Qdrant MCP-code-snippets

Up-to-Date Code for AI Copilots Through Semantic Context Lookup

Evgeniya Sukhodolskaya
& Till Bungert,

@Qdrant

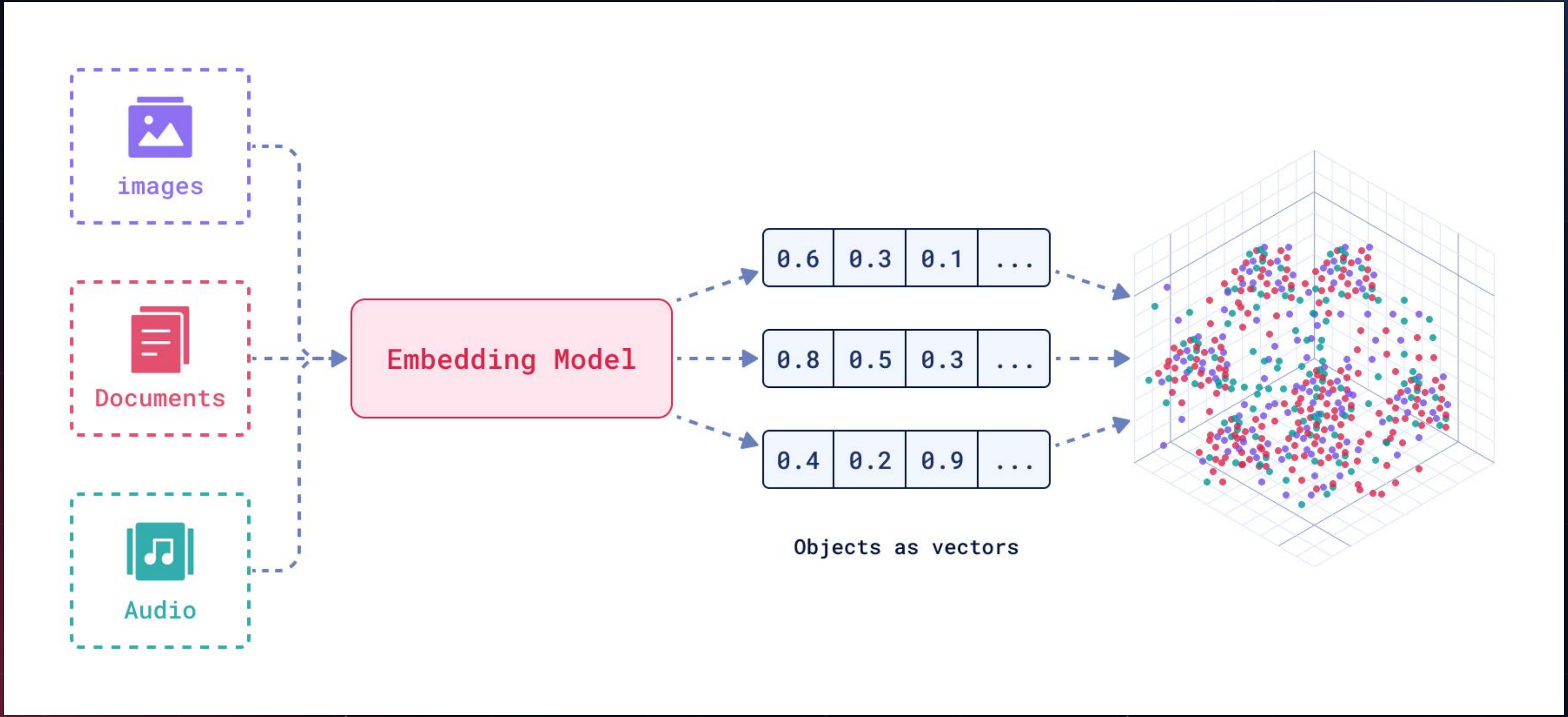
Qdrant is an open-source Vector Search Engine

