



Building a RAG Pipeline with MLOps Backend to Support Research in Ecology

SCREENSHOTS

Alexandra Sébastien
Arnaud Vanwelsenaere
Boris Zok

Project developed as part of the course:

Current Trends in Artificial Intelligence

INFO-H512

Prof. Hugues BERSINI, Johan LOECKX

Project related to the presentation "LLMs and RAG", by Lluc Bono, May 16th, 2025

Screen for exploring the data.

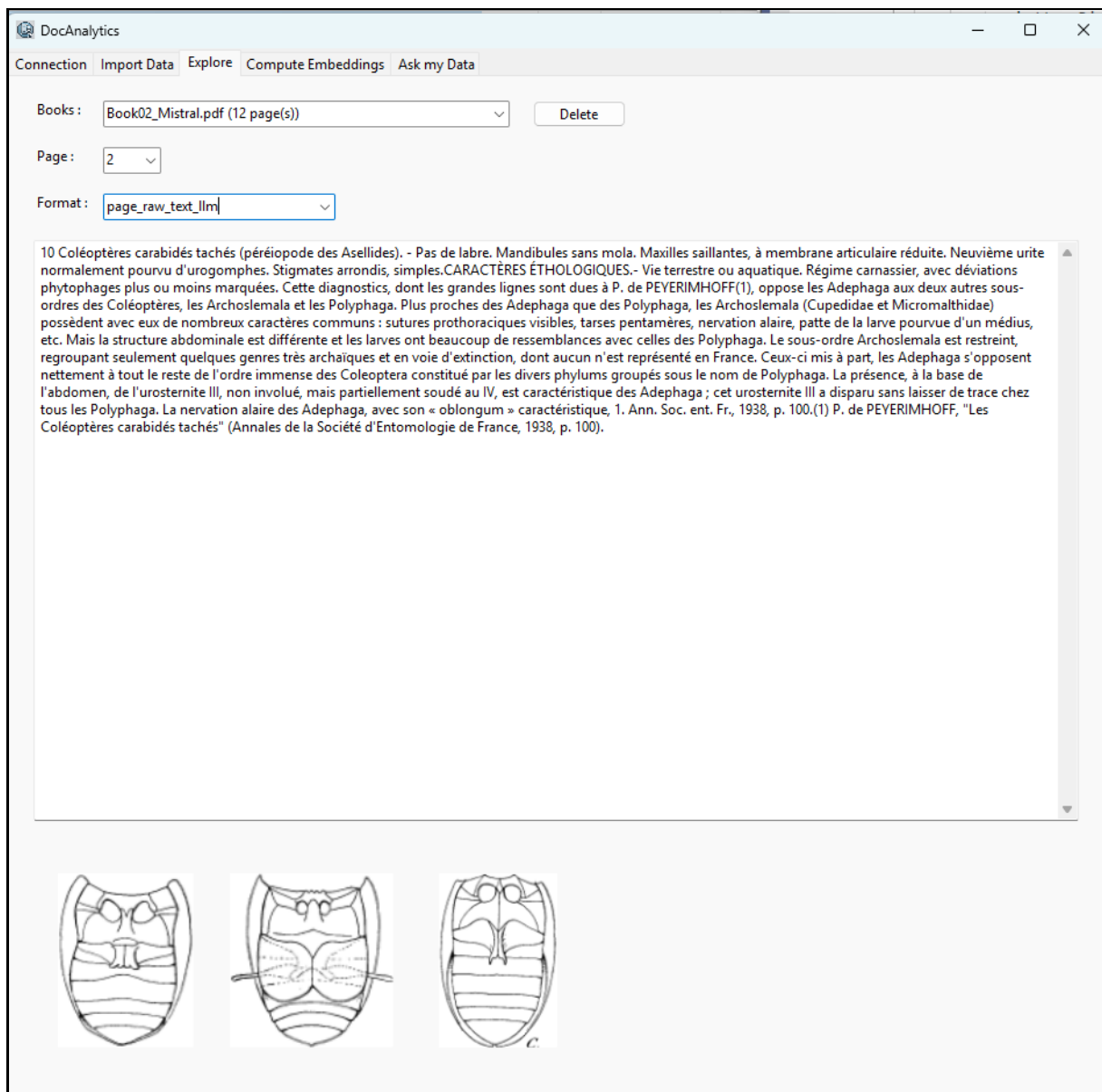


Figure 1

The user can select a document previously inserted into the SQL Server database. Then, for each page, they can choose the format of the text to view. The available formats are:

- **Raw** – the raw OCR output, unfiltered and uncorrected, as initially extracted from the scanned page using Tesseract.
- **OCR** – a refined version of the raw OCR result with basic whitespace normalization and line merging.

- **Cleaned** – an improved version of the OCR result where punctuation, sentence boundaries, and broken words have been heuristically corrected. This format aims to make the text more readable while remaining faithful to the original.
- **LLM** – a minimal rewrite of the page content produced by a local language model (LLM), preserving the original scientific content but improving grammar, style, and sentence flow.
- **LLM Explain** – an LLM-generated explanation or paraphrase of the page in plain language, designed to help users understand old or complex scientific terminology.

These formats allow the user to explore the content of the historical documents from multiple perspectives, whether they seek raw accuracy, enhanced readability, or simplified explanation.

Additionally, if scientific illustrations or figures were detected during the document import phase (via image segmentation and contour detection), they are also stored in the database and are displayed automatically for the corresponding page.

Embedding Computation Process and Performance Observations

The application includes a dedicated tab for computing semantic embeddings for each page of the selected document. As shown in **Figures 2 & 3**, once a document is selected, the user can trigger the embedding process via the "Compute" button. A progress bar tracks the operation in real time. When the computation completes, the application displays the number of chunks (i.e., text segments) generated and the total number of vector values inserted into the database.

