Context Engineering in Practice

From documents to answers with VectorDBs (Weaviate, Pinecone)

Anna Saranti

RINGANA - SoFresh IT Solutions GmbH
Vienna AI Engineering Meetup

2026.01.27

1. Text Chunking
2. Vector DBs
3. Query & result pre- and post-processing
4. Evaluation
5. Context Engineering

What are VectorDBs? (1/2)

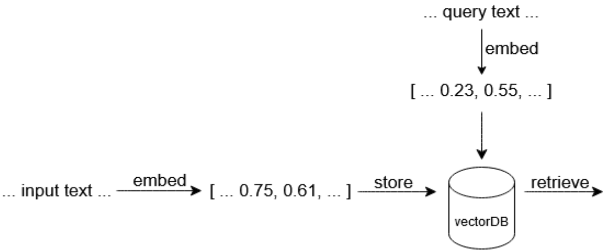
- Stores textual data (and images...)
  as number vectors **(embeddings)**
- **Not** designed to answer **concrete** questions,
  Not designed for **exact** match [2], f.e.:
  - Which customers bought product X in 2026?
  (SQL query)

  but rather handle use-cases like:
- My R&D department has a set of documents in different languages/countries containing information about ingredients and rules.
  What rule applies in country Y for ingredient W?

  $\implies$ you can't compare row-by-row!

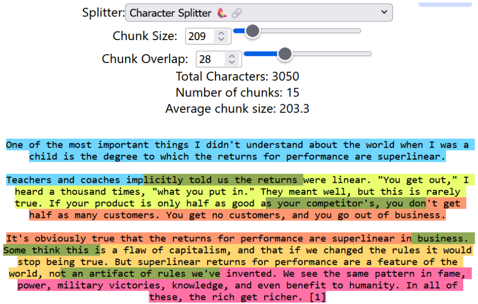
What are VectorDBs? (2/2)

- General pipeline for VectorDBs:
  1. embed
  2. store
  3. retrieve



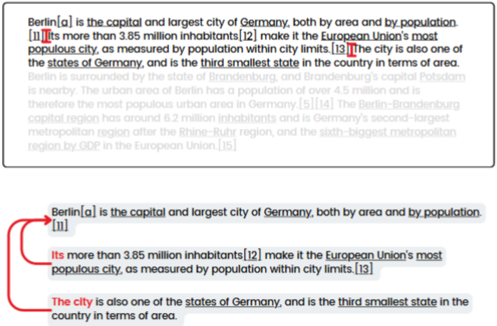
- What are the criteria?
  1. is the retrieved text **relevant**?
  2. what is the **latency** of retrieval?

Text Chunking (1/4)



- fixed-length, token-, character-based splitting, semantic, hierarchical [3], [4] (logical boundaries, sentences, paragraphs)
- 🌐 `https://chunkviz.up.railway.app/`
- `langchain_text_splitters.RecursiveCharacterTextSplitter` hierarchical split by separators until chunk meets size constraints

Text Chunking (2/4)



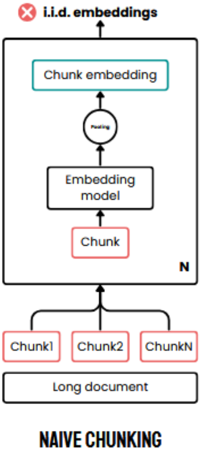
Michael Günther, Isabelle Mohr, Daniel James Williams, Bo Wang, and Han Xiao
Late chunking: contextual chunk embeddings using long-context embedding models
arXiv preprint arXiv:2409.04701, 2024.

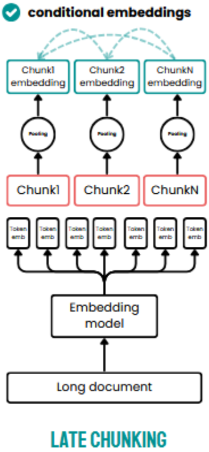
Late Chunking [1]: Capture **long-distance** semantic dependencies

🌐 `https://github.com/jina-ai/late-chunking`

🌐 `https://weaviate.io/blog/late-chunking`

Text Chunking (3/4)





Michael Günther, Isabelle Mohr, Daniel James Williams, Bo Wang, and Han Xiao
Late chunking: contextual chunk embeddings using long-context embedding models
arXiv preprint arXiv:2409.04701, 2024.

Text Chunking (4/4)

Simple example:

```
In order to become a partner, and you live in Spain or Mexico, you need to send an email to hr_ringana
Similarities: Contextual Retrieval: 0.8283 | Late Chunking: 0.8550

For those who live in Germany, you just need to send a thumbs-up
Similarities: Contextual Retrieval: 0.7517 | Late Chunking: 0.8193
```

Further resources:

- The 5 Levels Of Text Splitting For Retrieval: https://www.youtube.com/watch?v=8OJC21T2SL4
- **Agentic** Chunking
  An Agent analyses the document's structure and content to select the best chunking - no fixed rules [18]

https://weaviate.io/blog/chunking-strategies-for-rag

Table and Image extraction (2/2)

Saranti, Anna, et al. *From 3D pointcloud data to explainable geometric deep learning: State of the art and future challenges* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 14.6 (2024): e1554.