or you might have also heard the term “*vector database*”,

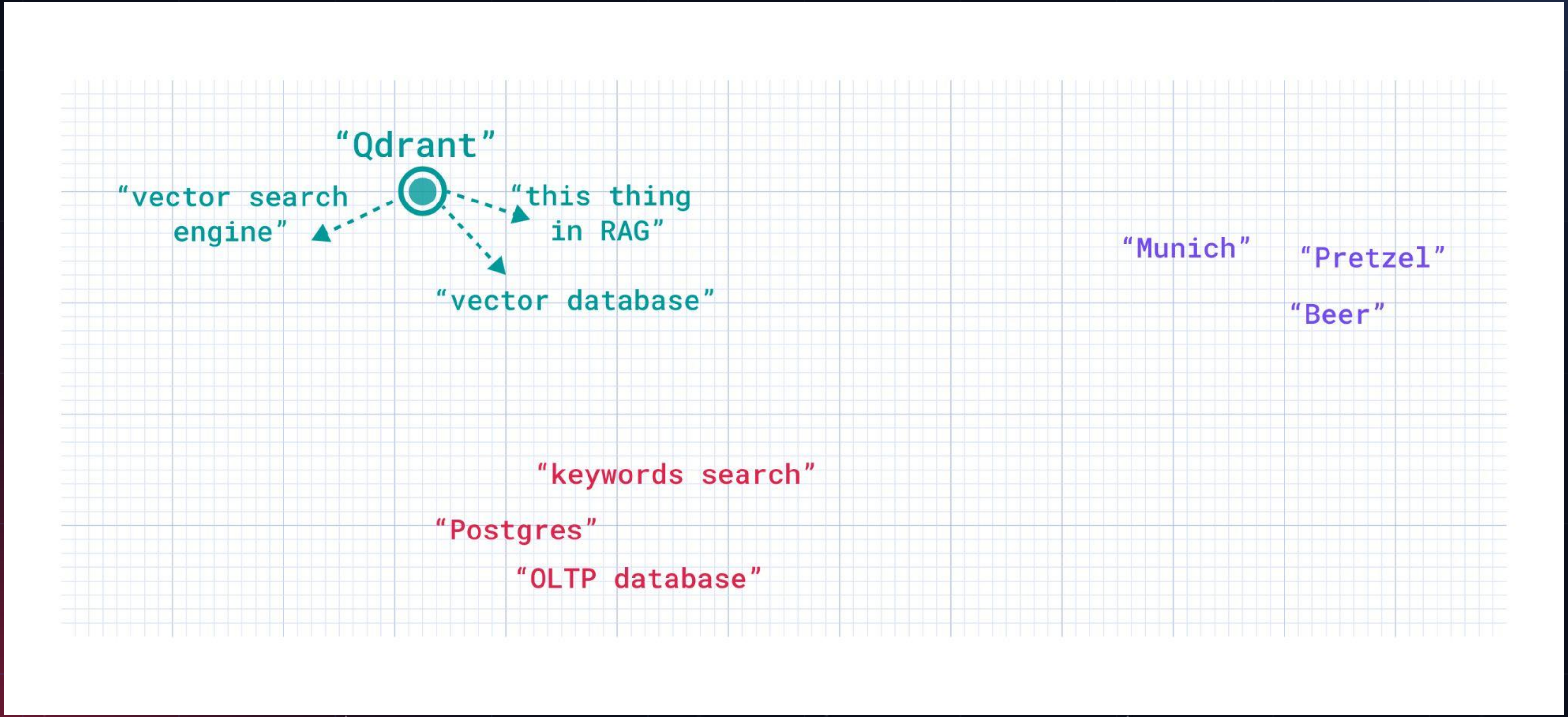
“*semantic similarity search engine*”,

“the thing under the hood of (Agentic) RAG”, …

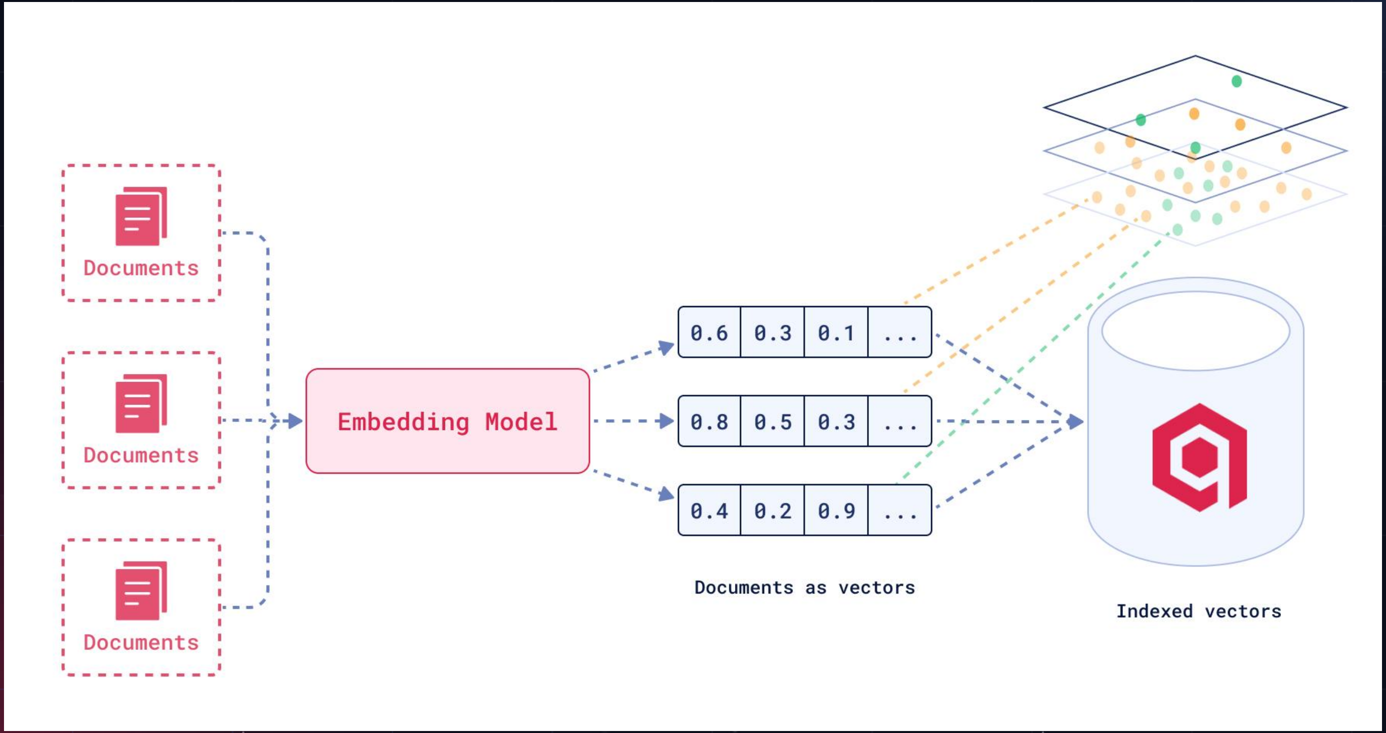
Qdrant is a Vector Search Engine



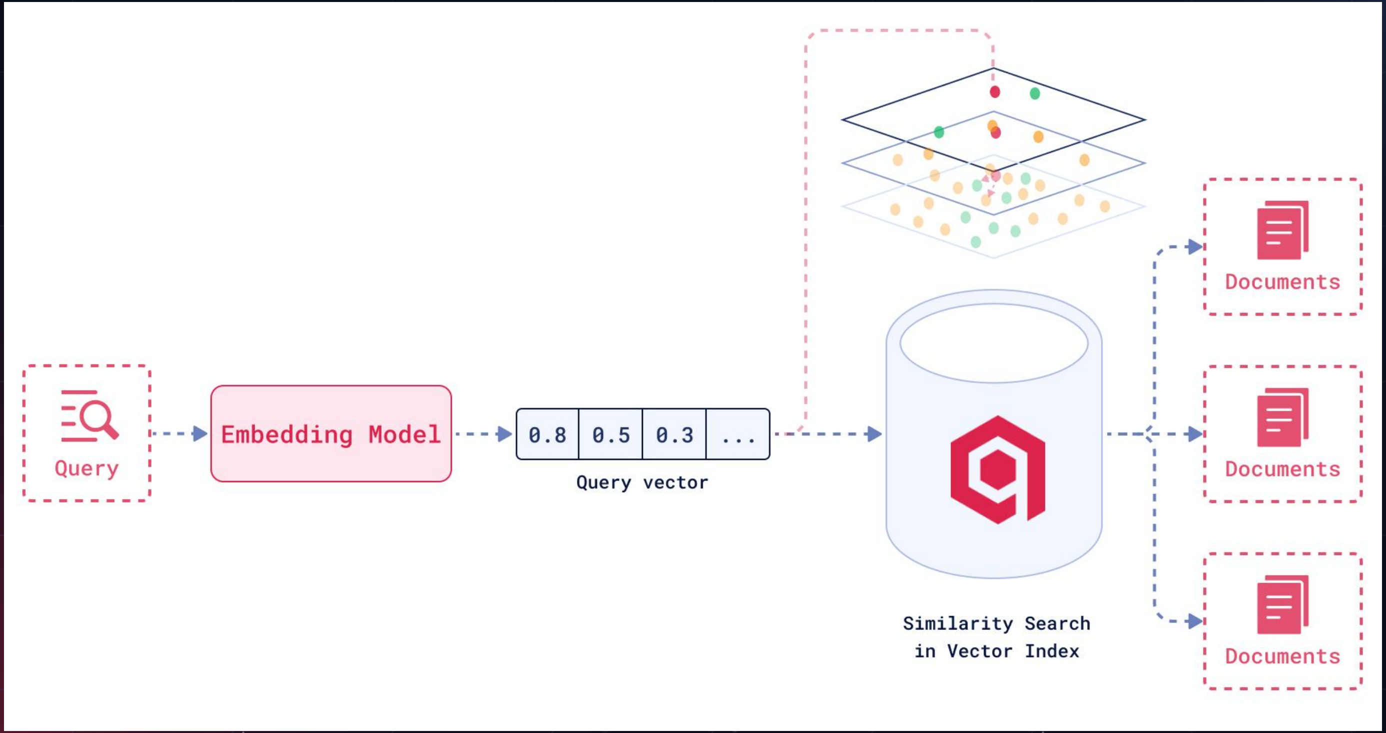
Qdrant is a Vector Search Engine



Qdrant is a Vector Search Engine: Store



Qdrant is a Vector Search Engine: Find



So Qdrant can be Used for Semantic Context Lookup

And AI Coding Assistants Need Context

👤 **User:**

- *Implement hybrid (dense + lexical/mixed/keywords & vectors) search in Qdrant*

🤖 **Coding Assistant:**

- Uses deprecated method “search” instead of “query_points”;
- Doesn’t know Qdrant, since 1.10.0, supports multistage searches with “prefetch” & fusion;
- Doesn’t know Qdrant can calculate Inverse Document Frequency (IDF) on the server side;
