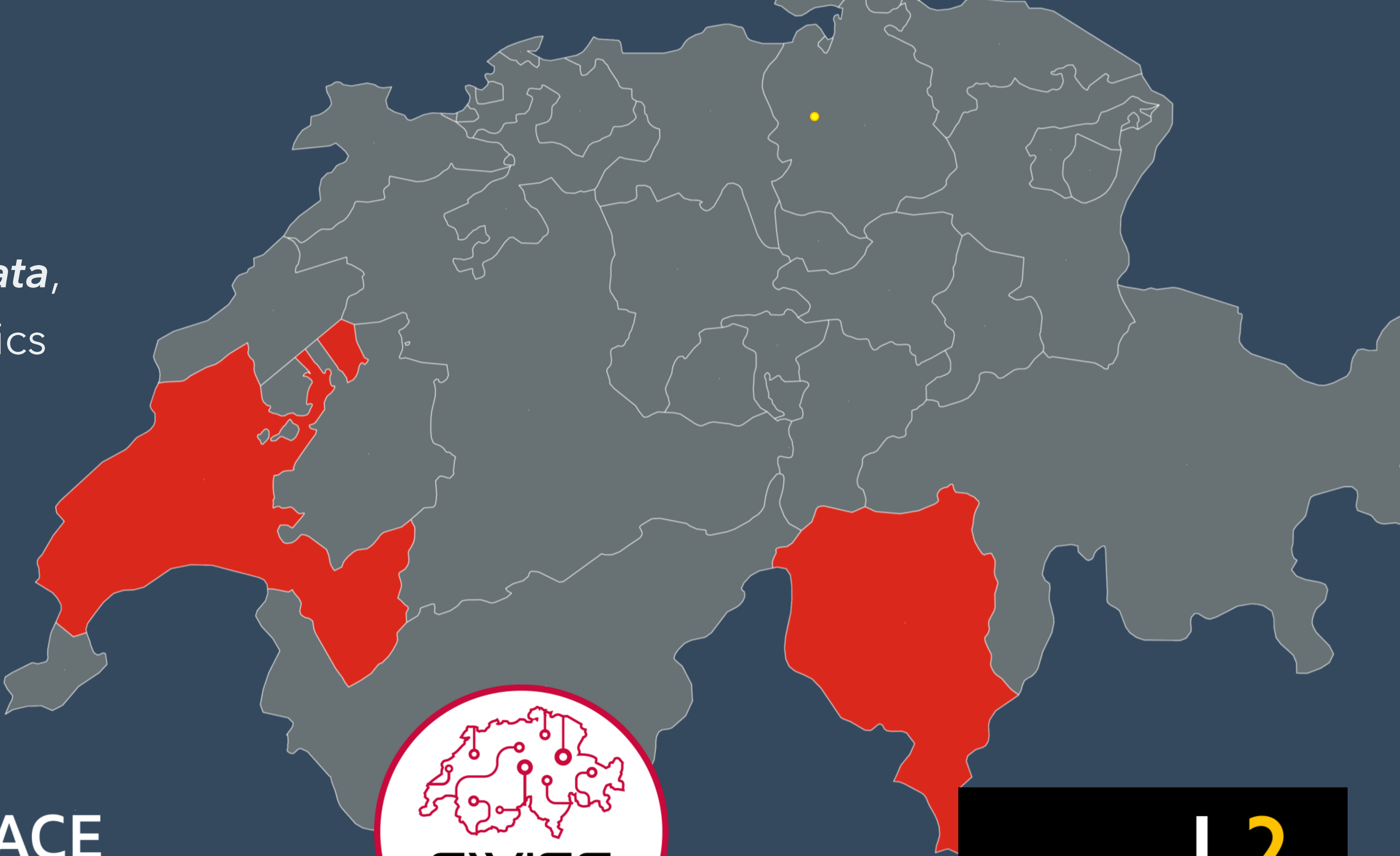
Can the modern Frankenstein pass the Turing test?

September 20, 2024

Gianni Ceresa

Working with ***data***,
Business Analytics
and EPM tools
for more than
15 years



Oracle ACE
Director

SWISS OUC

_SYM42

@G_Ceresa

DATAlysis

Oracle ACE

430+ technical experts helping peers globally

The **Oracle ACE Program** recognizes and rewards community members for their technical and community contributions to the Oracle community

3 membership tiers

Oracle ACE Director

Oracle ACE Pro

Oracle ACE Associate

For more details on Oracle ACE Program:
ace.oracle.com

Oracle ACE

Nominate

yourself or someone you know:

ace.oracle.com/nominate

Connect: **aceprogram_ww@oracle.com** Facebook.com/OracleACEs @oracleace Oracle ACE Program Group

DISCLAIMER

BETA

This topic could easily take a few hours, making it fit in 45 minutes requires choices on what to ignore.

Every week I try a different approach, the slides and the code were still changing in the train coming here 2h ago.

This is really what you can call “work in progress”, still not sure of the direction to give to this presentation…

Can the modern Frankenstein pass the Turing test?

Building your own custom Oracle Database ChatGPT-like solution

ChatGPT

ChatGPT is great for various tasks, you can ask it anything and it will (almost) never complain and will give you an answer.

How good is the answer?

Depends: how good was the question?

Just like you and me can make up a story, so can ChatGPT with hallucinations:

- False or misleading information presented as fact
- Analysts estimated that chatbots hallucinate as much as 27% of the time, with factual errors present in 46% of generated texts* *(this was in 2023, things change quickly)*

* https://en.wikipedia.org/wiki/Hallucination_(artificial_intelligence)

A chatbot powered by LLM based on facts you provide

Where to start to build an Oracle Database ChatGPT-like solution?

RAG: retrieval-augmented generation

A technique for enhancing the accuracy and reliability of generative AI models with facts fetched from external sources

RAG: one acronym, an almost infinite number of ways to achieve it…

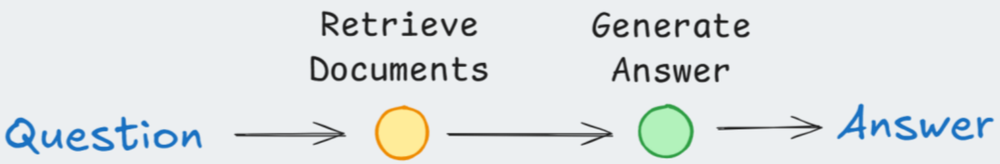
- From the simplest retrieval fed to a LLM
- To more advanced processes with decisions made before and evaluation of the quality of the facts provided and the quality of the answer generated

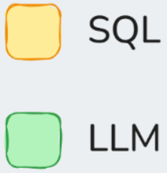
Let's see some examples…

RAG, one acronym, many ways: the very simple one

The bare minimum for RAG:

- Retrieve Documents
  - Query the database, in Oracle Database 23ai using VECTOR_DISTANCE, to get the closest pieces of content by vector distance to the question asked.
- Generate Answer
  - Ask a LLM to generate an answer to the question using the provided documents.

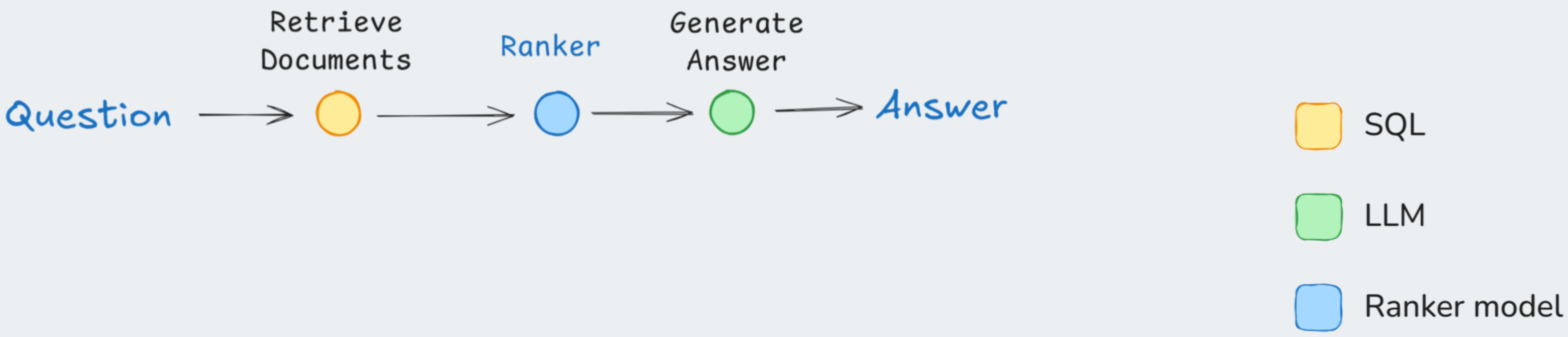




RAG, one acronym, many ways: the improved simple one

Adding an extra steps to the bare minimum to try to improve the result:

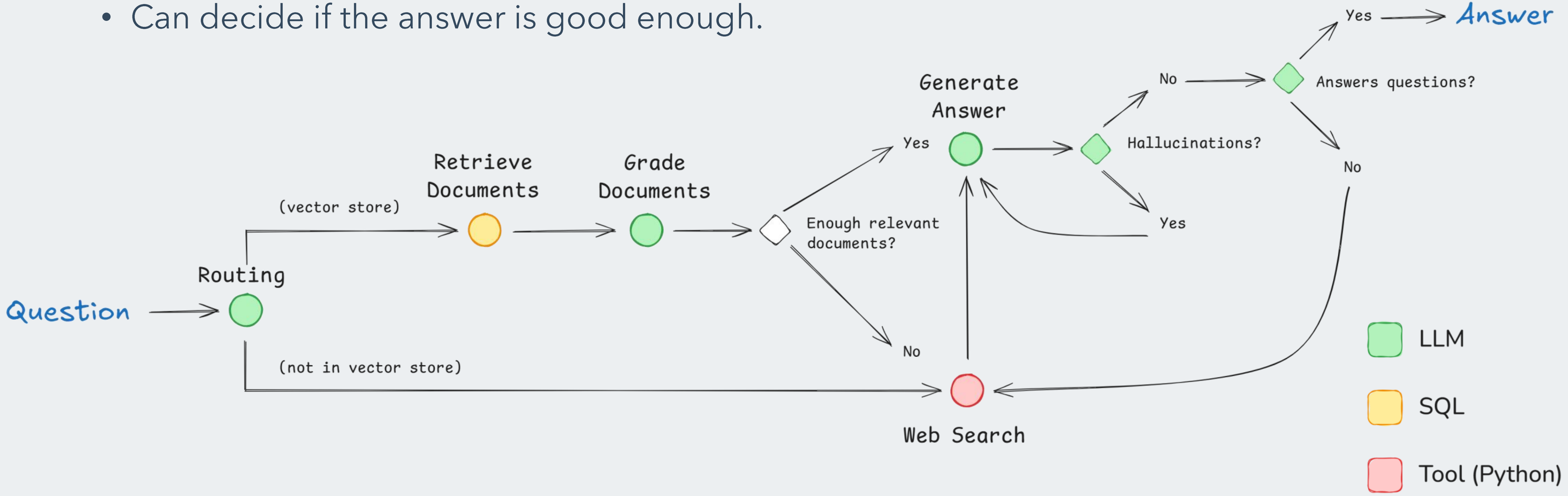
- Ranker
  - Retrieving documents is good, filtering the documents by the most relevant documents is better.
  - Retrieve more documents, rank them by relevance and take a shorter subset sorted by the relevance ranking (from most relevant to less relevant). For example retrieve 10 documents and take a TOP5 from the Ranker to pass to the LLM. Allows to remove “false positive” results of the retrieval step.



RAG, one acronym, many ways: the less simple but hopefully better

Your imagination is the limit:

- A LLM can do many things when instructed correctly
  - Can decide if the vector store is able to answer the question.
  - Can decide if documents are relevant for the question.
  - Can decide if the answer sounds plausible or not.
  - Can decide if the answer is good enough.