Image and table extraction with 🌐 `https://unstructured.io/`

Table and Image extraction (2/2)

- https://doclink.io/
- https://landing.ai/ NEW
  https://www.deeplearning.ai/short-courses/document-ai-from-ocr-to-agentic-doc-extraction
- Tables extracted as *.csv* file
  if too large → process row-by-row
  table's number and page "sufficiently" good for now

Different VectorDBs - How to choose?

VectorDB comparison: [2], [6]

- **FAISS:** For real-time recommendation engine
- **Milvus:** GPU-accelerated indexing -
  Good for billions of data
  (scalability and distributed environments)
- **Chroma:** Rapid experimentation before scaling
  f.e. hackathons
- **Weaviate:** Semantic search and graph-based indexing
  real-time recommendations/chatbots
- **Pinecone:** Automatic scaling of resources
  without managing infrastructure

Indexing for semantic search (1/4)

Index of nearest vector(s)? - implies a **proximity metric**
cosine similarity - norm necessary or not ? ☺

- K-nearest-neighbour (k-NN) [7]
- Hierarchical Navigable Small World (HNSW) graphs [8]
- Invertible File Indexing (IVF)
  partition the data into clusters

$\Longrightarrow$ approximation of nearest-neighbour (ANN)
reduce $\downarrow$ the nr. of comparisons

Indexing for semantic search (2/4)

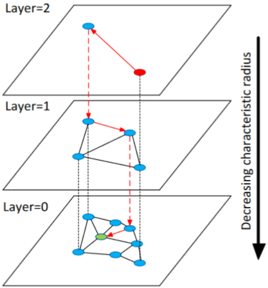


Fig. 1. Illustration of the Hierarchical NSW idea. The search starts from an element from the top layer (shown red). Red arrows show direction of the greedy algorithm from the entry point to the query (shown green).

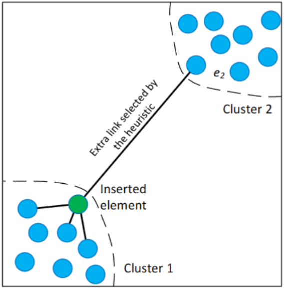


Fig. 2. Illustration of the heuristic used to select the graph neighbors for two isolated clusters. A new element is inserted on the boundary of Cluster 1. All of the closest neighbors of the element belong to the Cluster 1, thus missing the edges of Delaunay graph between the clusters. The heuristic, however, selects element e_2 from Cluster 2, thus, maintaining the global connectivity in case the inserted element is the closest to e_2 compared to any other element from Cluster 1.

Yu A. Malkov and Dmitry A. Yashunin, *Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs.*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(4):824–836, 2018

Indexing for semantic search (3/4)

Recall metric [2] for any ANN search:

$\{\text{Recall@k}\} = \frac{m}{k}$

- m: neighbours returned by an ANN search
- k: **true** nearest neighbours

"Perfect" recall == 1 $\iff \mathcal{O}(n \cdot d)$ for n d-dim vectors
Slow for large datasets

vs.

HNSW $\xrightarrow[\text{to}]{\text{close}} \mathcal{O}(log\ n)$
but consumes more memory because of the graph

Indexing for semantic search (4/4)

New data - new vectors added, others deleted,
what is the strategy?

1. IVF supports adding vectors without reindexing
2. Schedule retraining/rebuild the index
   at low-traffic periods
   maintain a "shadow" index
   queries should not be interrupted during rebuilds
3. Index merging at distributed systems
4. Versioning is the snapshotting of an index

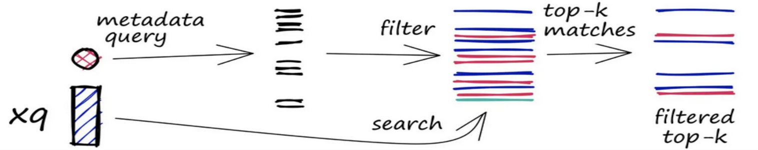
Takeaways of semantic search

1. Cosine similarity **or** dot product?
   does the norm contain information?
2. Choose the distance metric that matches the model's training (objective)
   If unknown, use A/B on labelled set [5]
3. Poor chunking cannot be compensated [5]
4. The pipeline indicates the improvement steps:
   - **First** improve retrieval (facts)
   - **Second** change the prompt
   - **Lastly** fine-tune if you see repeated failures that cannot be fixed with the above steps

Weaviate (1/13)

- HNSW for semantic search - set parameters to adjust memory, index build time, query time, and recall
- Keyword-based search (engine) too! -
  Best Matching 25 (BM25) [9]
  (think of it like TF-IDF for now) - cheap!