- Doesn’t know about local & cloud inference…

=> People come & say: “*Qdrant doesn’t have hybrid search functionality*” 🤦

And AI Coding Assistants Need Context

Why?
LLMs are retrained, but it's mostly **impossible to keep them** constantly **updated** & **in sync** with library versions.

Just our Qdrant code documentation covers:

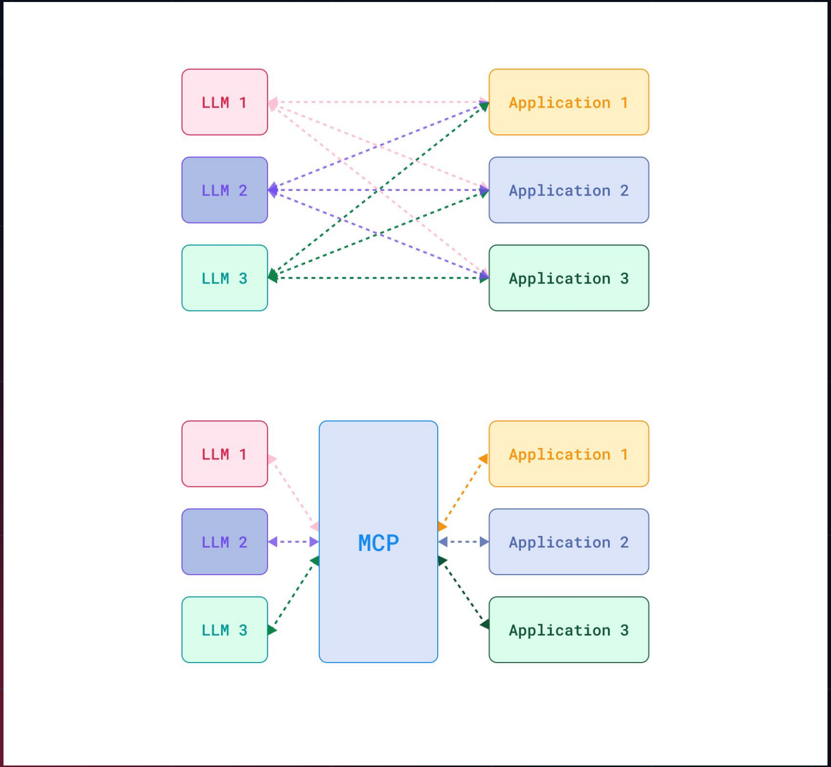
- **Several clients** (Java, Python, GO, C#, Rust, Typescript);
- **Several versions** (1.16.2 now), each with new features.

LLM-friendly API reference?
Yes, we have **/llm-full.txt** at our API Reference page, but it's big (**~53K tokens**) => Lost-in-the-middle effect.

How Can AI Coding Assistants Leverage Qdrant's Semantic Search?

Through an MCP Server

Model Context Protocol (MCP)



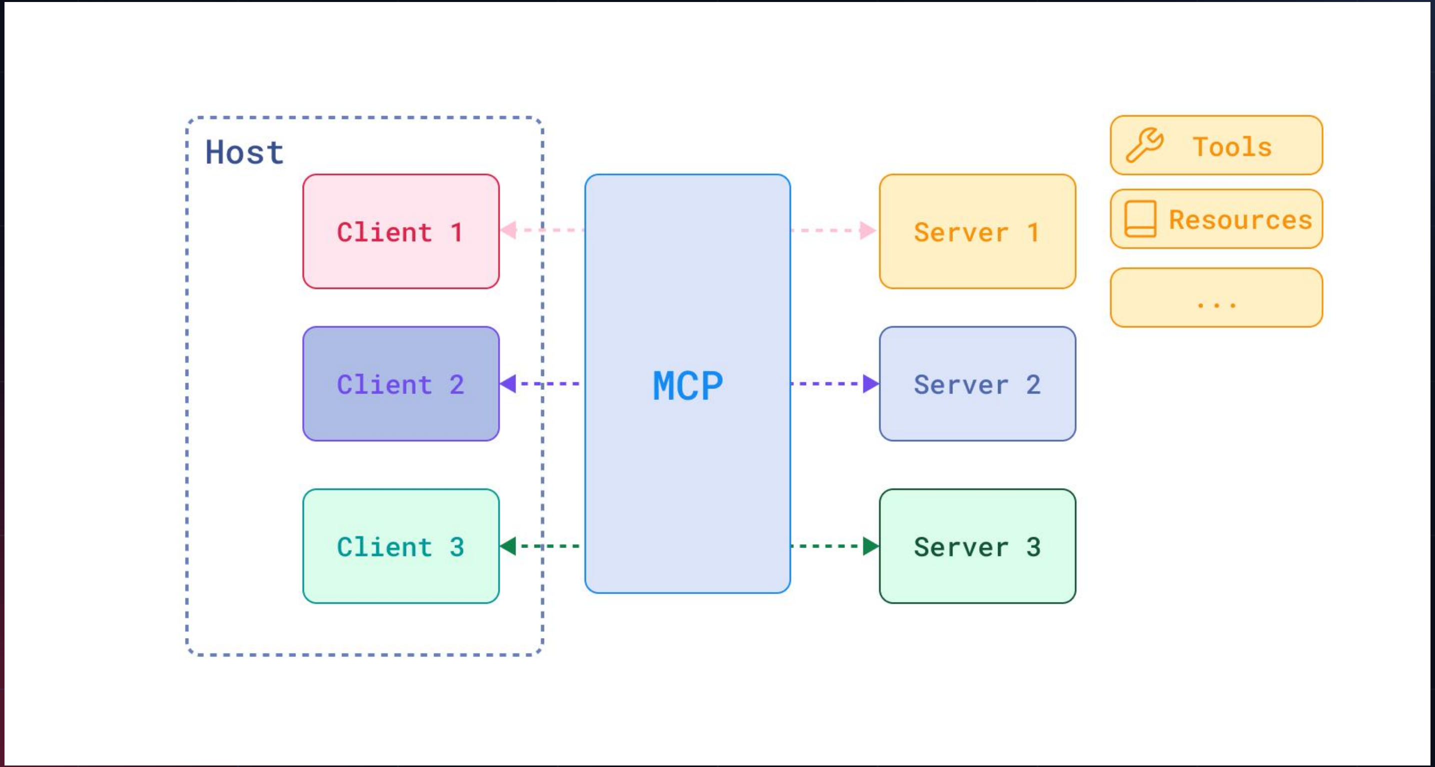
"MCP is an open protocol that standardizes how applications provide context to LLMs.