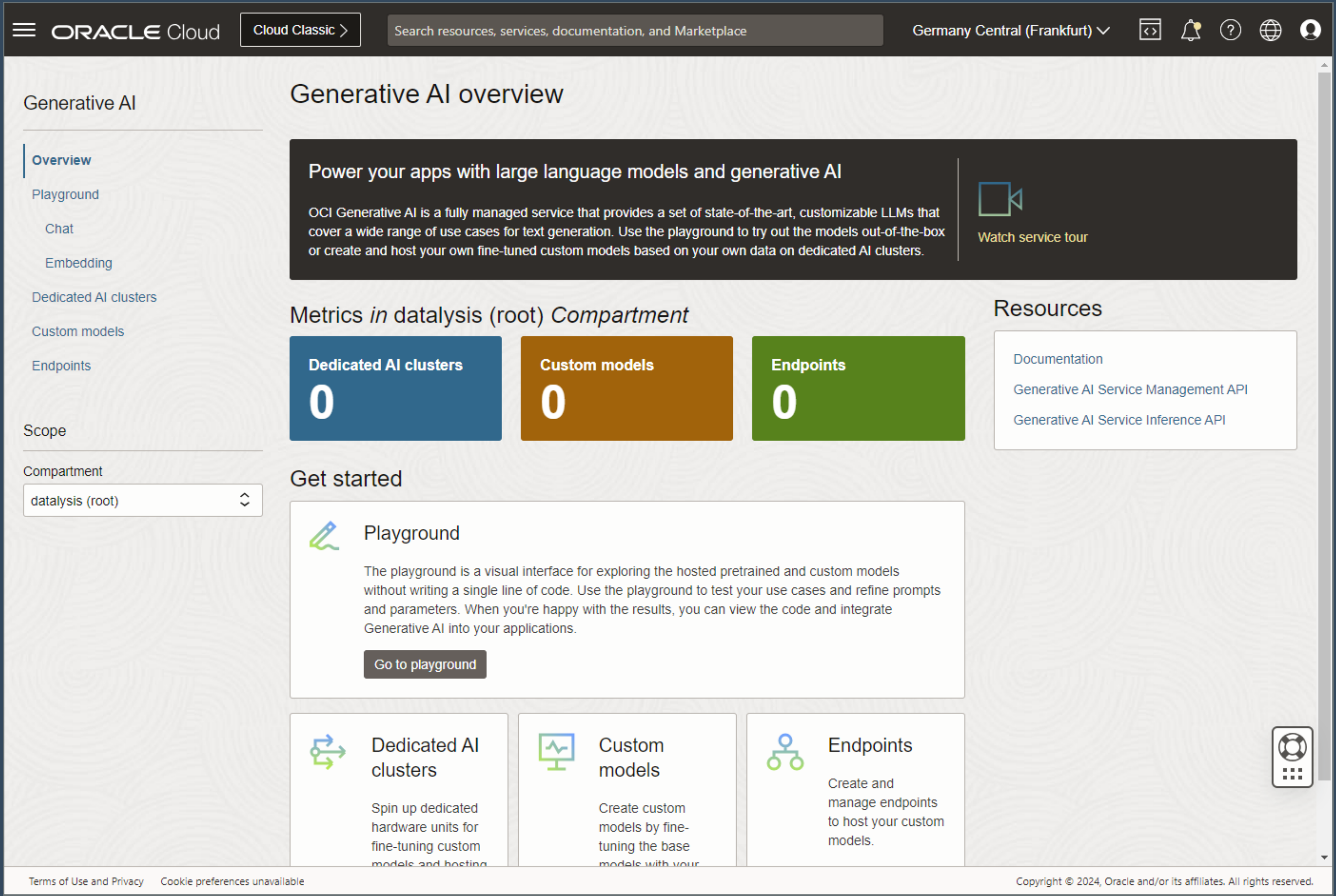
Recipe for a ChatGPT-like chatbot for Oracle Database

Ingredients:

- Data, documents, some source content
- Oracle Database 23ai
- An embedding model
  - In the database
  - Or a service somewhere called by the database
    - OCI GenAI, Cohere, GoogleAI, HuggingFace, OpenAI or VertexAI
  - Or some solution outside the database (Python, etc.)
- A LLM service somewhere
  - A service somewhere called by the database
    - OCI GenAI, Cohere, GoogleAI, HuggingFace, OpenAI or VertexAI
  - Or some solution outside the database (Python, Ollama etc.)
- Some code glueing together the pieces

A quick look at Oracle OCI GenAI

OCI GenAI


ORACLE Cloud
Cloud Classic >
Search resources, services, documentation, and Marketplace
Germany Central (Frankfurt)
Generative AI
Overview
Playground
Chat
Embedding
Dedicated AI clusters
Custom models
Endpoints
Scope
Compartment
datalysis (root)
Generative AI overview
Power your apps with large language models and generative AI
OCI Generative AI is a fully managed service that provides a set of state-of-the-art, customizable LLMs that cover a wide range of use cases for text generation. Use the playground to try out the models out-of-the-box or create and host your own fine-tuned custom models based on your own data on dedicated AI clusters.
Watch service tour
Metrics in datalysis (root) Compartment
Dedicated AI clusters
0
Custom models
0
Endpoints
0
Resources
Documentation
Generative AI Service Management API
Generative AI Service Inference API
Get started
Playground
The playground is a visual interface for exploring the hosted pretrained and custom models without writing a single line of code. Use the playground to test your use cases and refine prompts and parameters. When you're happy with the results, you can view the code and integrate Generative AI into your applications.
Go to playground
Dedicated AI clusters
Spin up dedicated hardware units for fine-tuning custom
Custom models
Create custom models by fine-tuning the base
Endpoints
Create and manage endpoints to host your custom models.
Terms of Use and Privacy
Cookie preferences unavailable
Copyright © 2024, Oracle and/or its affiliates. All rights reserved.

OCI GenAI

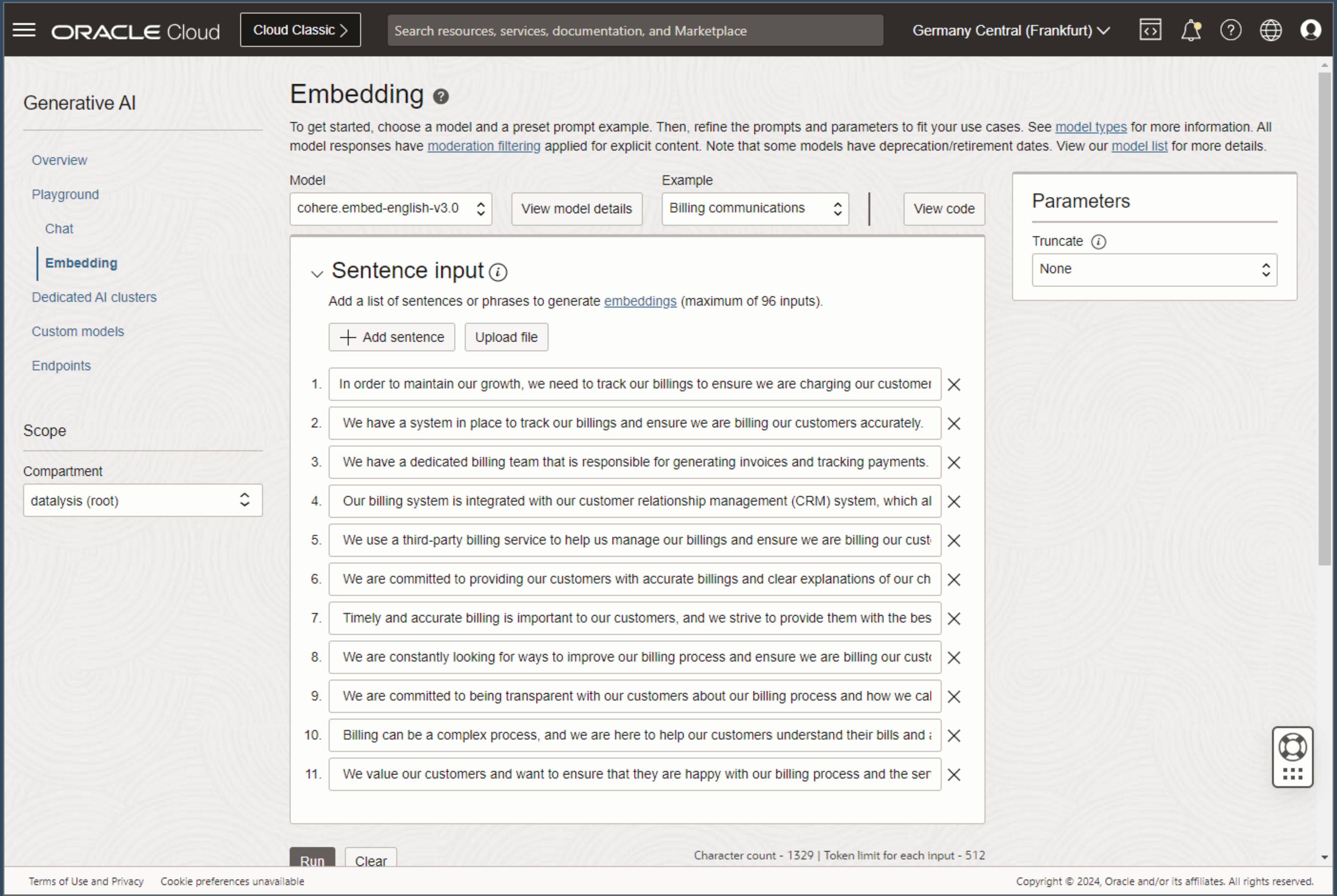
OCI Generative AI combines various services into one in Oracle Cloud.

- On-demand, pre-trained, models for embedding and chat LLMs
- Dedicated AI clusters ($$$)
- Custom models: from one of the available pre-trained model, fine-tune it with your own dataset

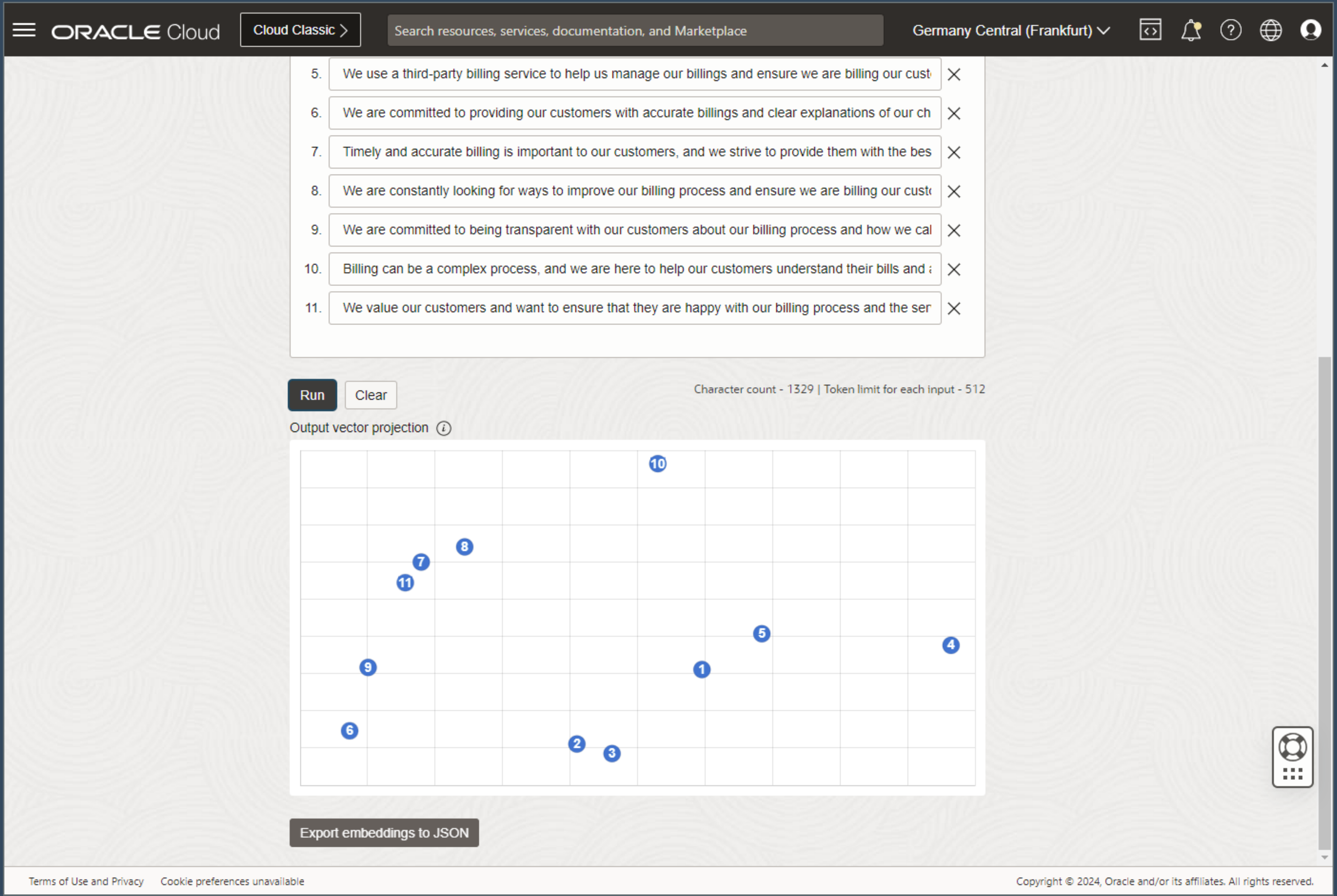
Available only in a limited number of regions, and not all models are available everywhere: https://docs.oracle.com/en-us/iaas/Content/generative-ai/pretrained-models.htm

Cheap and easily accessible for small tasks, calculate you potential costs for large activities (just to be aware of the cost involved).