  Filter first, then do semantic search
  better than imposing constraints in the prompt
  “don’t chase ghosts with prompts” [5]



https://www.pinecone.io/learn/vector-search-filtering/

Weaviate (2/13)

Schema definition:

- Classes with properties/fields
- Relations for relational queries
  cross-reference-edges modelled through graph relationships
- Vector index → HNSW → ANN
- Graph layer → relational queries

Weaviate (3/13)

- Metadata filtering **before** vector similarity
- Vector-based semantic search $\leftrightarrow$ dense vectors
- Keyword-based search (BM25) $\leftrightarrow$ sparse vectors
  search for specific terms (exact match)

- Hybrid search is a weighted combination:

$$\begin{aligned}\text{score} = \alpha \cdot \text{vector\_search\_score} + \\ (1-\alpha) \cdot \text{BM25\_search\_score}\end{aligned}$$

$\alpha \in [0, 1)$: $\alpha = 0.55$ (Weaviate) or $\alpha = 0.7$ (books)

  - adjust w.r.t. evaluation metrics (section 4)
  - dynamically adjusted with Reinforcement Learning (RL)

Weaviate (4/13)

```python
client.collections.create(
    name=parent_document_name,
    vector_config=vectorizer,
    properties=[
        # [A.01.] Title of the document (inside the document) ---------------------------------------------------
        Property(name="title", data_type=DataType.TEXT, description="Title of Document (inside the Document)",),
        # [A.02.] Source URL - document path/name ------------------------------------------------------------------
        Property(name="source_url", data_type=DataType.TEXT, description="Document's source URL - document path/name",),
        # [A.03.] Datetime of "publishing" ------------------------------------------------------------------------
        #         Expects a string of RFC 3339-formatted timestamps (basically strict ISO 8601) -------------------
        #         f.e. "2025-11-08T04:46:23Z" OR "2025-11-08T04:46:23+00:00" --------------------------------------
        Property(name="published_at", data_type=DataType.DATE, description="Document's Datetime of publishing",),
        # [A.04.] Versioning (from our side). Document(s), Chunk(s) and DomainEntitie(s) -------------------------
        #         that belong to the same version, are coherent -------------------------------------------------
        Property(name="version", data_type=DataType.TEXT, description="Document's version", ),
        # [A.05.] From when this document is valid/applies (Optional) - used for filtering -----------------------
        Property(name="valid_from", data_type=DataType.DATE, description="From this datetime this document is valid/applies"
        # [A.06.] Until when this document is valid/applies (Optional) - used for filtering ----------------------
        Property(name="valid_to", data_type=DataType.DATE, description="From this datetime this document is valid/applies",)
        # [A.07.] What are the languages inside the document -----------------------------------------------------
        Property(name="languages", data_type=DataType.TEXT_ARRAY, description="List of languages inside the Document",),
        # [A.08.] Summary of the whole document -------------------------------------------------------------------
        Property(name="summary", data_type=DataType.TEXT, description="Summary of the Document",),
        # [A.09.] Topics addressed in the Document ----------------------------------------------------------------
        Property(name="topics", data_type=DataType.TEXT_ARRAY, description="The topics addressed in the content of the Docume
    ],
)
```

Weaviate (5/13)

```python
# [D.1.] The references that describe the hierarchy ------------------------------
#        Documents are parents --------------------------------------------------
#        Chunks are children ----------------------------------------------------
parent_documents = client.collections.get(parent_document_name)
parent_documents.config.add_reference(
    ReferenceProperty(name="has_chunks", target_collection=child_chunk_name),
)

child_chunks = client.collections.get(child_chunk_name)
child_chunks.config.add_reference(
    ReferenceProperty(name="of_document", target_collection=parent_document_name)
)

# [D.2.] The references that support the versioning of Documents -----------------
#        Old Documents are "superseded_by" a newer version ------------------------
parent_documents.config.add_reference(
    ReferenceProperty(name="superseded_by", target_collection=parent_document_name)
)
# Newer Documents "supersedes" the older version ---------------------------------
parent_documents.config.add_reference(
    ReferenceProperty(name="supersedes", target_collection=parent_document_name)
)
```

Weaviate (6/13)

```
graphql_parent_document_inspection_query = """
    {
      Get {
        ParentDocument {
          source_url
          title
          valid_to
          superseded_by {
            ... on ParentDocument {
              title
            }
          }
          supersedes {
            ... on ParentDocument {
              title
            }
          }
          has_chunks {
            ... on ChildChunk {
              title
            }
          }
        }
      }
    }
```

⟹

```
get={'ParentDocument': [{'has_chunks': None,
  V1                     'source_url': '[redacted] Mails '
                                       '[redacted]_text.txt',
                         'superseded_by': [{'title': [redacted]
                                                     'Mails [redacted]'
                                                     [redacted]}],
                         'supersedes': None,
                         'title': '[redacted]Mails [redacted] '
                                  '[redacted] '
                                  '[redacted]'},
                        {'has_chunks': None,
  V2                     'source_url': '[redacted]Mails '
                                       '[redacted]_V2_20251110_text.txt',
                         'superseded_by': None,
                         'supersedes': [{'title': '[redacted]'
                                                  'Mails [redacted]'
                                                  '[redacted]'
                                                  [redacted]}],
                         'title': '[redacted]Mails [redacted]'
                                  '[redacted] '
```

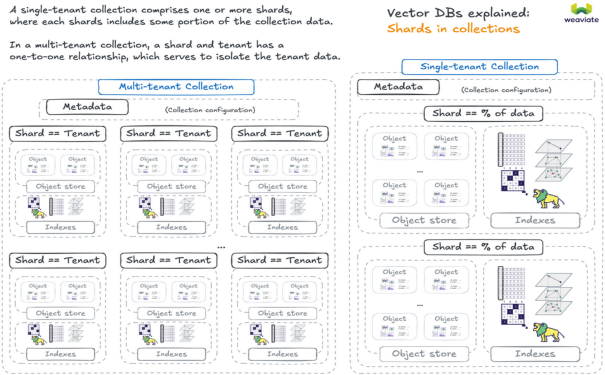
Queries with GraphQL 🌐 https://graphql.org/

Weaviate (7/13)

Can we have different document sets for different user types?