Think of MCP like a USB-C port for AI applications."

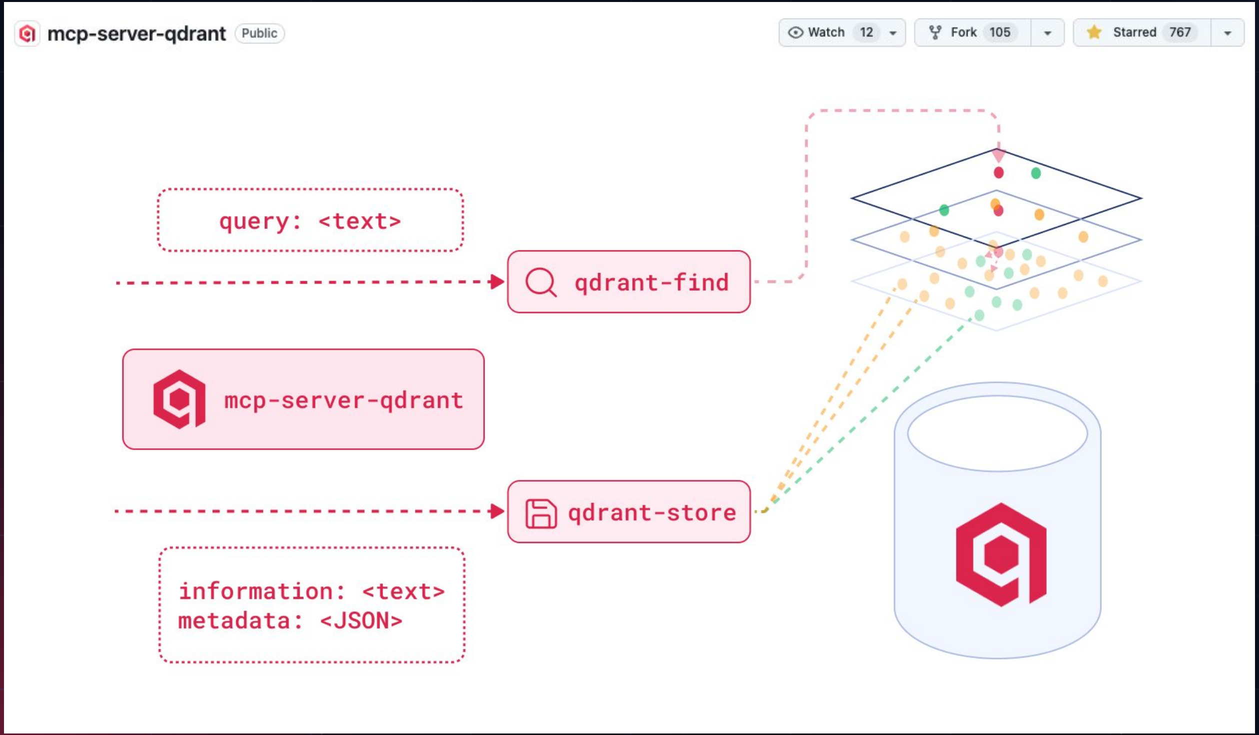
modelcontextprotocol.io

HuggingFace MCP course

Model Context Protocol (MCP)



Qdrant MCP Server



mcp-server-qdrant

Possible Applications

- Inline Retrieval-Augmented Generation (RAG) <originally used as a note-taker>;
- Automating your codebase documentation;
- Personalizing your code assistant with best practices, templates, and specifics of your codebase;
- **Providing up-to-date API reference & code documentation for AI coding assistants.**

What do We Want

We need, for all package functions, to **store**:

- Well-curated snippet (metadata, context for the agent);
- Accurate description of this snippet's functionality (vector);
- Programming language tag (metadata);
- Version tag (metadata);
- Package name tag (metadata).

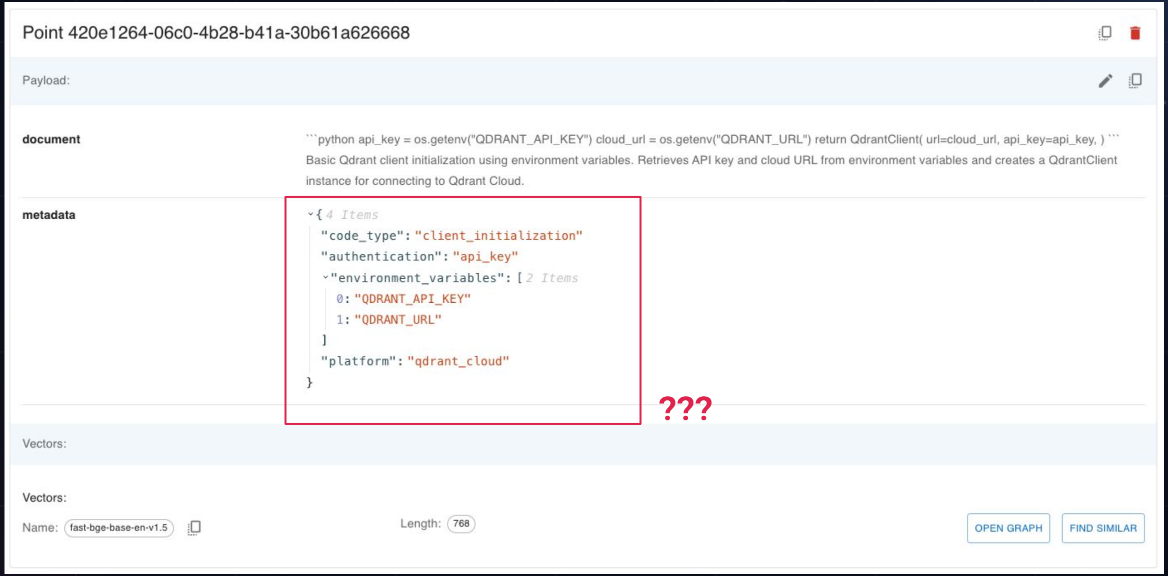
We need, for every user request, to **find:**