OCI GenAI: Embedding



OCI GenAI: Embedding



OCI GenAI

— Embedding Models

Convert text to vector embeddings to use in applications for semantic searches, text classification, or text clustering.

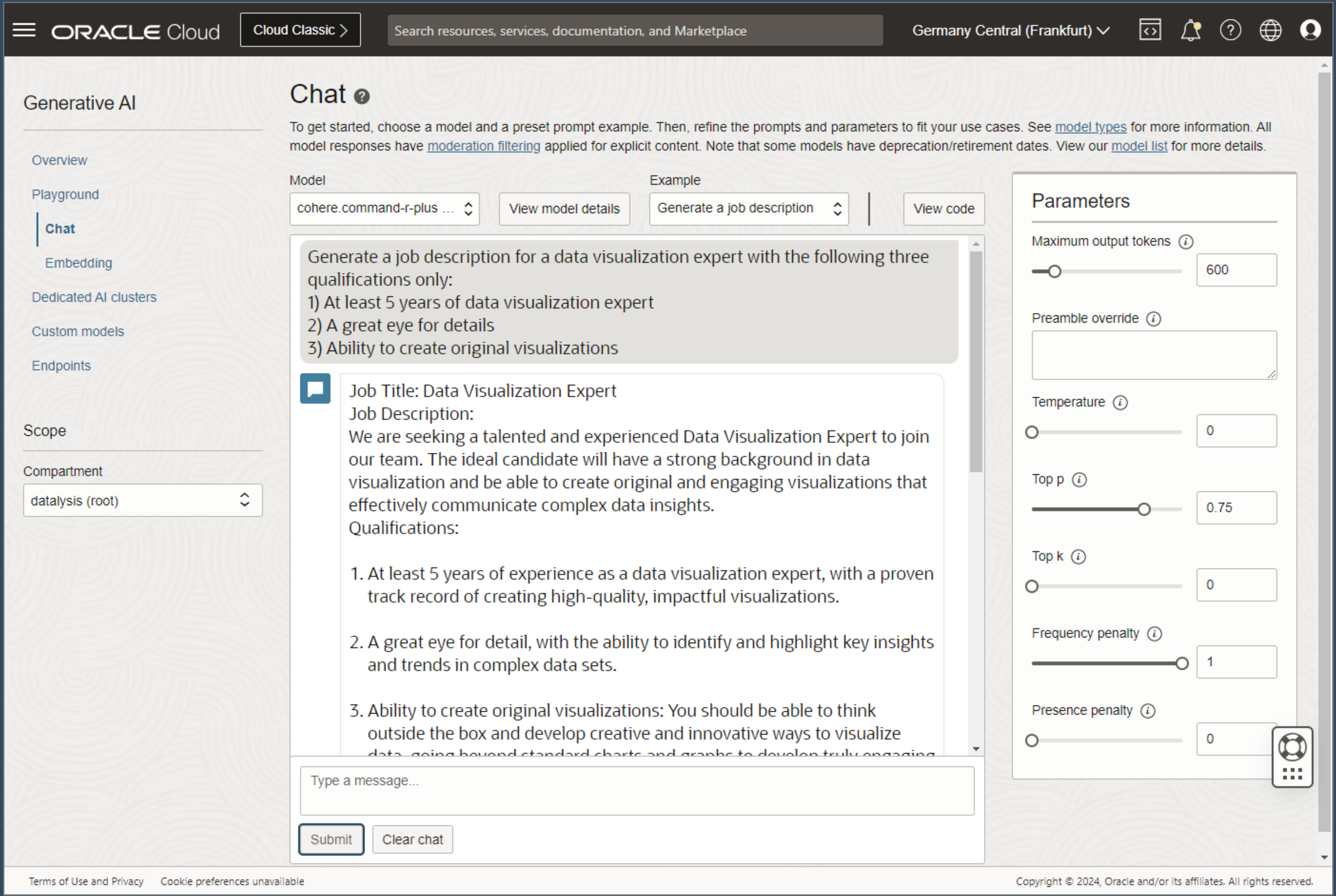
Model	Available in These Regions	Key Features
`cohere.embed-english-v3.0`	• **Brazil East (Sao Paulo)** • **Germany Central (Frankfurt)** • **UK South (London)** • **US Midwest (Chicago)**	• English or multilingual. • Model creates a 1024-dimensional vector for each embedding. • Maximum 96 sentences per run. • Maximum 512 tokens per embedding.
`cohere.embed-multilingual-v3.0`	• **Brazil East (Sao Paulo)** • **Germany Central (Frankfurt)** • **UK South (London)** • **US Midwest (Chicago)**	• English or multilingual. • Model creates a 1024-dimensional vector for each embedding. • Maximum 96 sentences per run. • Maximum 512 tokens per embedding.
`cohere.embed-english-light-v3.0`	• **US Midwest (Chicago)**	• Light models are smaller and faster than the original models. • English or multilingual. • Model creates a 384-dimensional vector for each embedding. • Maximum 96 sentences per run. • Maximum 512 tokens per embedding.
`cohere.embed-multilingual-light-v3.0`	• **US Midwest (Chicago)**	• Light models are smaller and faster than the original models.

Expand All Expandable Areas

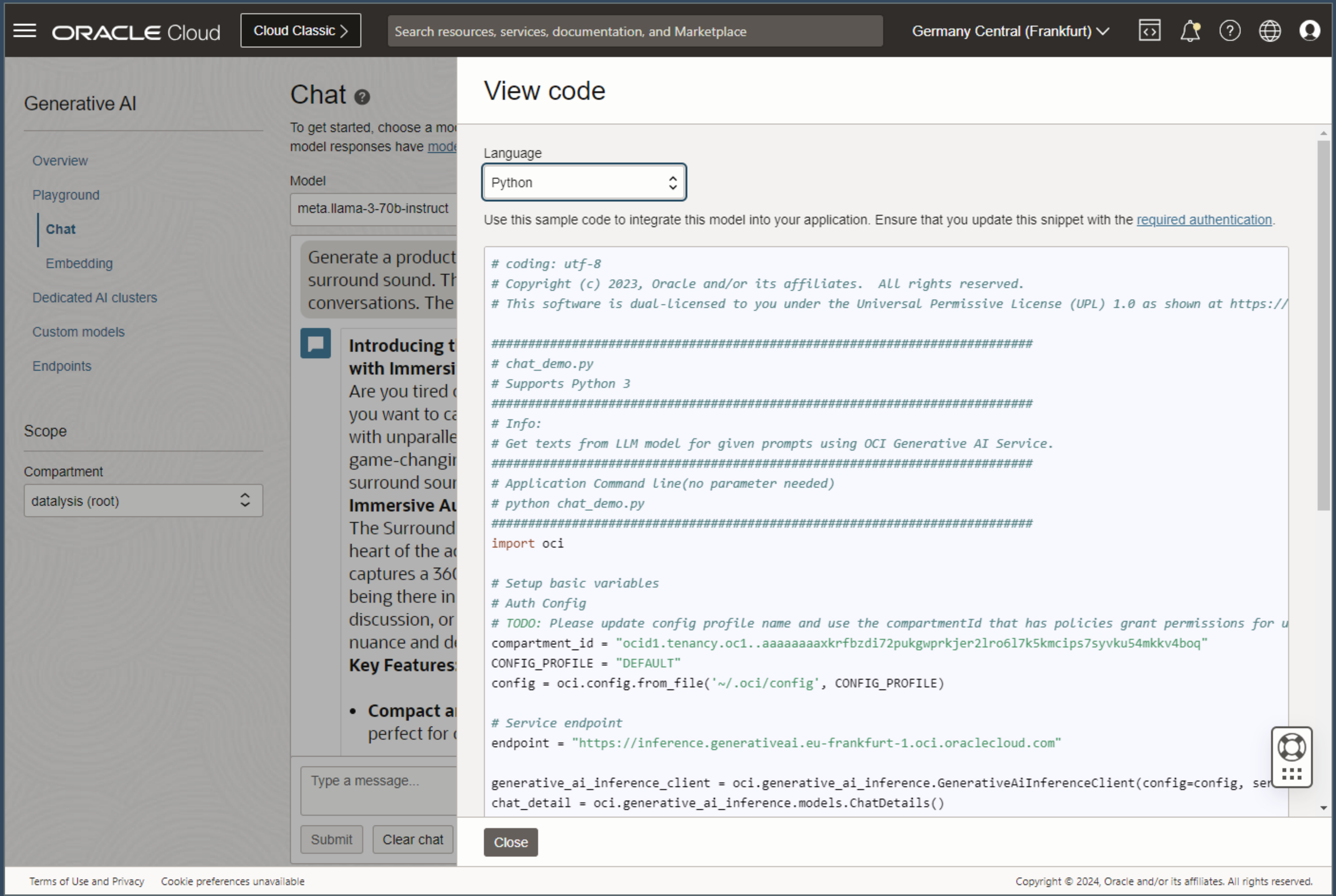
Was this article helpful?

Updated 2024-08-08

OCI GenAI: Chat



OCI GenAI: Chat



Java also available

OCI GenAI: Chat

— Chat Models (New)

Ask questions and get conversational responses through an AI chatbot.

Model	Available in These Regions	Key Features
`cohere.command-r-plus v1.2`	• **Brazil East (Sao Paulo)** • **Germany Central (Frankfurt)** • **UK South (London)** • **US Midwest (Chicago)**	• User prompt can be up to 128,000 tokens, and response can be up to 4000 tokens for each run. • Optimized for complex tasks, offers advanced language understanding, higher capacity, and more nuanced responses, and can maintain context from its long conversation history of 128,000 tokens. Also ideal for question-answering, sentiment analysis, and information retrieval.
`cohere.command-r-16k v1.2`	• **Brazil East (Sao Paulo)** • **Germany Central (Frankfurt)** • **UK South (London)** • **US Midwest (Chicago)**	• User prompt can be up to 16,000 tokens, and response can be up to 4000 tokens for each run. • Optimized for conversational interaction and long context tasks. Ideal for text generation, summarization, translation, or text-based classification. • **You can fine-tune this model with your dataset.**
`meta.llama-3-70b-instruct v1.0`	• **Brazil East (Sao Paulo)** • **Germany Central (Frankfurt)** • **UK South (London)**	• Model has 70 billion parameters. • User prompt and response can be up to 8000 tokens for each run. • **You can fine-tune this model with your dataset.**

Expand All Expandable Areas

Was this article helpful?

Updated 2024-08-08

Let’s start!