- **tenant**: is a logical partition within a class
  has its own vector index and metadata
  for different user groups → supports **jurisdiction**
- Use true multi-tenancy and not a `tenant_id` property/field
  "soft" multi-tenancy means that all tenants will share the same HNSW graph [5]
  → they will be affected by each other (vectors co-mingle)

Weaviate (8/13)



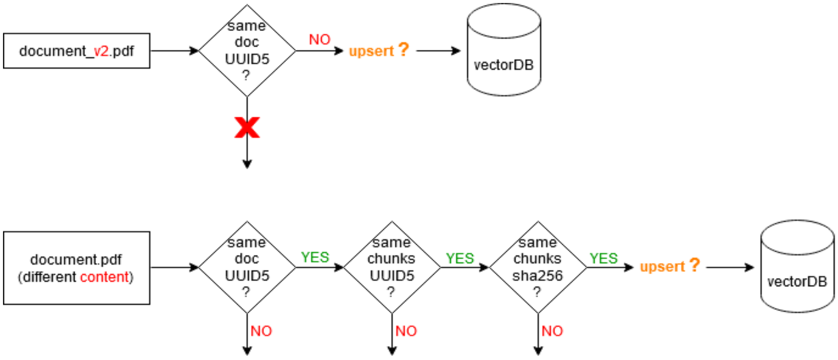
🌐 https://docs.weaviate.io/weaviate/concepts/cluster

- **Shards**: splits with disjoint subsets of object
  isolation supports security
  build and query at the same time ☺

Weaviate (9/13)

- `UUID5` for upserts and revised documents [5]

  Update: references `supersedes`, `superseded_by`



Weaviate (10/13)

- What about **delete**? -
  with the General Data Protection Regulation (GDPR) in mind
  retention period $3 - 6$ months [5]
- **hard** $\overset{\text{vs.}}{\leftrightarrow}$ **soft** delete ?

  soft delete:

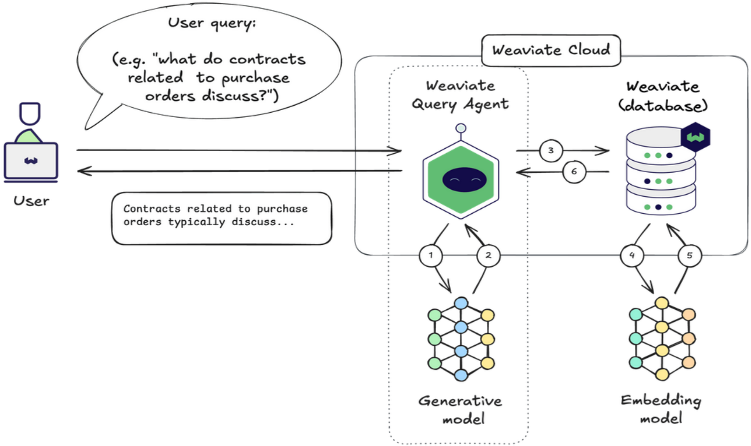
  { `state="invalid", valid_to="current datetime"` }
- References can maintain the connection
  to the (soft) deleted documents -
  Queries should only address { `state="valid"` }

Weaviate (11/13)

- Text and images vectorised into a shared space using models like Contrastive Language–Image Pretraining (CLIP) [11]
- Support for **multi-modal** queries: `"find images similar to this text"` [6] and vice-verca

```
query_image = {
    "query": """
    {
        ... nearImage: { ... image: "%s", ... }
    }
    """ % encode_image("red_shirt.jpg")
}
```

Weaviate (12/13)



https://docs.weaviate.io/agents/query

Weaviate (13/13)

Weaviates' **Query Agent**:

1. takes a user's prompt/question in natural language
2. determines which part of your VectorDB to query
3. decides the best way to structure the query
4. evaluates the search results and returns the answer

Weaviates' **Transformation Agent**:

- appends new/updates existing properties

Pinecone (1/2)

```
# [1.] Define filter(s) ------------------------------------
filters_dict = {
    "state": {"$eq": "active"},
    "languages": {"$in": ["Greek"]},
}

# [2.] Search in the dense_index ---------------------------
dense_index = pinecone_client.Index(index_name)
max_documents_retriever_k = config.rag.max_documents_retriever_k

results = dense_index.search(
    namespace=namespace_name,
    query={
        "top_k": max_documents_retriever_k,
        "inputs": {"text": user_query},
        "filter": filters_dict,
    },
)
```