- **Semantic match** by description of a code snippet, **filtered** by fitting version, package name, and programming language

What did We Get

qdrant-store: sometimes stores weird metadata

qdrant-find: troubles with consistent & correctly-formed filters



Point 420e1264-06c0-4b28-b41a-30b61a626668

Payload:

document ```python api_key = os.getenv("QDRANT_API_KEY") cloud_url = os.getenv("QDRANT_URL") return QdrantClient(url=cloud_url, api_key=api_key,) ``` Basic Qdrant client initialization using environment variables. Retrieves API key and cloud URL from environment variables and creates a QdrantClient instance for connecting to Qdrant Cloud.

metadata

```
{ 4 Items
  "code_type": "client_initialization"
  "authentication": "api_key"
  "environment_variables": [ 2 Items
    0: "QDRANT_API_KEY"
    1: "QDRANT_URL"
  ]
  "platform": "qdrant_cloud"
}
```

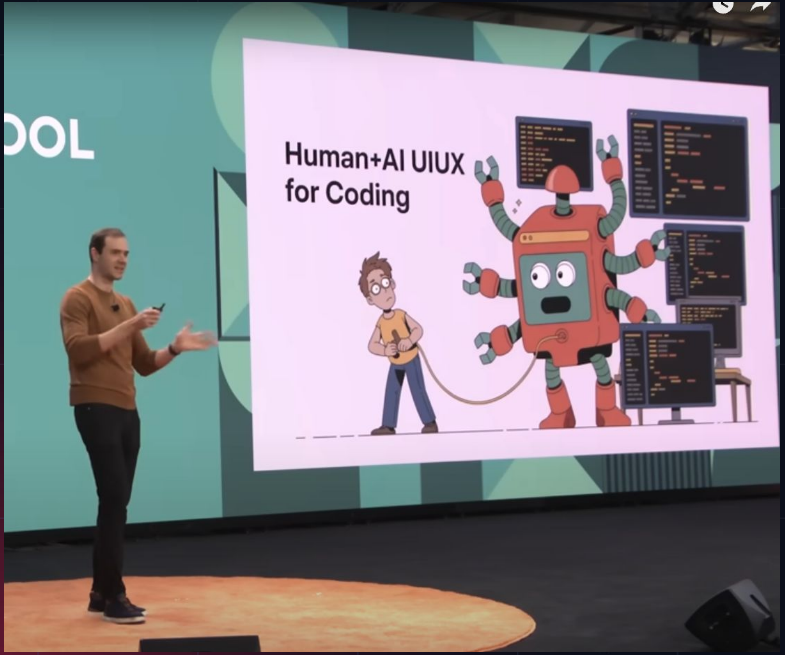
???

Vectors:

Vectors:

Name: fast-bge-base-en-v1.5 Length: 768

Keep Agents on a Leash



We need to help our AI coding assistants by avoiding situations with **too many degrees of freedom**.

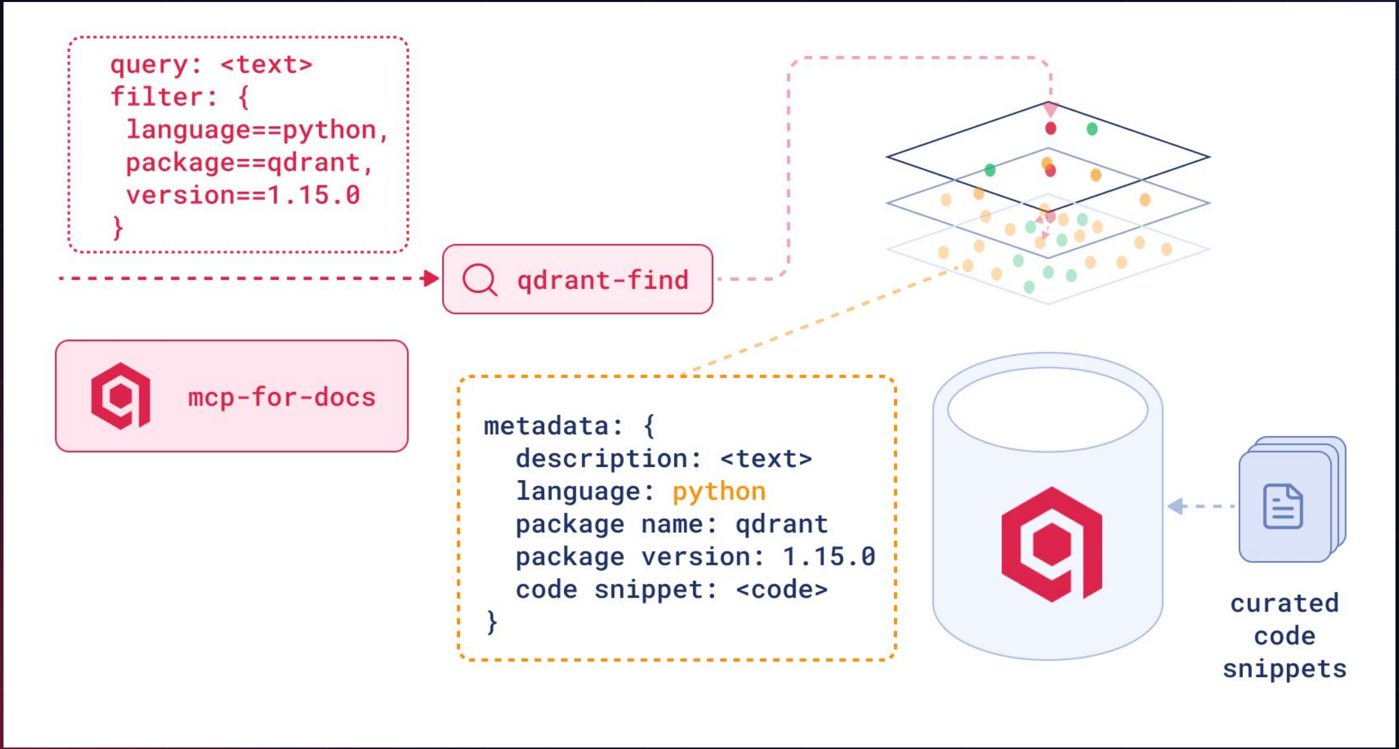
So, we updated our **mcp-server-qdrant** to use it as a **customizable base class**:

- **Customizable filters** restrictions & LLM-friendly format of metadata filtering
- Customizable switching `qdrant-store` tool on/off (**`read_only` mode**)

And built on top our
own custom MCP server, suitable solely as an API reference for AI coding assistants.