Everything begins with data

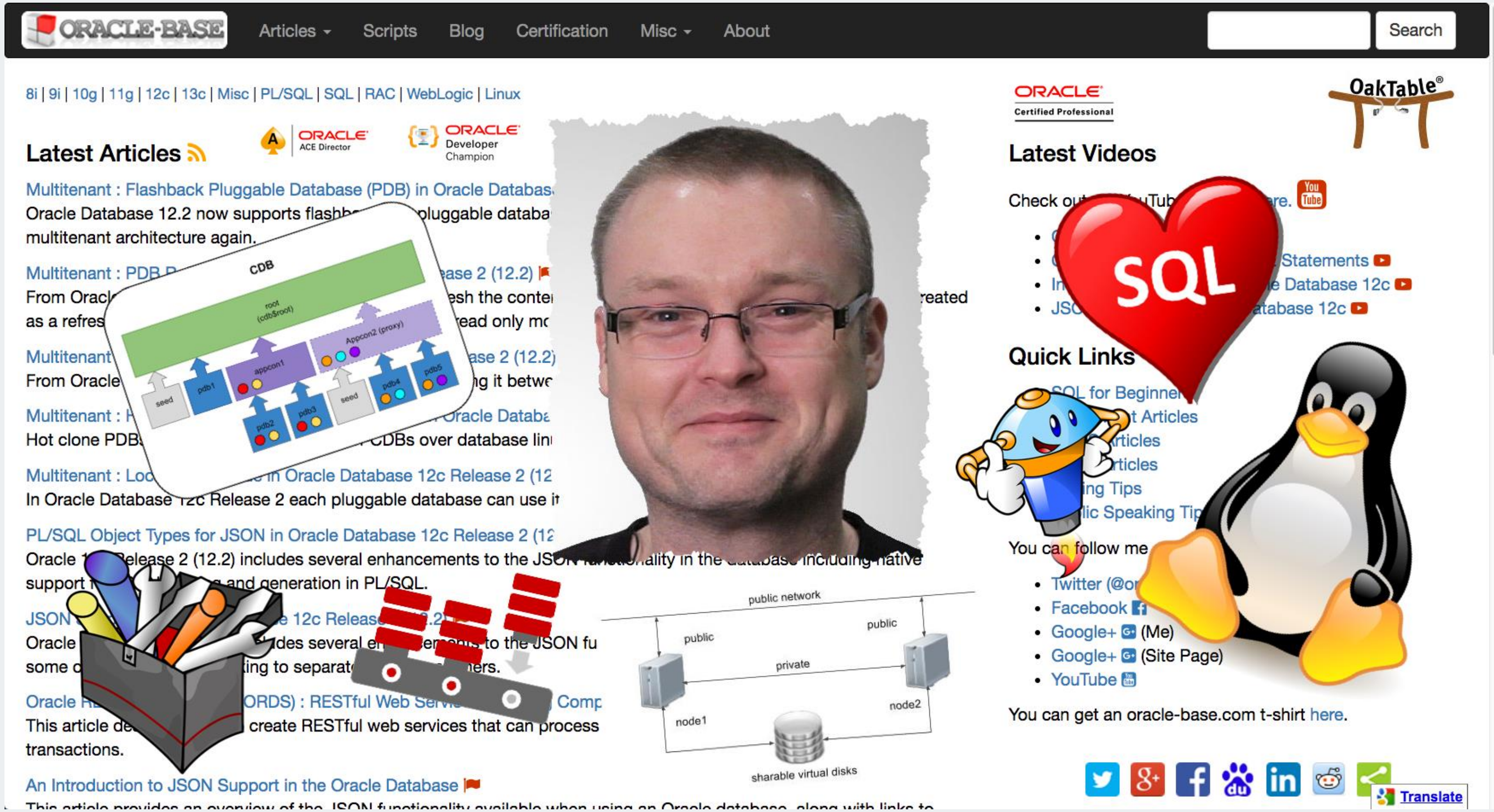
OracleBaseGPT

When googling something about the database, you often find oracle-base.com in the first results.

You maybe never met Tim Hall, but you know his website.

What happen if we take a chatbot powered by LLM and feed him with oracle-base.com articles?

Can we get a personal OracleBaseGPT?



But before…

Do not randomly steal content online from somebody's website without being allowed to do so: you can maybe navigate the website, but you don't own the content, you can't always do whatever you want with it.

Do not kill websites by sending tons of requests just because you are playing with some embedding exercise.

Be respectful, ask permission.

This presentation is done with the permission of Tim Hall to use his articles.

Data: know your source

Any ML, AI, LLM usage will generally follow the same rule:

- “garbage in, garbage out”

You must know your source, its format, and type.

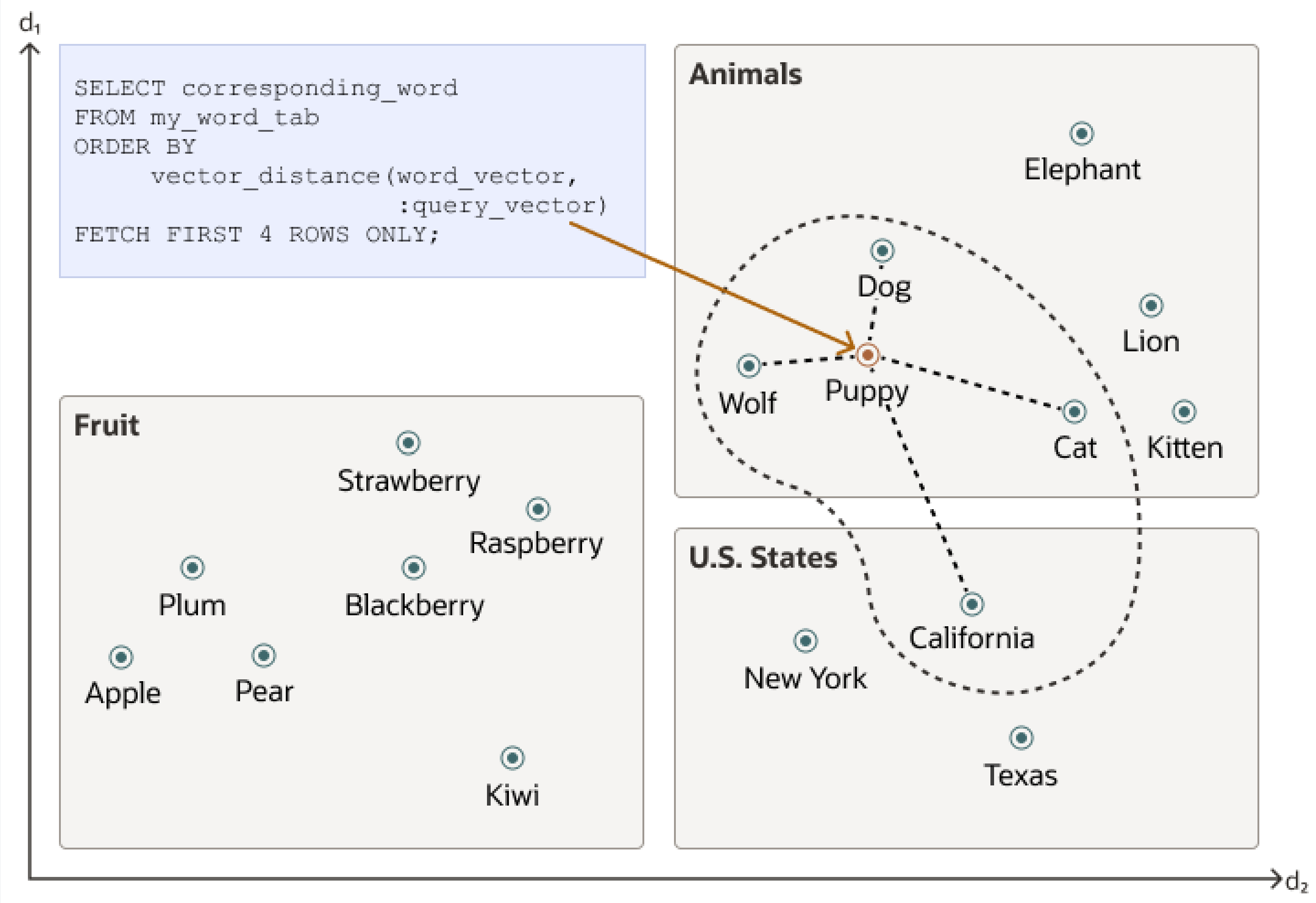
Data: know your source

This is even more important for RAG activities:

- Generate an embedding of garbage content and you will get a fairly pointless embedding.

Reality isn't as simple as the images in the Oracle Database documentation...

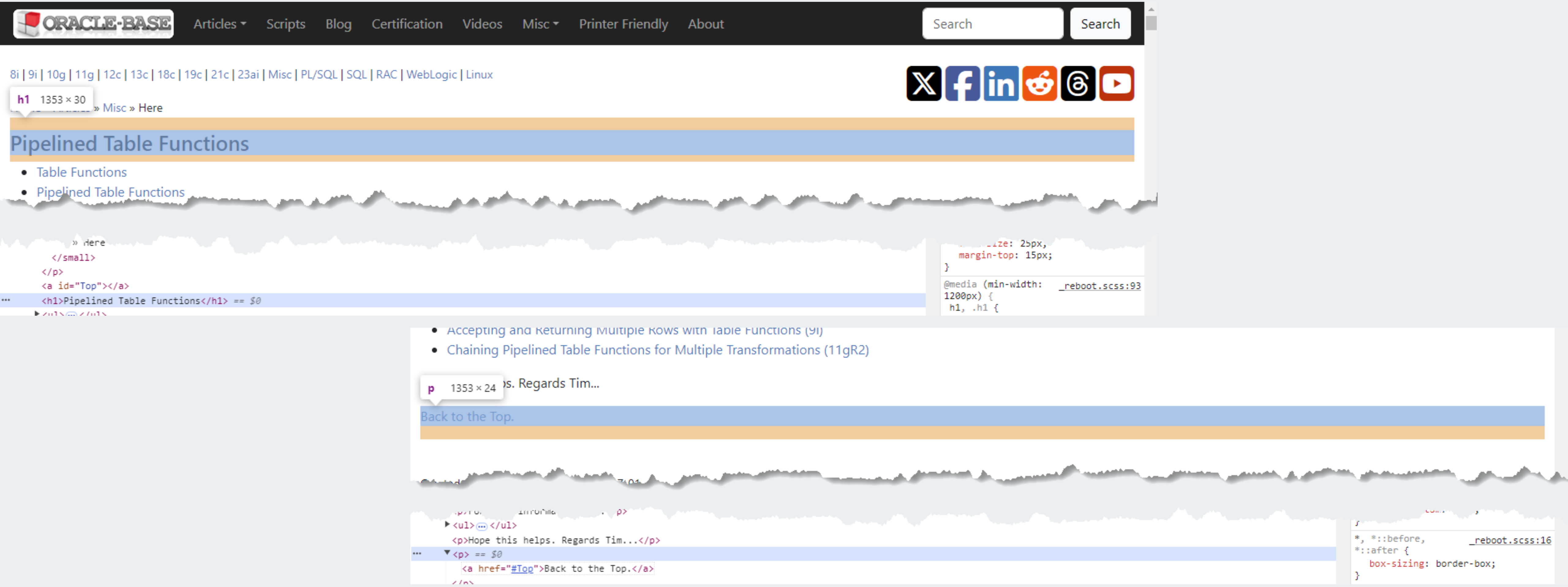
Fruit, Animals or U.S. States all nicely packed together is an illustration of an idealistic vision of how vectors looks like



Data: know your source

Are header and footer useful? Not really...

Identify the real content and how to get rid of the “noise” around it.



Start by chunking

Chunking: where the fun really starts...

The embedding of your content will generate a single vector for each input you provide.

Should you provide a whole document as one piece?

You will get a single vector for it! How precisely could a single vector represent a whole document and every concept contained in it?

Should you provide each word as a single piece?

You will get a vector for each word, but no context at all because your words will be individual, separate, pieces.

Finding the best way to chunk your documents can be very challenging!

- Chunking: splitting your source document into smaller pieces

Chunking: where the fun really starts...

If you search online you will find about as many chunking strategies as people talking about it...

Just to name a few simple techniques:

- Fixed size chunking
- Recursive chunking
- Structure based chunking

Chunking: where the fun really starts...

Fixed size chunking is the easiest. Chunk your content into pieces of fixed size.

An example:

- Chunk size: 100 characters
- Overlap: 0 characters

Many words cut in half,
sentences cut in half,
can we safely assume
the meaning is preserved?

A tablescan would, of course, still be expensive and do far more work than needed. Ideally we need an
index to be able to find the rows that match all three equality predicates. However, we can see that
there are roughly 2 rows per value for item_key, so an index on just (item_key) might be good enough.
Averages, of course, can be very misleading: we might have one row for each of 3.5M values of
item_key and one value of item_key with 3.5M rows in our table, so we do need to know more about
the data distribution before we can make any solid suggestions.

https://jonathanlewis.wordpress.com/2024/08/07/indexing-4/

Chunking: where the fun really starts...

Recursive chunking is a bit smarter. The content is split based on a set of separators. If the obtained chunk is bigger than the expected size, another chunking loop is performed on that piece till the result is “small enough”.

Separators can be “\n”, “.”, “,”, “ “ etc.

An example:

- Chunk size: 100 characters
- Overlap: 0 characters

No word cut in half, sentences still cut in half, can we safely assume the meaning is preserved?

A tablescan would, of course, still be expensive and do far more work than needed. Ideally we need an

index to be able to find the rows that match all three equality predicates. However, we can see that

there are roughly 2 rows per value for item_key, so an index on just (item_key) might be good enough.

Averages, of course, can be very misleading: we might have one row for each of 3.5M values of

item_key and one value of item_key with 3.5M rows in our table, so we do need to know more about

the data distribution before we can make any solid suggestions.

https://jonathanlewis.wordpress.com/2024/08/07/indexing-4/

Chunking: where the fun really starts...