- Serverless index and autoscaling (automates manual tuning of indexes)
- **namespace** $\leftrightarrow$ tenants
- (Hybrid) queries as in Weaviate

Pinecone (2/2)

```
# [1.] Dictionary with all metadata ----------------------------
metadata_dict = {
    "state": state,
    "of_document": of_document,
    "has_chunks": has_chunks,
}

# [2.] Pinecone common record, with ID and the body of the chunk
#      the body is indexed for the queries ---------------------
#      the rest is helpful metadata ----------------------------
self._pinecone_common_record = {
    "id": uuid_5,
    "body": body,
    "metadata": metadata_dict,
}
```

- Relational queries not so straightforward as in Weaviate they are expressed as fields **!**

Other Vector DB-related topics (1/2)

- Service-level agreements (SLA):
  Guarantees to meet service 1) performance, 2)availability, 3) reliability targets
- Logging for GDPR (and co.)
- Product Quantisation (PQ) and Scalar Quantisation (SQ)
  Quantise only after a clean baseline
  on all metrics [5]

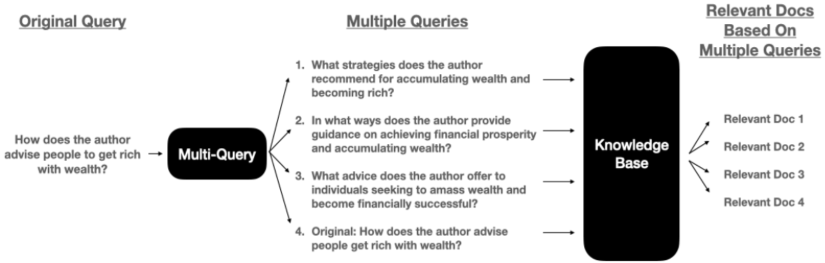
Other Vector DB-related topics (2/2)

Security:

- Personally Identifiable Information (PII) can be embedded!
  Adversarial attacks that reconstruct PII from the embeddings
  1) detect PII, 2) mask or remove before embedding
- Differential Privacy (DP) -
  adds noise and reduces ↓ retrieval accuracy
- Homomorphic encryption are expected to be integrated by Weaviate [6]
- Keep only essential metadata to reduce ↓ exposure
- Role-based Access Control (RBAC) [5]:
  restrictions based on user roles
  Log operations: who, what, when, how

1. Text Chunking
2. Vector DBs
3. **Query & result pre- and post-processing**
4. Evaluation
5. Context Engineering

Multi-Query (1/2)



🌐 https://fullstackretrieval.com/

More techniques like contextual compression,
like a further "filtering" step with an LLM and provided context
supported by: 🌐 https://www.langchain.com/

Multi-Query (2/2)

```
prompt_template = """You are an AI language model assistant.
Your task is to generate 10 different versions of the given
user question to retrieve relevant documents from a vector
database. As a first step, use synonyms of the main keywords,
objects, adjectives and verbs in the question. As a second
step, try to figure out what further topics the user posing
the question might be interested in, that are not contained
explicitly in the original question.As a third step, try to
predict what the next question the user might ask after the
original question so that the retrieved documents can cover
that as well. In a fourth step, try to generalize the
question. By generating multiple perspectives on the user
question, your goal is to help the user overcome some of the
limitations  of distance-based similarity search.
Provide these alternative questions separated by newlines.
Original question: {question}"""
```

Reranking, multi-hop queries, ...

- Rerank the (k) results, particularly those that are semantically close (share vocabulary)
  with a cross-encoder: (query, context) $\rightarrow$ joint score

  uses Maximum Marginal Relevance (MMR) to help diversify the retrieved top-k:

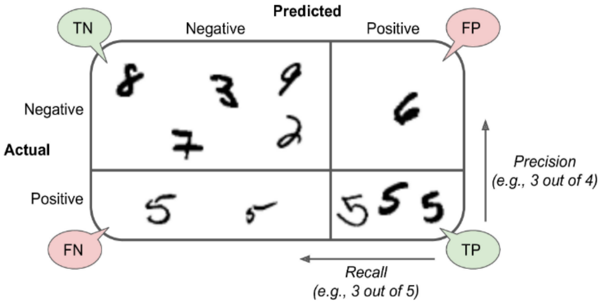
  1. Start with the best-scoring chunk
  2. Repeat
     add the candidate that is relevant to the query
     but least similar to what you already selected.

- Multi-hop query: result of second (2nd) query depends on result of the first (1st)

p90/p95, latency and other metrics

- **p95**: response time below which 95% of requests complete only 5% are lower than $<$ **p95**
- Clickable citation support (supported by properties/fields like `source_url`)
- If the returned context is not "sufficiently" long or the metrics are not "sufficiently" good
  1. ask clarifying question
  2. escalate to human
  3. open a ticket with retrieved passages attached
- Use small "golden/representative" set
  Run evaluations on big datasets during the night

Retrieval Augmented Generation Assessment Suite - RAGAS (1/4)



Aurélien Géron *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly Media, 2019

- Confusion matrix for 2-class classification: [10]
  "ground-truth" class ↔ **predicted** class
- Retrieval-Augmented Generation (RAG):
  "ground-truth" context ↔ **retrieved** context
  "correct" answer ↔ **actual** answer

Retrieval Augmented Generation Assessment Suite - RAGAS (2/4)

- What is compared?
  - relevant vs. irrelevant contexts (implied here: chunks)
  - answer w.r.t. all fetched contexts - whether relevant/irrelevant
  - user's question and answer alone - without contexts
- A plethora of metrics, f.e.