Qdrant MCP-code-snippets

#1 Remote MCP server: mcp-for-docs

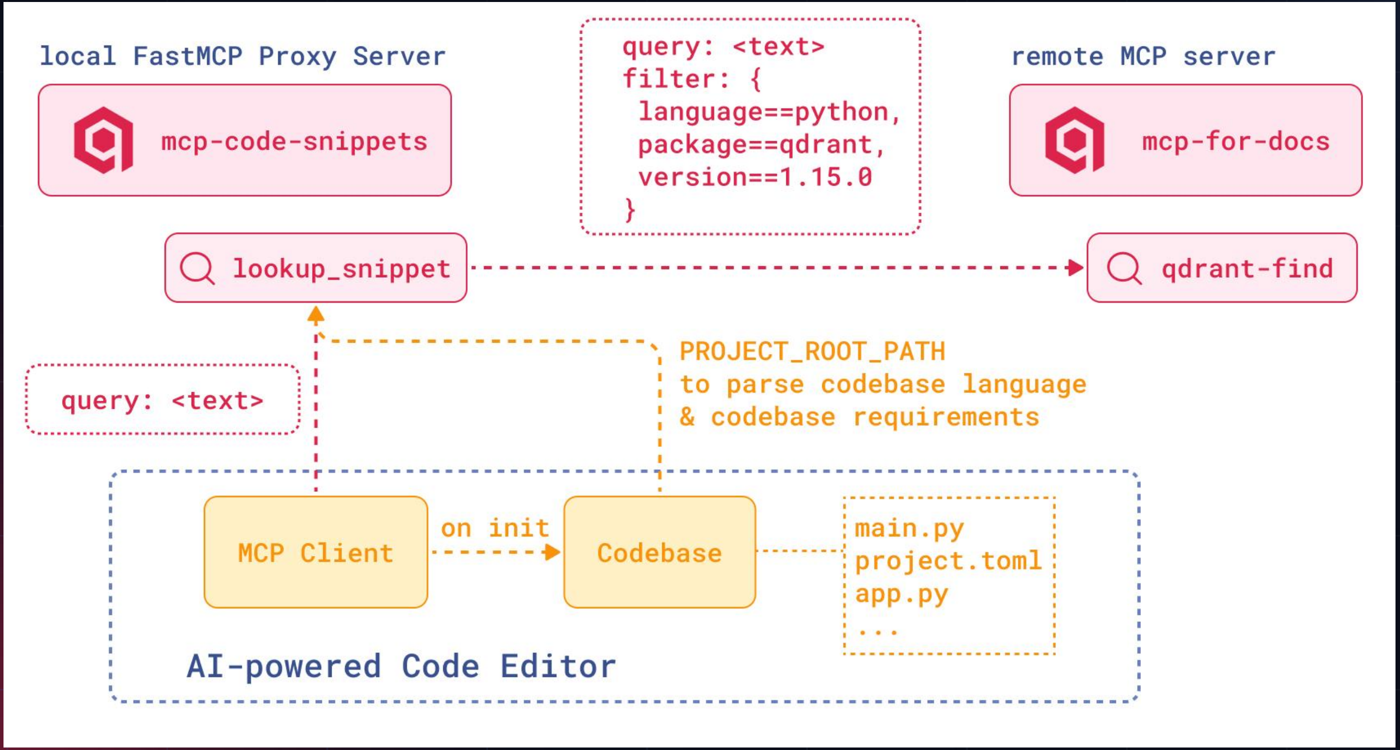


```
query: <text>
filter: {
 language==python,
 package==qdrant,
 version==1.15.0
}
```

```
metadata: {
  description: <text>
  language: python
  package name: qdrant
  package version: 1.15.0
  code snippet: <code>
}
```