Structure based chunking works with the structure of the document. HTML has a structure, paragraphs, headings should be used (but you can abuse HTML massively and break and structured approach).

You can chunk by section of the HTML by headings tags: `<h1>, <h2>, <h3>`

You can then chunk by paragraph, knowing to which section it belongs. Maybe even “enriching” the paragraph by adding the heading to it.

The issue is that nothing guarantee the size of a paragraph, it could still be very long.

Chunking methods can be combined: structure based first, recursive chunking after on the result of the structure chunking.

After chunking comes embedding, turning inputs into vectors

Embedding in the database

- Embedding in the database requires to first load a model able to perform embedding. Oracle provides some pre-trained models in the ONNX format that can be loaded.

```
begin
  dbms_vector.drop_onnx_model (
    model_name => 'ALL_MINILM_L12_V2',
    force => true);

  dbms_vector.load_onnx_model (
    directory  => 'model_dir',
    file_name  => 'all_MiniLM_L12_v2.onnx',
    model_name => 'ALL_MINILM_L12_V2');
end;
/
```

```
select
vector_embedding(all_minilm_l12_v2 using 'Quick test' as data) AS my_vector;
```

Embedding from the database using a webservice

The DBMS_VECTOR and DBMS_VECTOR_CHAIN packages can call external webservice for embedding.

You first need to create a credential for the service with the required authentication details.

You also need to allow to call the external URL.

```
-- execute an embedding task for OCI GenAI
declare
  input clob;
  params clob;
  v vector;
begin
  dbms_output.put_line('Embedding task for OCI GenAI');
  input := 'Hello world';
  params := '
{
  "provider": "OCIGenAI",
  "credential_name": "CRED_OCI_GENAI",
  "url": "https://inference.generativeai.eu-frankfurt-1.oci...",
  "model": "cohere.embed-english-v3.0",
  "batch_size": 1
}';

  v := dbms_vector.utl_to_embedding(input, json(params));
  dbms_output.put_line(vector_serialize(v));
exception
  when OTHERS THEN
    DBMS_OUTPUT.PUT_LINE (SQLERRM);
    DBMS_OUTPUT.PUT_LINE (SQLCODE);
end;
/
```

How I do chunking and embedding for OracleBaseGPT

Chunking outside the database

HTML splitter first, recursive splitter after.

Using LangChain and Hugging Face packages.

Trying to chunk based on token size using a tokenizer for the embedding model.

```
### Does still create chunks too big for the embedding model (512 tokens) from time to time...

from langchain_text_splitters import HTMLHeaderTextSplitter, RecursiveCharacterTextSplitter
from langchain_core.documents import Document
from transformers import AutoTokenizer

# define a tokenizer for Cohere-embed-english-v3.0 to try to count up to 512 tokens per chunk
model_id = "Cohere/Cohere-embed-english-v3.0"
tokenizer = AutoTokenizer.from_pretrained(model_id)

# Recursive splitter based on chunk size counted in tokens
text_splitter = RecursiveCharacterTextSplitter.from_huggingface_tokenizer(
    tokenizer, chunk_size=400, chunk_overlap=10
)

# list of headers to split on
headers_to_split_on = [
    ("h1", "Header 1"),
    ("h2", "Header 2"),
    ("h3", "Header 3"),
]

# HTML splitter to split by the structure of the web page
html_splitter = HTMLHeaderTextSplitter(headers_to_split_on=headers_to_split_on)

def chunk_html_content(id: int, url: str, content: str) -> list[Document]:
    # html split on headers
    html_header_splits = html_splitter.split_text(content)

    # split chunks
    chunks = text_splitter.split_documents(html_header_splits)

    # dirty add of metadata
    for idx, c in enumerate(chunks):
        chunks[idx].metadata = c.metadata | {'source': url, 'id': id}

    return chunks
```

Embedding outside the database

Embedding is done right after chunking, in Python. Using the `cohere.embed-english-v3.0` model of OCI GenAI.

A parameter is set to truncate inputs if too large (512 tokens max) and not stop with an error.

The result is a list of vectors.

```
import oci

# setup basic variables
compartment_id = "ocid1.compartment.oc1..aaaaaaaaryltzreesv2f4pa6vj4lnf52yemrtywqo5pcwse3pknps7rwk3fq"
oci_config = oci.config.from_file('~/.oci/config', "DEFAULT")

# service endpoint
endpoint = "https://inference.generativeai.eu-frankfurt-1.oci.oraclecloud.com"

# define the GenAI client to execute embedding calls
generative_ai_inference_client = oci.generative_ai_inference.GenerativeAiInferenceClient(
    config=oci_config,
    service_endpoint=endpoint,
    retry_strategy=oci.retry.NoneRetryStrategy(),
    timeout=(10,240)
)

def embed_list_of_chunks(chunks: list) -> list:
    inputs = [chunk.page_content for chunk in chunks]

    embed_text_detail = oci.generative_ai_inference.models.EmbedTextDetails()
    embed_text_detail.serving_mode = oci.generative_ai_inference.models.OnDemandServingMode(model_id="cohere.embed-english-v
    embed_text_detail.inputs = inputs
    embed_text_detail.truncate = "END" # if the chunk is too big, truncate the end
    embed_text_detail.compartment_id = compartment_id
    embed_text_response = generative_ai_inference_client.embed_text(embed_text_detail)

    return embed_text_response.data.embeddings
```

Chunks and vectors are inserted in Oracle Database 23ai

An index can be added on the vector column to make queries faster.

The dataset is quite small: only 9'608 chunks (for the 1'327 articles of oracle-base.com).
The embeddings column is of type `VECTOR(1024, FLOAT64)`

```
1  select * from webpages_chunks order by 1, 2;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SQL HISTORY

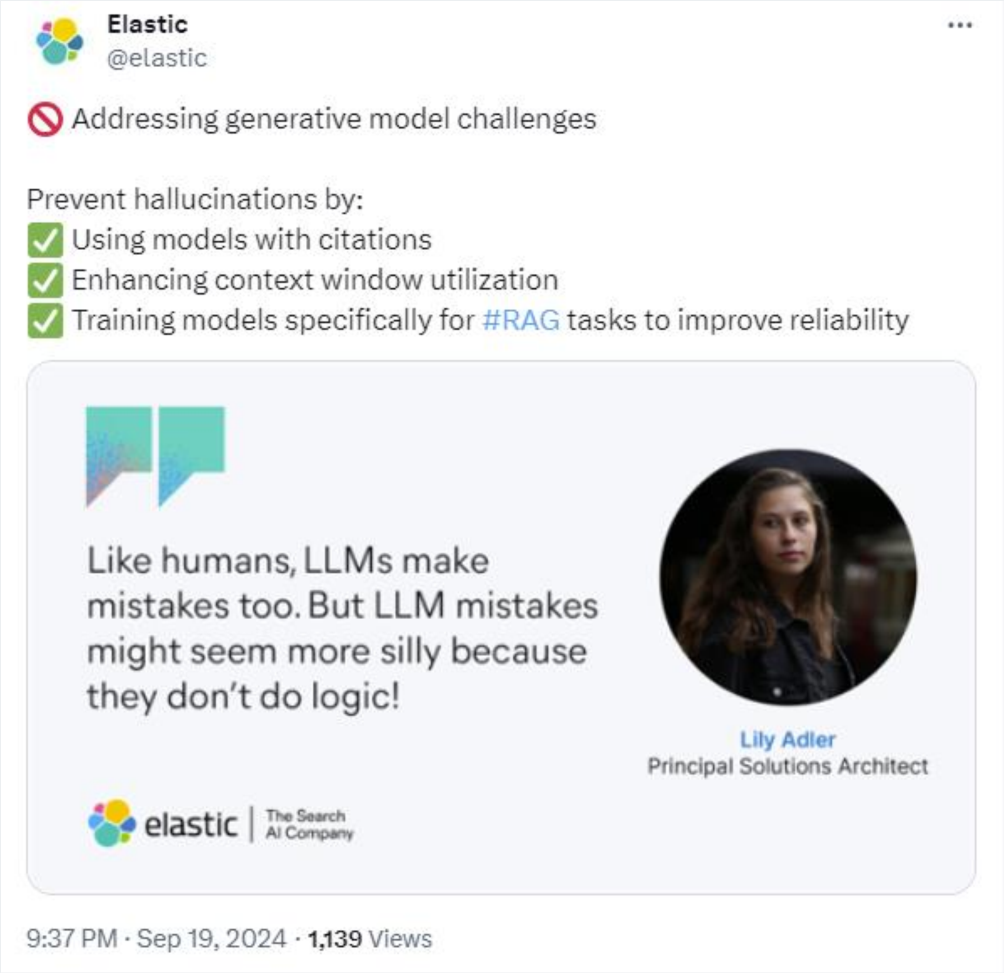
Fetched 200 rows in 0.815 seconds

	ID	WEBPAGE_ID	URL	CONTENT	EMBEDDING
1	31216	125	https://oracle-base.com/artic…	Oracle Data Pump was introduced in Oracle 10g. This article provi…	[-2.18505859E-002,-2.48260498E-002,-4.17480469E-002,-1.83105469E-002,-2.272
2	31217	125	https://oracle-base.com/artic…	The COMPRESSION parameter allows you to decide what, if anything,…	[-2.0111084E-002,-2.35900879E-002,5.37872314E-003,-2.11334229E-002,-4.21752
3	31218	125	https://oracle-base.com/artic…	Data pump encryption is an Enterprise Edition feature, so the par…	[-4.01916504E-002,-7.07626343E-003,-2.66113281E-002,1.95922852E-002,-2.5924
4	31219	125	https://oracle-base.com/artic…	The use of encryption is controlled by a combination of the ENCRY…	[-7.51113892E-003,-2.91595459E-002,1.11618042E-002,-1.16729736E-002,-5.0628
5	31220	125	https://oracle-base.com/artic…	The ENCRYPTION_ALGORITHM parameter specifies the encryption algor…	[-4.67834473E-002,-1.3168335E-002,5.68771362E-003,5.30395508E-002,-7.049560
6	31221	125	https://oracle-base.com/artic…	The ENCRYPTION_MODE parameter specifies the type of security used…	[-2.13928223E-002,-3.49121094E-002,2.32849121E-002,-1.67236328E-002,-5.6030
7	31222	125	https://oracle-base.com/artic…	The TRANSPORTABLE parameter is similar to the TRANSPORT_TABLESPAC…	[6.23321533E-003,-2.16674805E-002,3.26538086E-002,-1.48391724E-003,-3.26538
8	31223	125	https://oracle-base.com/artic…	The PARTITION_OPTIONS parameter determines how partitions will be…	[1.53808594E-002,-7.2555542E-003,2.14538574E-002,-3.96118164E-002,-4.306030
9	31224	125	https://oracle-base.com/artic…	The REUSE_DUMPFILES parameter can be used to prevent errors being…	[4.22668457E-003,2.9296875E-003,4.97436523E-002,-2.38647461E-002,-4.8828125
10	31225	125	https://oracle-base.com/artic…	This parameter allows a table to be renamed during the import ope…	[-1.8157959E-002,1.75323486E-002,-7.63702393E-003,-4.21905518E-003,-1.86309
11	31226	125	https://oracle-base.com/artic…	During import operations using the external table acces method, s…	[1.18865967E-002,-1.35269165E-002,-2.46734619E-002,-8.09326172E-002,1.00326
12	31227	125	https://oracle-base.com/artic…	During an export, if XMLTYPE columns are currently stored as CLOB…	[-1.30767822E-002,-5.0994873E-002,1.54571533E-002,-1.42288208E-002,-5.72509
13	31228	125	https://oracle-base.com/artic…	During export and import operations, the REMAP_DATA parameter all…	[-2.4520874E-002,-1.00021362E-002,1.50909424E-002,-3.71360779E-003,-3.53393
14	31229	125	https://oracle-base.com/artic…	Worker processes that have stopped due to certain errors will now…	[1.63421631E-002,-3.43933105E-002,-2.39715576E-002,-5.00183105E-002,-1.9226
15	31230	127	https://oracle-base.com/artic…	The Database Replay functionality of Oracle 11g allows you to cap…	[-3.4942627E-003,-4.54711914E-002,-2.56652832E-002,-9.10644531E-002,-4.9438

Generate answers with a LLM

LLMs are a bit like teaching to kids…

By design a LLM isn't really smart. Many even struggle to perform mathematical operations and logical thinking. And it's expected: a LLM isn't a "brain", it does generate text.



LLMs are a bit like teaching to kids…

By design a LLM isn't really smart. Many even struggle to perform mathematical operations and logical thinking. And it's expected: a LLM isn't a "brain", it does generate text.

The quality of the prompt is what will decide if the LLM will understand your request or not.

LLMs also don't really have any memory, the memory is provided as part of the prompt (the input message).

Current LLMs in GenAI are chat models, they do have roles for the messages:

- System : provides general instructions to the model, how it should behave
- User : the messages from the users, the "questions"
- Chatbot / Assistant : the messages generated by the model as answer to the inputs received

Models have different prompt formats

Meta | Documentation | Trust & Safety | Community | Try 405B on Meta AI ↗ | Download models

Overview

Models

- Llama 3.1
- Llama Guard 3
- Prompt Guard
- Llama 3
- Llama Guard 2
- Other models

Getting the Models

- Meta
- Hugging Face
- Kaggle
- 405B Partners

Running Llama

- Linux
- Windows
- Mac
- Cloud

Model Cards & Prompt formats

Llama 3

On this page

- Llama 3
  - Model Card
  - Special Tokens used with Llama 3
  - Llama 3
  - Llama 3 Instruct

Model Card

You can find details about this model in the model card.

Special Tokens used with Llama 3

<|begin_of_text|>: This is equivalent to the BOS token

<|eot_id|>: This signifies the end of the message in a turn.

<|start_header_id|>{role}<|end_header_id|>: These tokens enclose the role for a particular message. The possible roles can be: system, user, assistant.

<|end_of_text|>: This is equivalent to the EOS token. On generating this token, Llama 3 will cease to generate more tokens.

A prompt can optionally contain a single system message, or multiple alternating user and assistant messages, but always ends with the last user message followed by the assistant header.

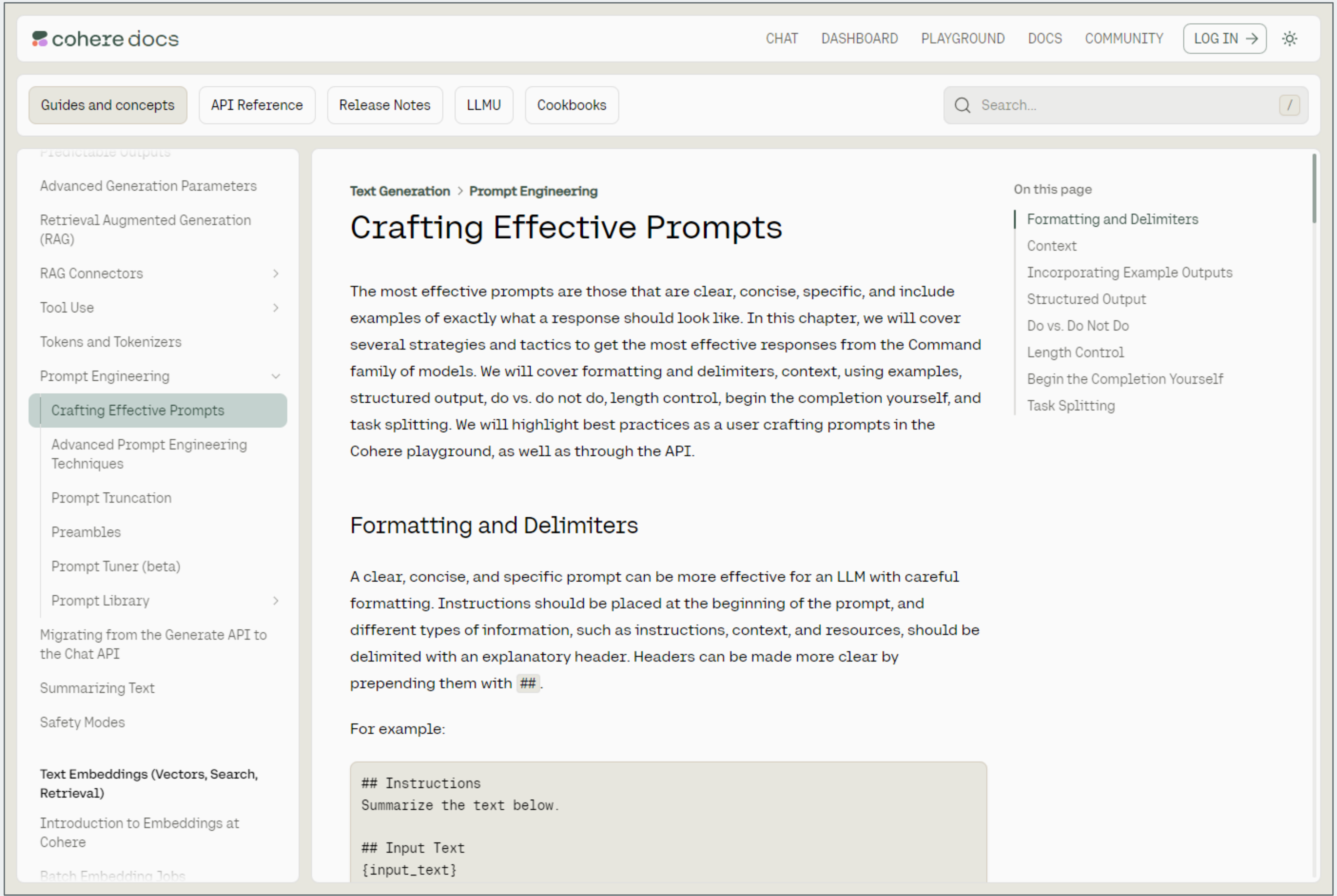
Llama 3

Code to produce this prompt format can be found here.

Note: Newlines (0x0A) are part of the prompt format, for clarity in the example, they have been represented as actual new lines.

https://www.llama.com/docs/model-cards-and-prompt-formats/meta-llama-3

Models have different prompt formats



cohere docs

CHAT DASHBOARD PLAYGROUND DOCS COMMUNITY LOG IN

Guides and concepts | API Reference | Release Notes | LLMU | Cookbooks

Search... /

Text Generation > Prompt Engineering

Crafting Effective Prompts

The most effective prompts are those that are clear, concise, specific, and include examples of exactly what a response should look like. In this chapter, we will cover several strategies and tactics to get the most effective responses from the Command family of models. We will cover formatting and delimiters, context, using examples, structured output, do vs. do not do, length control, begin the completion yourself, and task splitting. We will highlight best practices as a user crafting prompts in the Cohere playground, as well as through the API.

Formatting and Delimiters

A clear, concise, and specific prompt can be more effective for an LLM with careful formatting. Instructions should be placed at the beginning of the prompt, and different types of information, such as instructions, context, and resources, should be delimited with an explanatory header. Headers can be made more clear by prepending them with `##`.

For example:

```
## Instructions
Summarize the text below.

## Input Text
{input_text}
```

On this page

Models can have special features like "documents"

cohere docs

CHAT DASHBOARD PLAYGROUND DOCS COMMUNITY LOG IN →

Guides and concepts | API Reference | Release Notes | LLMU | Cookbooks

Search... /

Predictable Outputs
- Advanced Generation Parameters
- Retrieval Augmented Generation (RAG)
- RAG Connectors
- Tool Use
- Tokens and Tokenizers
- Prompt Engineering
  - Crafting Effective Prompts
  - Advanced Prompt Engineering Techniques
  - Prompt Truncation
  - Preambles
  - Prompt Tuner (beta)
  - Prompt Library
- Migrating from the Generate API to the Chat API
- Summarizing Text
- Safety Modes

Text Embeddings (Vectors, Search, Retrieval)
- Introduction to Embeddings at Cohere
- Batch Embedding Jobs

For the example above, we can split the original news article into different sections and attach them via the `documents` parameter. The Chat API will then provide us not only with the completion but also citations that ground information from the documents. See the following:

PYTHON

```python
# Sections from the original news article
documents = [
    {"title": "background", "snippet": "From the beginning of the IIHF Ice
    {"title": "expectations", "snippet": "At the time, the National Hockey
    {"title": "legacy", "snippet": "While Canada won the series, the Soviet
]

# New request
query = '''The 1972 Canada-USSR Summit Series was an eight-game ice hockey

# Call the model
response = co.chat(
  message=query,
  documents=documents,
  model="command-r-plus",
  temperature=0.3
)
```

The model returns a high quality summary in `response.text`:

```
The 1972 Canada-USSR Summit Series marked a significant moment in the history o
showcasing a high-stakes competition between the Canadian national team and the
elite hockey squad. Here are some key points about the series:
```

On this page
- Formatting and Delimiters
- Context
- Incorporating Example Outputs
- Structured Output
- Do vs. Do Not Do
- Length Control
- Begin the Completion Yourself
- Task Splitting

Models can have special features like "documents"

oci
2.134.0

Search docs

- Installation
- Configuration
- Using FIPS-validated Libraries
- Forward Compatibility
- New Region Support
- Backward Compatibility
- Quickstart
- Known Issues
- Logging
- Exception handling
- Uploading Large Objects
- Raw Requests
- Composite Operations and Waiters
- Pagination
- API Reference
  - Access Governance Cp
  - Adm
  - Ai Anomaly Detection
  - Ai Document
  - Ai Language
  - Ai Speech
  - Ai Vision
  - Analytics
  - Announcements Service
  - Apigateway
  - Apm Config

class `oci.generative_ai_inference.models.CohereChatRequest`(***kwargs*)

Bases: `oci.generative_ai_inference.models.base_chat_request.BaseChatRequest`

Details for the chat request for Cohere models.

Attributes

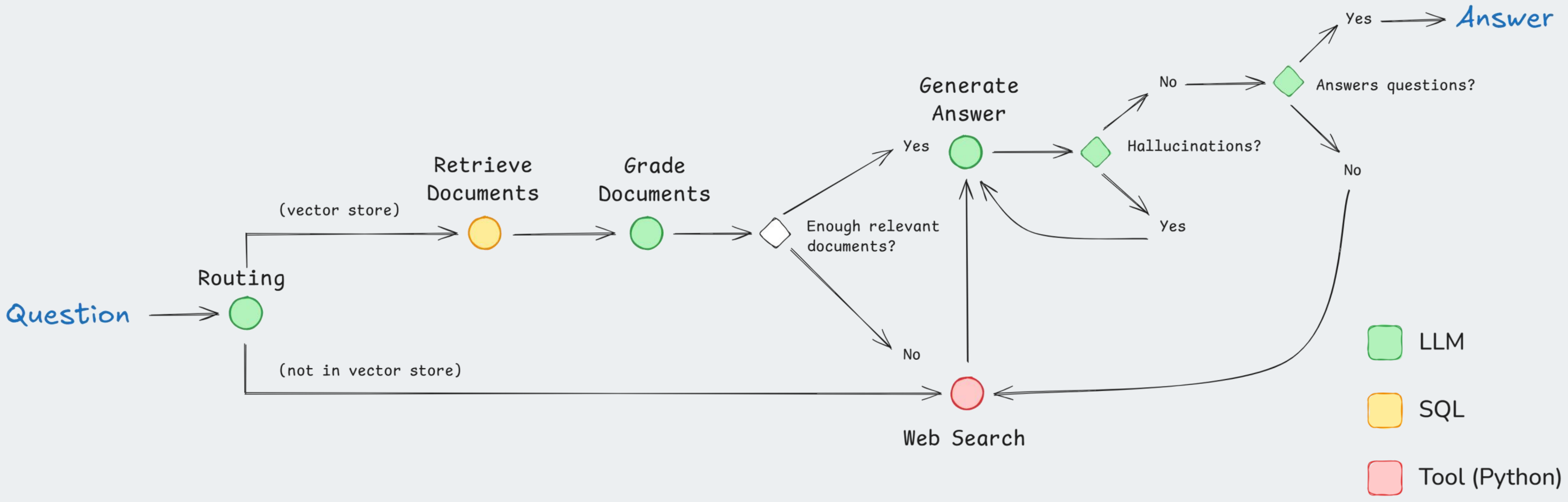
`API_FORMAT_COHERE`	str(object='') -> str
`API_FORMAT_GENERIC`	str(object='') -> str
`CITATION_QUALITY_ACCURATE`	A constant which can be used with the citation_quality proper
`CITATION_QUALITY_FAST`	A constant which can be used with the citation_quality proper
`PROMPT_TRUNCATION_AUTO_PRESERVE_ORDER`	A constant which can be used with the prompt_truncation pro
`PROMPT_TRUNCATION_OFF`	A constant which can be used with the prompt_truncation pro
`api_format`	**[Required]** Gets the api_format of this BaseChatRequest.
`chat_history`	Gets the chat_history of this CohereChatRequest.
`citation_quality`	Gets the citation_quality of this CohereChatRequest.
`documents`	Gets the documents of this CohereChatRequest.
`frequency_penalty`	Gets the frequency_penalty of this CohereChatRequest.
`is_echo`	Gets the is_echo of this CohereChatRequest.
`is_force_single_step`	Gets the is_force_single_step of this CohereChatRequest.
`is_raw_prompting`	Gets the is_raw_prompting of this CohereChatRequest.
`is_search_queries_only`	Gets the is_search_queries_only of this CohereChatRequest.
`is_stream`	Gets the is_stream of this CohereChatRequest.
`max_tokens`	Gets the max_tokens of this CohereChatRequest.
`message`	**[Required]** Gets the message of this CohereChatRequest.
`preamble_override`	Gets the preamble_override of this CohereChatRequest.

Read the documentation of the model creator.
Then read the oracle documentation and see if you find the same features.

OracleBaseGPT uses LLM for various tasks

For OracleBaseGPT the model `meta.llama-3-70b-instruct v1.0` is being used for all the LLMs tasks.

The whole process, the flow of activities, is managed by LangGraph (a Phyton package designed for exactly that: define flows that involve cycles, essential for most agentic architectures).



OracleBaseGPT: some prompts examples

How to use a LLM to decide if the content of my Oracle Database 23ai table (the chunks and vectors) is going to be able to answer a question or it's better to directly perform a web search?

Being as explicit and precise as possible in the instruction given to the LLM on how it has to behave passing a "system" message.