  Faithfulness: Is the returned answer in accordance with the "ground-truth"?
  ("Ground-truth" verifiable? with DB queries?)
  Are the hallucinations "minimal"?

- RAGAS uses an LLM-as-a-Judge system
- You can customise your own metrics! ☺

🌐 `https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/`

Retrieval Augmented Generation Assessment Suite - RAGAS (3/4)

```
from ragas.metrics import (
 # [A.] RAG ==========================================
 # [RAG - 1.] Context Precision with and without an LLM
 LLMContextPrecisionWithoutReference, NonLLMContextPrecisionWithReference,
 # [RAG - 2.] Context Recall with and without an LLM
 LLMContextRecall, NonLLMContextRecall,
 # [RAG - 3.] ContextEntityRecall - split the retrieved context (chunks) in entities
 ContextEntityRecall,
 # [RAG - 4.] Noise Sensitivity
 NoiseSensitivity,
 # [RAG -5.] ResponseRelevancy
 ResponseRelevancy,
 # [RAG - 6.] Faithfulness and FaithfulnesswithHHEM
 Faithfulness, FaithfulnesswithHHEM,
 # [B.] NVIDIA ========================================
 AnswerAccuracy, ContextRelevance, ResponseGroundedness,
 # [C.] Natural Language Comparison (NLC) =============
 # [C.1.] LLM NLC metrics (with an LLM)
 FactualCorrectness, SemanticSimilarity,
 # [C.2.] Non-LLM NLC metrics (without an LLM)
 BleuScore, RougeScore, ExactMatch, StringPresence,
 # [D.] Other metrics (not included in the documentation)
 AspectCritic, SimpleCriteriaScore, RubricsScore,
)
```

Retrieval Augmented Generation Assessment Suite - RAGAS (4/4)

Actionable directives [5]:

- Faithfulness $\downarrow$ low,
  but the retrieved context contains the fact
  $\implies$ improve the prompt to enforce citations
- Context recall $\downarrow$ low,
  indicates retrieval issues
  $\implies$ add filters, adjust alpha, chunking, use synonyms
- Both recall and faithfulness $\uparrow$ high,
  but relevance $\downarrow$ low
  $\implies$ rewrite the query
- The retrieved context seems right,
  but the ground truth differs
  $\implies$ source documents are wrong/outdated

Context Engineering (1/11)

Context injection: retrieving documents,
then "feeding" to the LLM

Generation: the LLM generates the response $\xleftarrow[\text{on}]{\text{based}}$
the **query** and the **context**

in the prompt: `"answer the question based on the provided context"`

"The model treats context as **evidence**
not as **instructions**" [5]

Grounded (only from context) $\neq$ **Guardrail**

Context Engineering (2/11)

- Agents as **architects** and **users** of the context [18] that enables **adaptive** strategy
- Goals: context summarisation/pruning/offloading, dynamic tool selection, adaptive retrieval
- Avoid: context poisoning, distraction, confusion, clash [18]

  $\Longrightarrow$ ***"tail architecture to the task"***

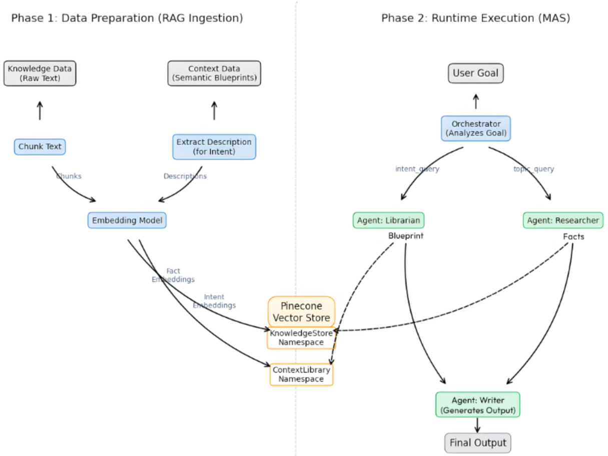
- ***How*** to orchestrate the context to achieve the goals ?
  How to have planning with tool discovery and detection, action and observation?

Context Engineering (3/11)

- **Input:** user's high-level goal
  How can a Vector DB help to translate this
  into an **adaptive** plan?
- **Output:** complex, multi-step process automatically managed
- Structured message Model Context Protocol (MCP)-like
  to pass the engineered context between the agents
  (not only for agent-tool communication)
  containing **tasks** and **results**

  `https://modelcontextprotocol.io/docs/getting-started/intro`

Context Engineering (4/11)



Denis Rothman *Context Engineering for Multi-Agent Systems: Move beyond prompting to build a Context Engine, a transparent architecture of context and reasoning* Packt Publishing, 2025.