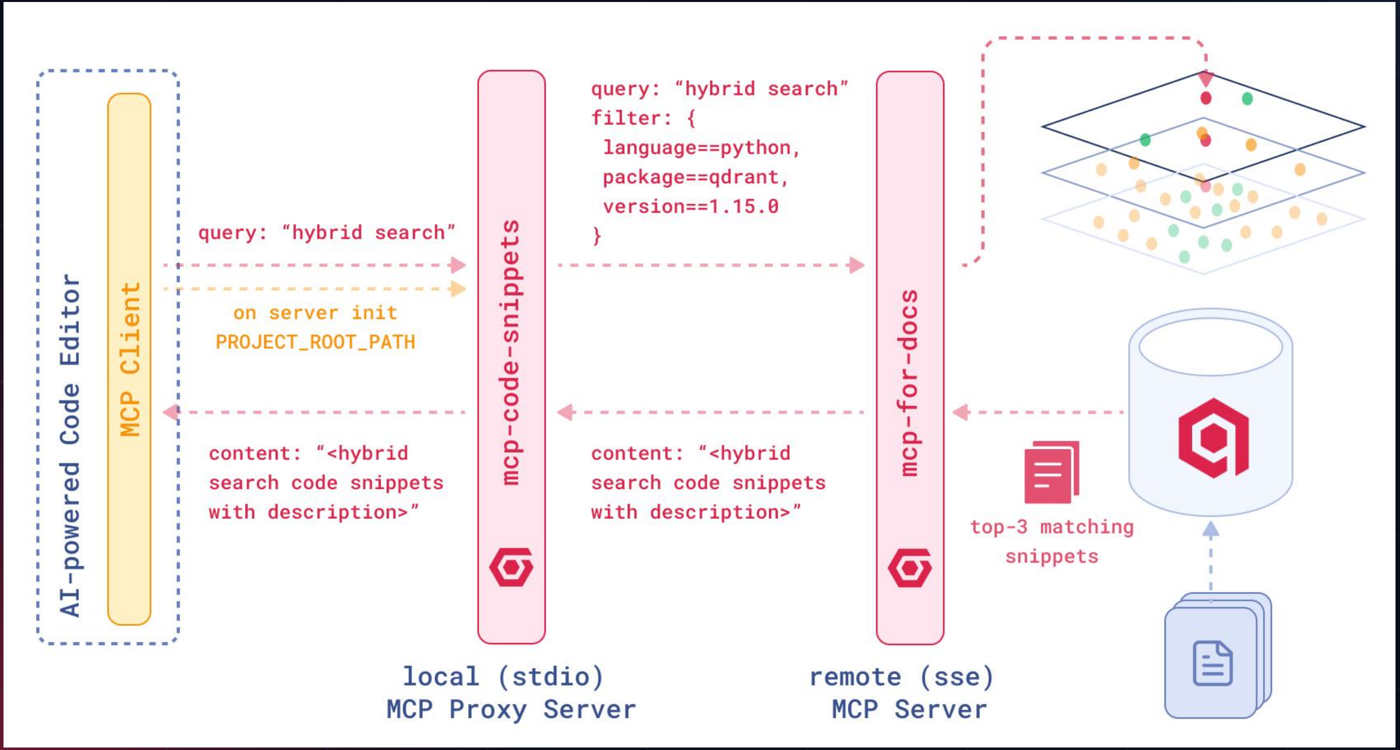
mcp-for-docs

#2 Local MCP Proxy: mcp-code-snippets



mcp-code-snippets

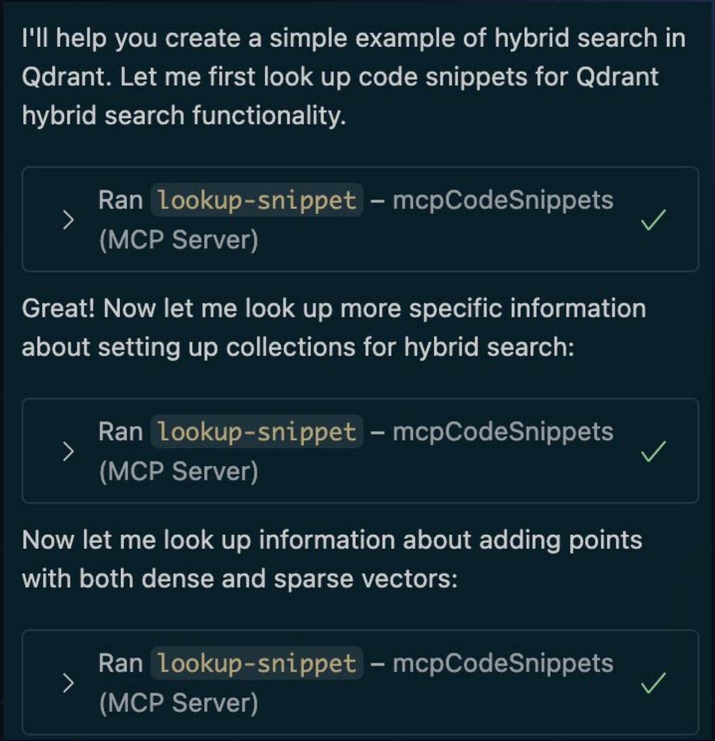
How it was Supposed to Work



And it Did!

At the end of July, `**mcp-code-snippets**` could be installed in any popular IDE and would supply AI coding assistants with the latest Qdrant code snippets.

“Hybrid Search with Qdrant” –>



I'll help you create a simple example of hybrid search in Qdrant. Let me first look up code snippets for Qdrant hybrid search functionality.

Ran `lookup-snippet` – mcpCodeSnippets (MCP Server) ✓

Great! Now let me look up more specific information about setting up collections for hybrid search:

Ran `lookup-snippet` – mcpCodeSnippets (MCP Server) ✓

Now let me look up information about adding points with both dense and sparse vectors:

Ran `lookup-snippet` – mcpCodeSnippets (MCP Server) ✓

So We Set a New Goal

Turn `**mcp-code-snippets**` into a **universal open-source API reference for AI coding assistants**.

Basically, extend the snippets dataset to cover every popular library out there.

Why?

- We believe in **open source** over gatekeeping.
- We know how to do search (if not us, then who?), so we have all the potential to build a **high-quality** tool for supplying AI coding assistants with context.
- It's a fun, simple project. How long could it take?

Well, How Hard Can it Be

Qdrant snippets are manually curated & on our Github. Let's just parse other GitHub repos with docs!

Problems:

- Some part is in docstrings, some in .md/.rst files, some buried in source code.
- Auto doc generation tools can't be reused, they're not universal.
- Some source files aren't accessible.

Solution:
Parse HTML directly and do it in a unified, scalable way → build an API.

What Can Even Go Wrong...

HTML parsing doesn't provide snippet languages in a universal way.
And we **need them** to filter.

Oh, but there are open-source language-detecting tools! Wait... but... the quality is bad?!

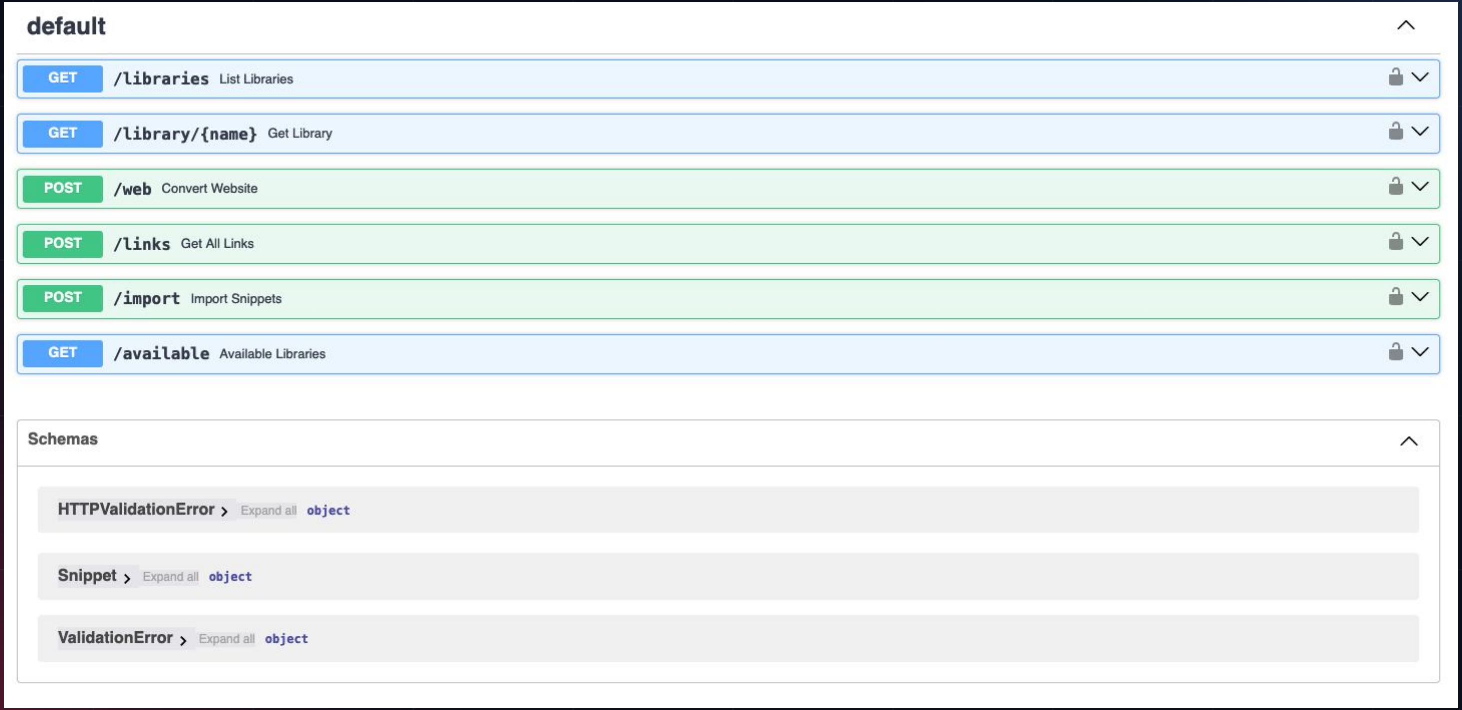
You're here: fine-tuning CodeBERT three times