```
<|begin_of_text|>
<|start_header_id|>system<|end_header_id|> You are an expert at routing a user question to a vectorstore or web search. Use the vectorstore for questions on Oracle, Oracle database, SQL, PL/SQL, query and performance tuning. You do not need to be stringent with the keywords in the question related to these topics. Otherwise, use web-search. Give a binary choice 'web_search' or 'vectorstore' based on the question. Return the a JSON with a single key 'datasource' and no preamble or explanation.
Question to route: {question} <|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

For Llama you can see how the last part of the input is always to give the hand back to the LLM to "speak" when using prompts with the proper tokens.

OracleBaseGPT: some prompts examples

Asking the LLM to generate an answer to the question, saying it is an Oracle Database assistant and its answer should be concise and 3 sentences maximum.

```
<|begin_of_text|>
<|start_header_id|>system<|end_header_id|> You are an Oracle Database assistant for question-answering tasks.
Use the following pieces of retrieved context to answer the question. If you don't know the answer, just say that you don't know.
Use three sentences maximum and keep the answer concise <|eot_id|>
<|start_header_id|>user<|end_header_id|>
Question: {question}
Context: {context}
Answer: <|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

OracleBaseGPT: some prompts examples

Asking the LLM to evaluate if the answer another LLM call did generate is really based on the documents (the data) provided or not. And telling the LLM that the answer should be a JSON with a precise format, no freedom to say anything else.

```
<|begin_of_text|>
<|start_header_id|>system<|end_header_id|> You are a grader assessing whether
an answer is grounded in / supported by a set of facts. Give a binary 'yes' or 'no' score to indicate
whether the answer is grounded in / supported by a set of facts. Provide the binary score as a JSON with a
single key 'score' and no preamble or explanation. <|eot_id|>
<|start_header_id|>user<|end_header_id|> Here are the facts:

 -------

{documents}

 -------

Here is the answer: {generation}  <|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

A lot can be done in the database 23ai

No Python required.
Can be enough for a number of requirements…

Cleaning, Chunking and Embedding in the database

In Oracle Database 23ai, most tasks related to RAG, vectors, embeddings etc. are packed in 2 packages:

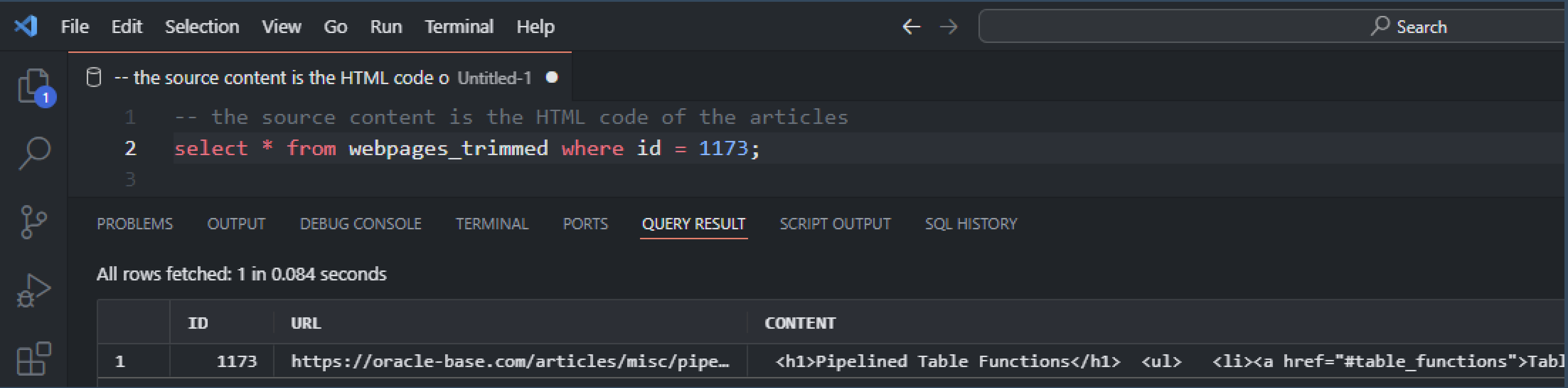
- DBMS_VECTOR
- DBMS_VECTOR_CHAIN

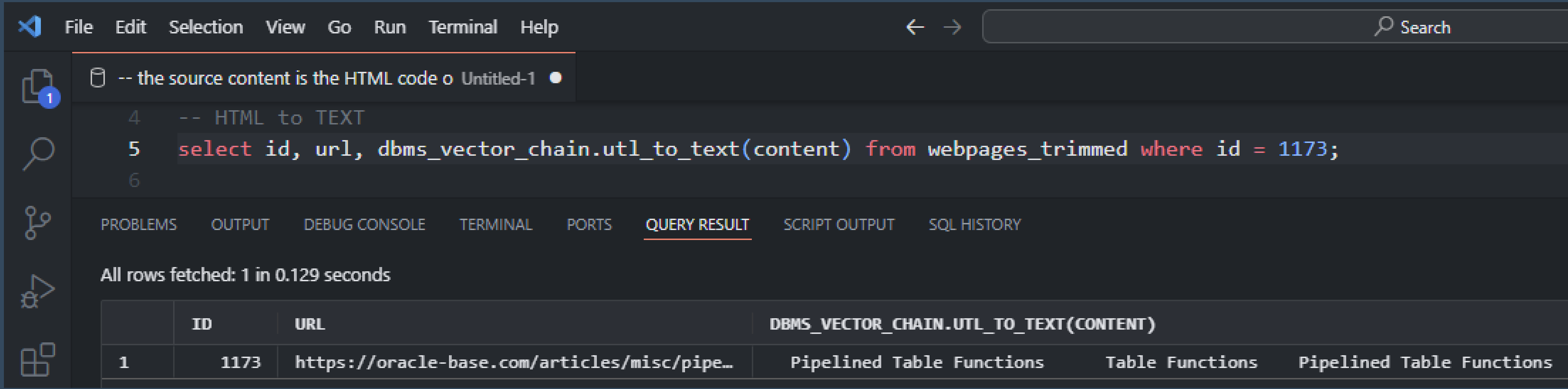
The documentation provides details on the available procedures and the various parameters. There are also some example provided. There seems to easily be mismatches between the documentation and what the current database does (because each release can change things and add new features), be ready for some debugging…

- https://docs.oracle.com/en/database/oracle/oracle-database/23/vecse/summary-dbms_vector-subprograms-arpls.html
- https://docs.oracle.com/en/database/oracle/oracle-database/23/vecse/summary-dbms_vector_chain-subprograms-arpls.html

HTML to text? Yes

Use DBMS_VECTOR_CHAIN.UTL_TO_TEXT



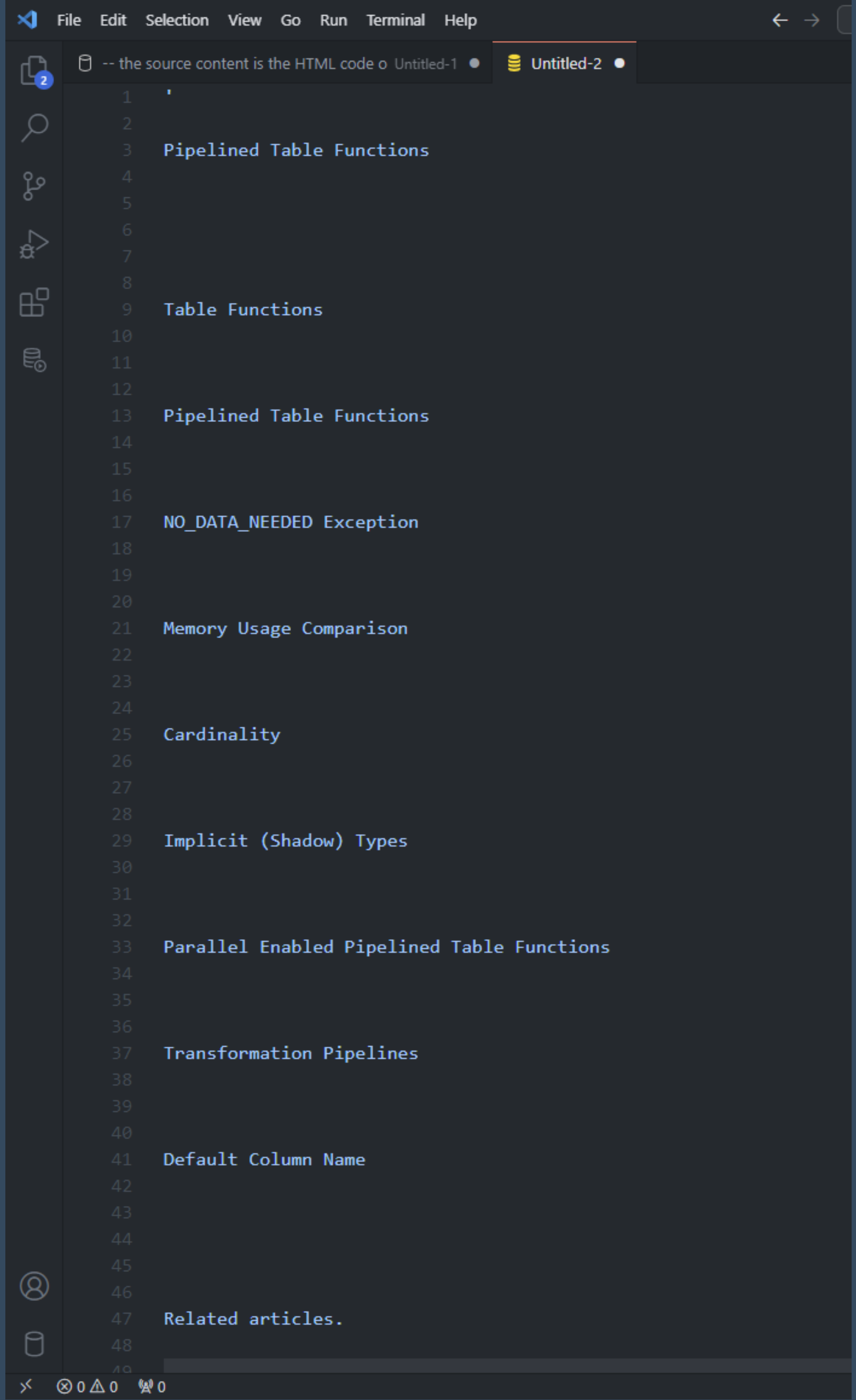


HTML to text? Yes

The result isn't the cleanest…

- No cleaning is done
- Lot of new lines added

But it's text, just text, without HTML tags



Chunking? Yes

Use DBMS_VECTOR_CHAIN.UTL_TO_CHUNKS

File Edit Selection View Go Run Terminal Help

Search

Untitled-1

```sql
select w.id, w.url,
   json_value(t.column_value, '$.chunk_id' returning number) as chunk_id,
   json_value(t.column_value, '$.chunk_length' returning number) as chunk_size,
   json_value(t.column_value, '$.chunk_data') as chunk_data
from webpages_trimmed w,
dbms_vector_chain.utl_to_chunks(
   dbms_vector_chain.utl_to_text(w.content),
   json('{"normalize":"all"}')
) t
where w.id = 1173
order by 3;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY

Fetched 116 rows in 0.146 seconds

	ID	URL	CHUNK_ID	CHUNK_SIZE	CHUNK_DATA
1	1173	https://oracle-base.com/articles/mis…	1	349	Pipelined Table Functions Table Functions Pipelined Table Functions NO_DATA_NEEDED Ex
2	1173	https://oracle-base.com/articles/mis…	2	566	Table functions are used to return PL/SQL collections that mimic tables. They can be que
3	1173	https://oracle-base.com/articles/mis…	3	307	named row and table types to be created as database objects. -- Create the types to sup
4	1173	https://oracle-base.com/articles/mis…	4	284	-- Build the table function itself.CREATE OR REPLACE FUNCTION get_tab_tf (p_rows IN NUME
5	1173	https://oracle-base.com/articles/mis…	5	108	RETURN l_tab;END;/-- Test it.SELECT *FROM TABLE(get_tab_tf(10))ORDER BY id DESC;
6	1173	https://oracle-base.com/articles/mis…	6	359	ID DESCRIPTION---------- --10 Descriptio
7	1173	https://oracle-base.com/articles/mis…	7	313	1 Description for 110 rows selected.SQL> Notice the above output is in reverse order be
8	1173	https://oracle-base.com/articles/mis…	8	359	ID DESCRIPTION---------- --10 Descriptio

Chunking? Yes

DBMS_VECTOR_CHAIN.UTL_TO_CHUNKS has many parameters (all defined in a single JSON parameter).

In theory it can do everything you need.

In practice, does it do what you really want and need?

Partially yes,

fully probably no…

PARAMS

Specify the input parameters in JSON format.