- Dual-RAG Multi-Agent System (MAS) [18]

Context Engineering (5/11)

```
context_blueprints = [
    {
        "id": "blueprint_suspense_narrative",
        "description": "A precise Semantic Blueprint designed to generate
suspenseful and tense narratives, suitable for children's stories. Focuses on
atmosphere, perceived threats, and emotional impact. Ideal for creative writing.",
        "blueprint": json.dumps({
            "scene_goal": "Increase tension and create suspense.",
            "style_guide":
"Use short, sharp sentences. Focus on sensory details (sounds, shadows). Maintain
a slightly eerie but age-appropriate tone.",
            "participants": [
              { "role": "Agent", "description": "The protagonist experiencing
the events." },
              { "role": "Source_of_Threat", "description": "The underlying
danger or mystery." }
            ],
          "instruction": "Rewrite the provided facts into a narrative adhering
strictly to the scene_goal and style_guide."
          })
    },
```

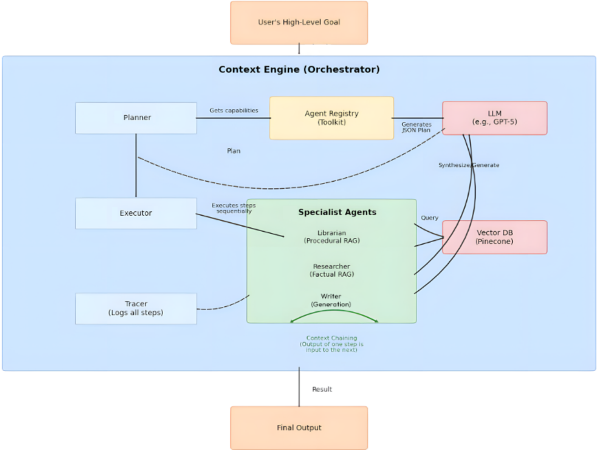
Denis Rothman *Context Engineering for Multi-Agent Systems: Move beyond prompting to build a Context Engine, a transparent architecture of context and reasoning* Packt Publishing, 2025.

Context Engineering (6/11)

- Store and retrieve **procedural instructions** from a VectorDB to implement a procedural/dynamic RAG
- Two (2) different namespaces
- Create an **Agent Registry** to be used by a **Planner**

  The **Agent Registry** has a list of available capabilities for the LLM called by the **Planner** to create a dynamic, step-by-step execution plan
- **Executor** executes the plan
- **Tracer** logs all steps (what & why)

Context Engineering (7/11)



Denis Rothman *Context Engineering for Multi-Agent Systems: Move beyond prompting to build a Context Engine, a transparent architecture of context and reasoning* Packt Publishing, 2025.

Context Engineering (8/11)

- `get_capabilities_description()`:
  returns $\forall$ agent
  - purpose
  - inputs
  - outputs
- **Librarian** agent:
  semantically search the Context Library namespace trying to find the match to the user's intent
- **Context Chaining**:
  output of one agent $\xrightarrow{\text{becomes}}$ input to the next agent

Context Engineering (9/11)

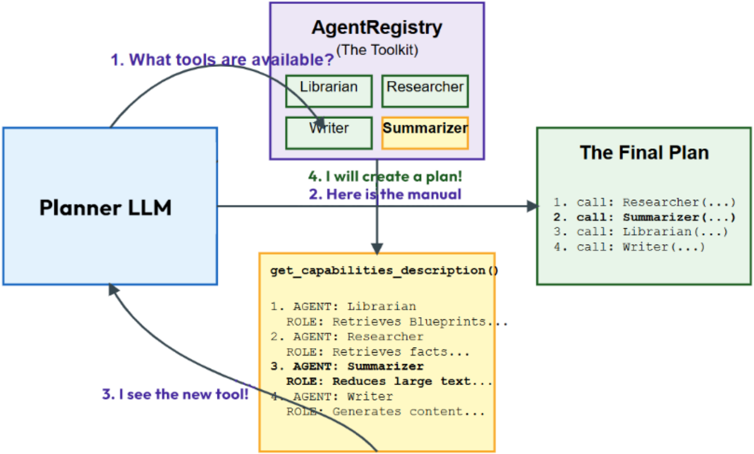
```
def agent_context_librarian(mcp_message):
    """
    Retrieves the appropriate Semantic Blueprint from the Context Library.
    """
    print("\\n[Librarian] Activated. Analyzing intent...")
    requested_intent = mcp_message['content']['intent_query']
    results = query_pinecone(requested_intent, NAMESPACE_CONTEXT, top_k=1)

    if results:
        match = results[0]
        print(f"[Librarian] Found blueprint '{match['id']}' (Score:
{match['score']:.2f})")
        blueprint_json = match['metadata']['blueprint_json']
        content = {"blueprint": blueprint_json}
    else:
        print("[Librarian] No specific blueprint found. Returning default.")
        content = {"blueprint": json.dumps({"instruction": "Generate the content
neutrally."})}

    return create_mcp_message("Librarian", content)
```

Denis Rothman *Context Engineering for Multi-Agent Systems: Move beyond prompting to build a Context Engine, a transparent architecture of context and reasoning* Packt Publishing, 2025.