- Generated a training dataset the same way as Guesslang (20k examples per language)
- Had to `teach codeberta-small-v1` C#

85% accuracy on Qdrant code snippets vs 65% for the base model

Our CodeBERT

Finally, docs2snippets*



default

GET /libraries List Libraries

GET /library/{name} Get Library

POST /web Convert Website

POST /links Get All Links

POST /import Import Snippets

GET /available Available Libraries

Schemas

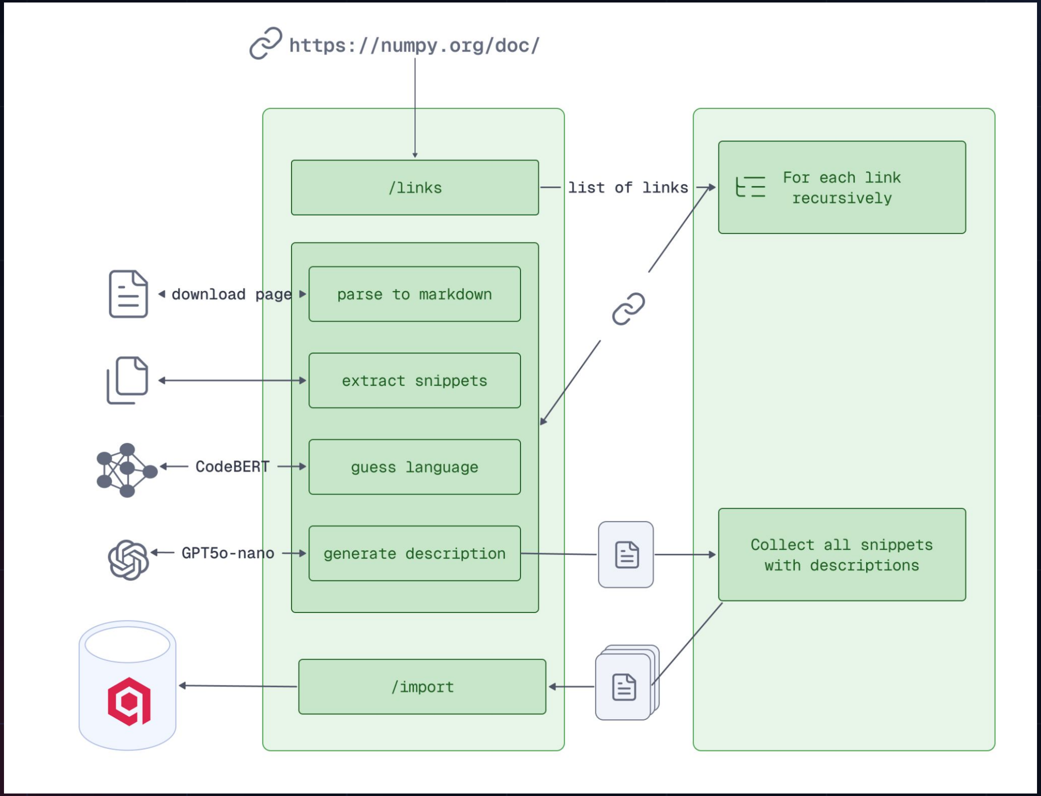
HTTPValidationError Expand all object

Snippet Expand all object

ValidationError Expand all object

*Will be open-sourced soon, monitor Qdrant org on GitHub

Dissecting docs2snippets



Where We Are Now

Numbers:

- 40 libraries (Python, most popular), plus Qdrant in all 6 languages
- ~50k code snippets

And it works! You'll see in a minute :)

Plans:

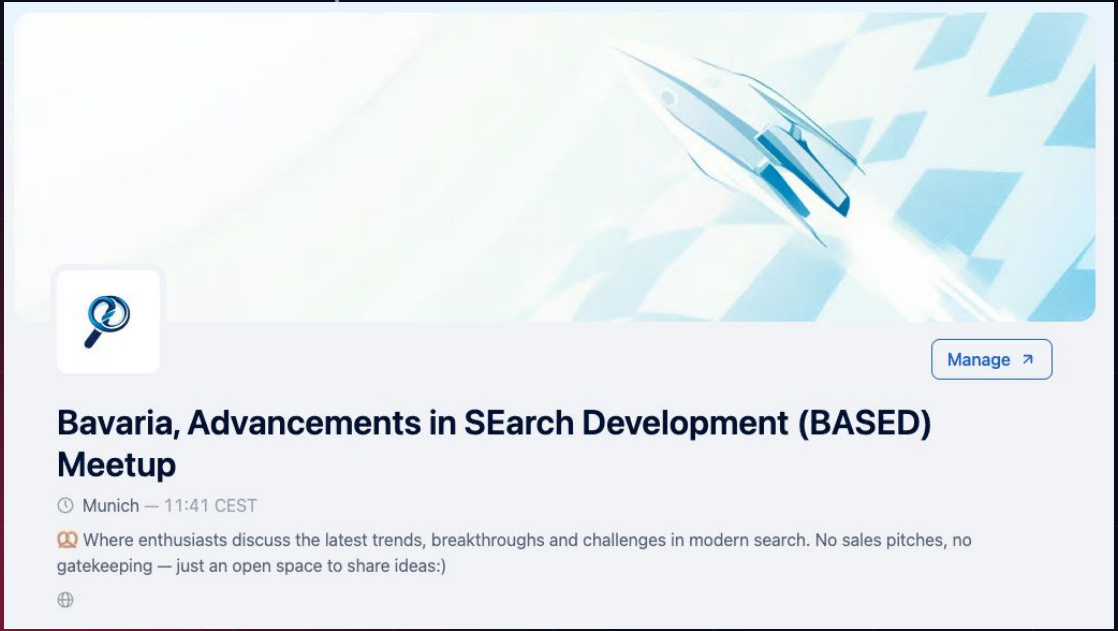
- Open-source docs2snippets
- Search optimization (prompt engineering, code-to-text, hybrid search, etc.)
- Eval & benchmarks
- Much-much-much more libraries across all 6 languages
- Package versioning support

Goal stays the same, and we're almost there!

Demo
(say hi to Till)

Thank You!
Let’s Keep the Tech Yapping Going:)

My Search Meetup, the next one is on the 26th of March, 2026



Bavaria, Advancements in SEarch Development (BASED) Meetup

Munich — 11:41 CEST

Where enthusiasts discuss the latest trends, breakthroughs and challenges in modern search. No sales pitches, no gatekeeping — just an open space to share ideas:)

Till Bungert

Evgeniya Sukhodolskaya