are **challenging**
- **commercial** models are in most cases still better... but
- there are very promising **open-source ~30b alternatives**
- **SQL-dedicated** and fine-tuned models can turn out a bit a disappointing (beam search?), e.g. :
 - unnecessary joins elimination
 - wrong data types inference
 - occasional hallucinations

Future directions

- Implementation of in-context learning such as Query Similarity Selection (few-shot strategy)
- Model(s) fine-tuning using using dbt examples, especially for data quality aspects
- Fine-tuning focused on platform-to-platform migrations

A network diagram consisting of grey dots connected by thin grey lines, forming a complex web-like structure on the left side of the slide.

**Welcome to the
GID data copilot
DEMO**

Two thick, bright green diagonal bars located in the bottom right corner of the slide.

AI & LLMOps free consultation



Thank you !