Context Engineering (10/11)



Denis Rothman *Context Engineering for Multi-Agent Systems: Move beyond prompting to build a Context Engine, a transparent architecture of context and reasoning* Packt Publishing, 2025.

Context Engineering (11/11)

Integrate a new agent:

1. Add the `agent_description` function
   to the **Agent Registry**
2. Update the `get_capabilities_description()`
   the text that the **Planner** reads to understand
   - what tools exist - how to use them

Future directions

- Can RAG answer causal queries?
  "What are the strongest causal influences ...?"

- YES, if your documents are stored
  in a causal database (CDB) [15]

Text → Causal Models → CausalDB → Causal query answering ✓

Acknowledgments (1/2)

- RINGANA's AI Team:
  Mona Saleh, Thomas Kopper, Niklas Muhr
  https://www.ringana.com/
- Connor Alkin, Rohan Thomas, Damien Gasparina, Victoria Slocum
  https://weaviate.io/
- Denis Rothman
  https://www.linkedin.com/in/denis-rothman/
- Bodgan Pirvu
  https://www.linkedin.com/in/bogdan-pirvu/

Acknowledgments (2/2)

- 🌐 Austrian Chamber for Workers and Employees (AK)
- 🌐 BOKU Staff council - scientific staff
- 🌐 Austrian Data Protection Authority
- 🌐 Austrian Federal Ministry of the Interior (Bundesministerium für Inneres)
- 🌐 Federal Criminal Police Office (Bundeskriminalamt)
- 🌐 Victims of Cybercrime in the City of Vienna (Opfer von Internet-Kriminalität der Stadt Wien)

Questions?

Contact **only** on LinkedIn:

https://www.linkedin.com/in/dr-techn-dipl-ing-anna-saranti-865b7812a/

due to reasons explained on:
https://annasaranti.ai/personal-statement/

---

https://www.ringana.com/
https://www.sofresh-it.com/

References (1/10)

[1] Michael Günther, Isabelle Mohr, Daniel James Williams, Bo Wang and Han Xiao
Late chunking: contextual chunk embeddings using long-context embedding models
arXiv preprint arXiv:2409.04701, 2024.

[2] Tony Larson
Vector Database Engineering: Building Scalable AI Search & Retrieval Systems with FAISS, Milvus, Pinecone, Weaviate, RAG Pipelines, Embeddings, High Dimension Indexing (with Mathematical Equations)
Amazon Digital Services LLC - Kdp, 2025

References (2/10)

[3] Wensheng Lu, Keyu Chen, Ruizhi Qiao and Xing Sun
HiChunk: Evaluating and Enhancing Retrieval-Augmented Generation with Hierarchical Chunking
arXiv preprint arXiv:2509.11552, 2025

[4] Shuchen Wu, Noémi Éltető Ishita Dasgupta and Eric Schulz
Learning Structure from the Ground up —Hierarchical Representation Learning by Chunking
Advances in Neural Information Processing Systems, 35, 36706-36721, 2022

References (3/10)

[5] Christian Hinkler
Practical Weaviate: RAG and Vector Database Patterns for LLMs
Independently published, 2025.

[6] Manson Wall
Vector Databases: Practical AI Workflows, Mathematical Foundations, and Real-World Applications with FAISS, Milvus, Pinecone, Weaviate, and Chroma.
Independently published, 2025

References (4/10)

[7] Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer New York, 2006

[8] Yu A. Malkov and Dmitry A. Yashunin
Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(4):824–836, 2018

[9] Stephen Robertson, Hugo Zaragoza
The Probabilistic Relevance Framework: BM25 and Beyond (Vol. 4)
Now Publishers Inc, 2009

References (5/10)

[10] Aurélien Géron
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems
O'Reilly Media, 2019

References (6/10)

[11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger and Ilya Sutskever
Learning Transferable Visual Models From Natural Language Supervision
International conference on machine learning. PmLR, 2021.

References (7/10)

[12] Shahul Es, Jithin James, Luis Espinosa Anke and Steven Schockaert
Ragas: Automated Evaluation of Retrieval Augmented Generation
Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations., 2024

[13] Nelson F. Liu, Tianyi Zhang and Percy Liang
Evaluating Verifiability in Generative Search Engines
arXiv preprint arXiv:2304.09848, 2023

References (8/10)

[14] Matthias Fey, Vid Kocijan, Federico Lopez, Jan Eric Lenssen and Jure Leskovec
KumoRFM: A Foundation Model for In-Context Learning on Relational Data
https://kumo.ai/company/news/kumo-relational-foundation-model/

[15] Sridhar Mahadevan
Csql: Mapping Documents into Causal Databases
arXiv preprint arXiv:2601.08109, 2026

References (9/10)

[16] Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, Chenlin Zhou, Jiayi Mao, Tianze Xia, Jiafeng Guo and Shenghua Liu
A Survey of Context Engineering for Large Language Models
arXiv preprint arXiv:2507.13334, 2025

References (10/10)

[17] Denis Rothman
Context Engineering for Multi-Agent Systems: Move beyond prompting to build a Context Engine, a transparent architecture of context and reasoning
Packt Publishing, 2025.

[18] Weaviate
Context Engineering
Weaviate, 2025.