```
{
    "by”            :     mode,
    "max”           :     max,
    "overlap"       :     overlap,
    "split"         :     split_condition,
    "custom_list"   :     [ split_chars1, ... ],
    "vocabulary"    :     vocabulary_name,
    "language"      :     nls_language,
    "normalize"     :     normalize_mode,
    "norm_options"  :     [ normalize_option1, ... ],
    "extended"      :     boolean
}
```

Copy

Embedding? Yes

DBMS_VECTOR_CHAIN
has you covered (again)

The “PARAMS” parameter allows you to say “what” does the embedding and how.

211.1.8 UTL_TO_EMBEDDING and UTL_TO_EMBEDDINGS

Use the `DBMS_VECTOR_CHAIN.UTL_TO_EMBEDDING` and `DBMS_VECTOR_CHAIN.UTL_TO_EMBEDDINGS` chainable utility functions to convert plain text to one or more vector embeddings.

Purpose

To perform a text to embedding transformation by accessing:

- Oracle Database as the service provider: Calls the pretrained ONNX format embedding model that you have loaded into the database (default setting)
- Third-party embedding model: Makes a REST call to your chosen third-party service provider, such as Cohere, Google AI, Hugging Face, Oracle Cloud Infrastructure (OCI) Generative AI, OpenAI, or Vertex AI

Syntax

```
DBMS_VECTOR_CHAIN.UTL_TO_EMBEDDING (
    DATA            IN CLOB,
    PARAMS          IN JSON default NULL
) return VECTOR;
```

Copy

```
DBMS_VECTOR_CHAIN.UTL_TO_EMBEDDINGS (
    DATA            IN VECTOR_ARRAY_T,
    PARAMS          IN JSON default NULL
) return VECTOR_ARRAY_T;
```

Copy

DATA

`UTL_TO_EMBEDDING` converts text (`CLOB`) to a single embedding (`VECTOR`).

`UTL_TO_EMBEDDINGS` convert an array of chunks (`VECTOR_ARRAY_T`) to an array of embeddings (`VECTOR_ARRAY_T`).

All together? Yes

All these steps can be chained together.

The result can be inserted in a table, giving you the chunks and the vectors in a single step.

```sql
select
    w.id, w.url, et.embed_id chunk_id, et.embed_data chunk_data,
    to_vector(et.embed_vector) chunk_embedding
from webpages_trimmed w,
dbms_vector_chain.utl_to_embeddings(
    dbms_vector_chain.utl_to_chunks(
        dbms_vector_chain.utl_to_text(w.content),
        json('{"normalize": "all"}')
    ),
    json('{"provider": "OCIGenAI", "credential_name": "CRED_OCI_GENAI", "url": "https://inference.generativeai.e
) t,
JSON_TABLE(t.column_value, '$[*]'
    COLUMNS (embed_id NUMBER PATH '$.embed_id',
             embed_data VARCHAR2(4000) PATH '$.embed_data',
             embed_vector CLOB PATH '$.embed_vector')
) et
where w.id = 1173;
```

OracleBaseGPT: demo

OracleBaseGPT (demo)

OracleBaseGPT

Ask Tim questions...

This is a RAG demo based on 1327 articles published on oracle-base.com (up to 1st July 2024). Oracle Database 23ai is used to store and search vectors. Oracle OCI GenAI is used for embedding, text generation and LLM usage in general.

- **embedding:** model `cohere.embed-english-v3.0` in the Germany Central (Frankfurt) region
- **LLM:** `meta.llama-3-70b-instruct v1.0` in the Germany Central (Frankfurt) region

✅ Connected to Oracle Database 23ai [v23.6.0.24.7]

What is up?

OracleBaseGPT (demo)

OracleBaseGPT

Ask Tim questions...

This is a RAG demo based on 1327 articles published on oracle-base.com (up to 1st July 2024).
Oracle Database 23ai is used to store and search vectors. Oracle OCI GenAI is used for embedding, text generation and LLM usage in general.

- **embedding:** model `cohere.embed-english-v3.0` in the Germany Central (Frankfurt) region
- **LLM:** `meta.llama-3-70b-instruct v1.0` in the Germany Central (Frankfurt) region

✅ Connected to Oracle Database 23ai [v23.6.0.24.7]

What are pipelined table functions?

OracleBaseGPT: Pipelined table functions are a type of table function that allows rows to be piped out of the function as they are created, saving memory and enabling subsequent processing to start before all rows are generated. They include the PIPELINED clause and use the PIPE ROW call to push rows out of the function. This approach negates the need to build huge collections, making it particularly useful for large warehousing ETL operations.

Sources:

What is up?

OracleBaseGPT (demo)

generation and LLM usage in general.

- **embedding:** model `cohere.embed-english-v3.0` in the Germany Central (Frankfurt) region
- **LLM:** `meta.llama-3-70b-instruct v1.0` in the Germany Central (Frankfurt) region

✅ Connected to Oracle Database 23ai [v23.6.0.24.7]

What are pipelined table functions?

OracleBaseGPT: Pipelined table functions are a type of table function that allows rows to be piped out of the function as they are created, saving memory and enabling subsequent processing to start before all rows are generated. They include the PIPELINED clause and use the PIPE ROW call to push rows out of the function. This approach negates the need to build huge collections, making it particularly useful for large warehousing ETL operations.

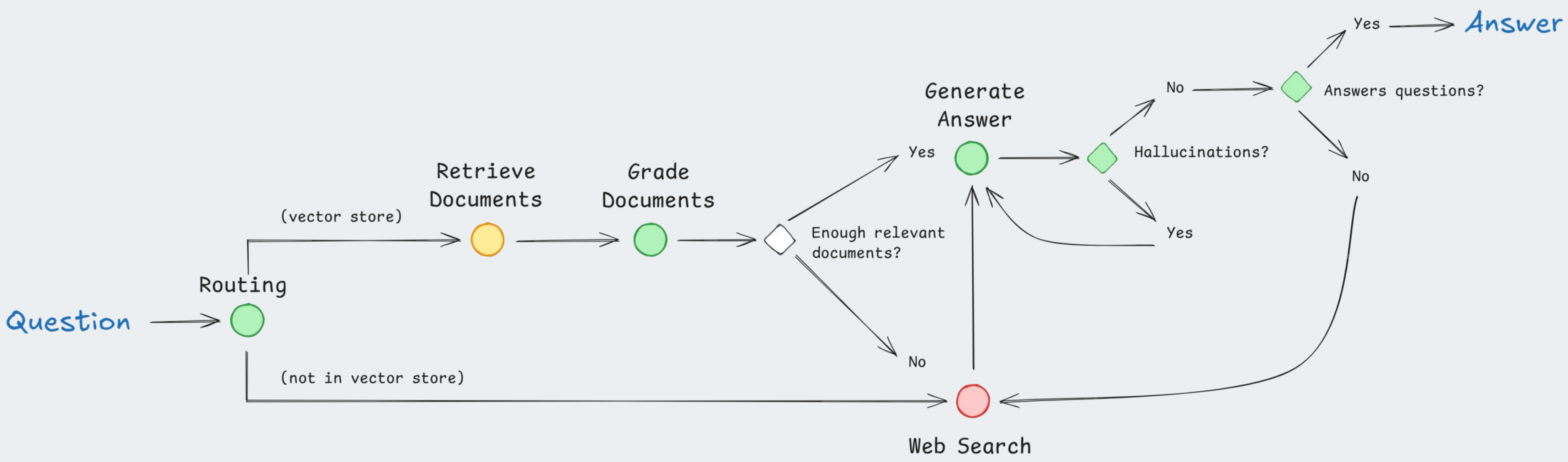
Sources:

https://oracle-base.com/articles/misc/pipelined-table-functions
https://oracle-base.com/articles/12c/plsql-to-sql-interface-enhancements-for-plsql-only-data-types-12cr1
https://oracle-base.com/articles/misc/list-files-in-a-directory-from-plsql-and-sql-dbms-scheduler
https://oracle-base.com/articles/misc/sql-for-beginners-the-from-clause

Execution log:

What is up?

OracleBaseGPT (demo)



OracleBaseGPT (demo)

Execution log:

```
[ INFO     ] Successfully connected to Oracle Database (23.6.0.24.7)
[ DEBUG    ] ---ROUTE QUESTION---
[ DEBUG    ] question: What are pipelined table functions?
[ DEBUG    ] route: vectorstore
[ DEBUG    ] ---ROUTE QUESTION TO RAG---
[ DEBUG    ] ---RETRIEVE---
[ DEBUG    ] Finished running: retrieve
[ DEBUG    ] ---CHECK DOCUMENT RELEVANCE TO QUESTION---
[ DEBUG    ] ---GRADE: DOCUMENT RELEVANT---
[ DEBUG    ] ---GRADE: DOCUMENT RELEVANT---
[ DEBUG    ] ---GRADE: DOCUMENT RELEVANT---
[ DEBUG    ] ---GRADE: DOCUMENT RELEVANT---
[ DEBUG    ] ---GRADE: DOCUMENT RELEVANT---
[ DEBUG    ] ---GRADE: DOCUMENT RELEVANT---
[ DEBUG    ] ---GRADE: DOCUMENT RELEVANT---
[ DEBUG    ] ---GRADE: DOCUMENT NOT RELEVANT---
[ DEBUG    ] ---GRADE: DOCUMENT RELEVANT---
[ DEBUG    ] ---GRADE: DOCUMENT RELEVANT---
[ DEBUG    ] ---DOCUMENT RELEVANCE: 9 of 10---
[ DEBUG    ] ---ASSESS GRADED DOCUMENTS---
[ DEBUG    ] ---DECISION: GENERATE---
[ DEBUG    ] Finished running: grade_documents
[ DEBUG    ] ---GENERATE---
[ DEBUG    ] ---CHECK HALLUCINATIONS---
[ DEBUG    ] ---DECISION: GENERATION IS GROUNDED IN DOCUMENTS---
[ DEBUG    ] ---GRADE GENERATION vs QUESTION---
[ DEBUG    ] ---DECISION: GENERATION ADDRESSES QUESTION---
[ DEBUG    ] Finished running: generate
[ DEBUG    ]
[ DEBUG    ] answer:
[ DEBUG    ] Pipelined table functions are a type of table function that allows
```

What is up?

Building a custom ChatGPT-like chatbot

Oracle Database 23ai provides mostly storage for vectors and distance queries. Depending on your workload, embedding is better executed outside (you maybe need your database resources for queries and not generating vectors).

To generate a text with a LLM you can't do it fully internally in the database for now.

OCI GenAI is very easy to use and doesn't require any commitment (reserving hardware or resources).

But keep in mind you don't own the models, and Oracle do release new ones from time to time, and then remove the old ones.

- You should always perform embeddings with the same model (except if the model explicitly says it is 1:1 compatible with another one, generally not the case)

Basic RAG doesn't bring much to the table, building more advances agents powered by LLM and retrievals has lot of potential.

LLMs starts being able to use tools and perform a step by step logical thinking.