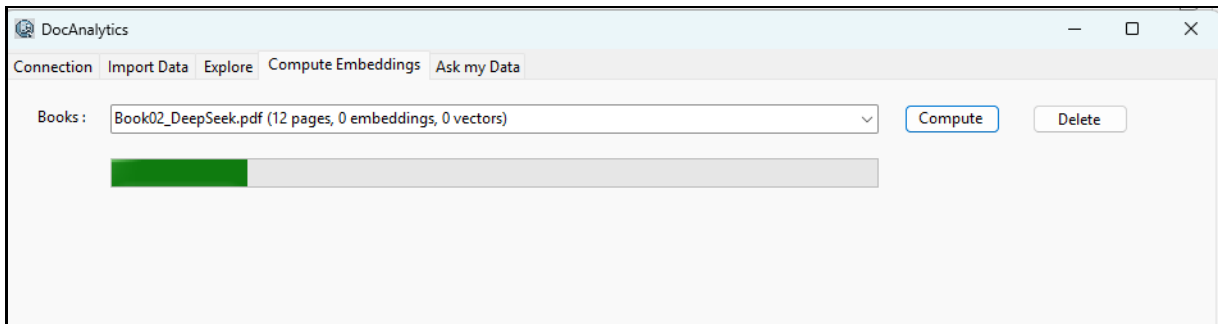


Figure 2

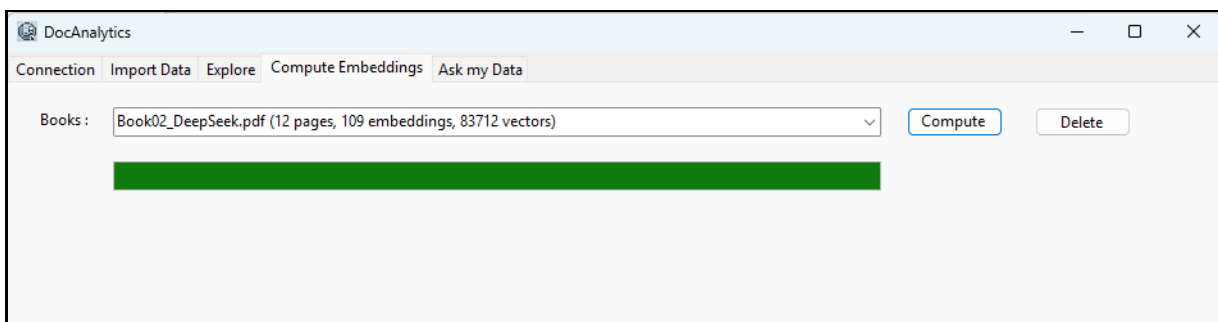
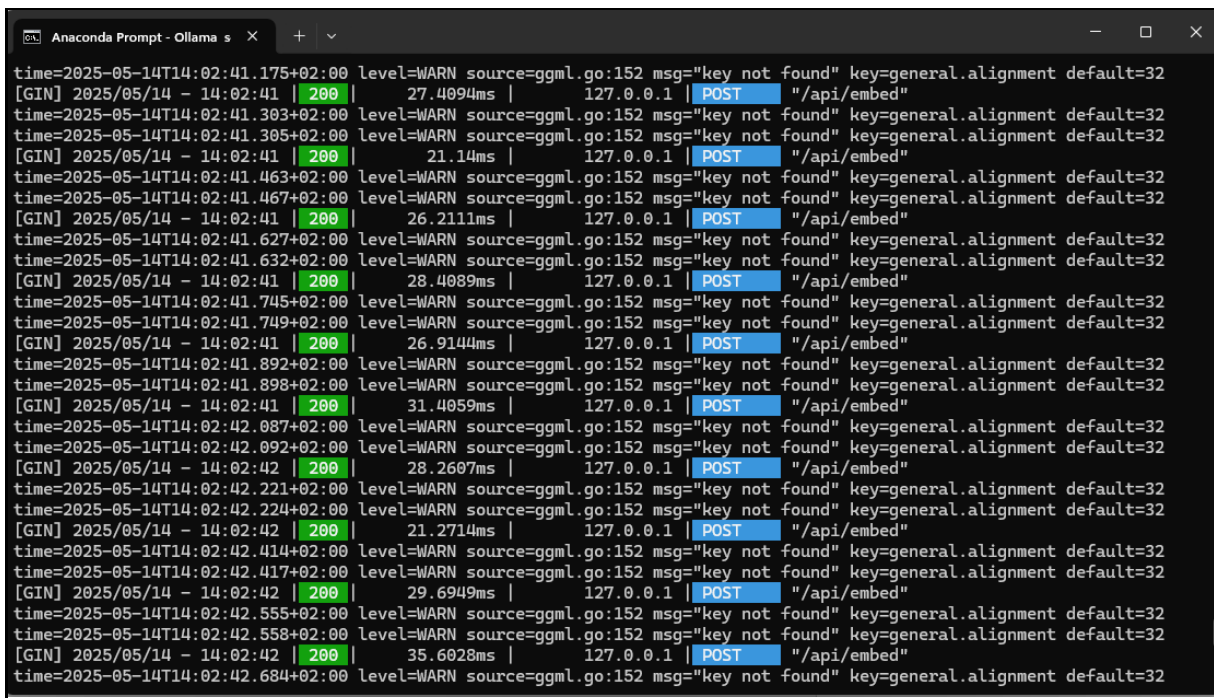


Figure 3

Internally, the system breaks down each cleaned or enhanced page into smaller textual chunks before computing an embedding vector with the nomic-embed-text model.

The embedding process leverages Ollama, a local inference engine that exposes a simple HTTP API. The C# Windows application communicates with Ollama via standard POST requests to the /api/embed endpoint.

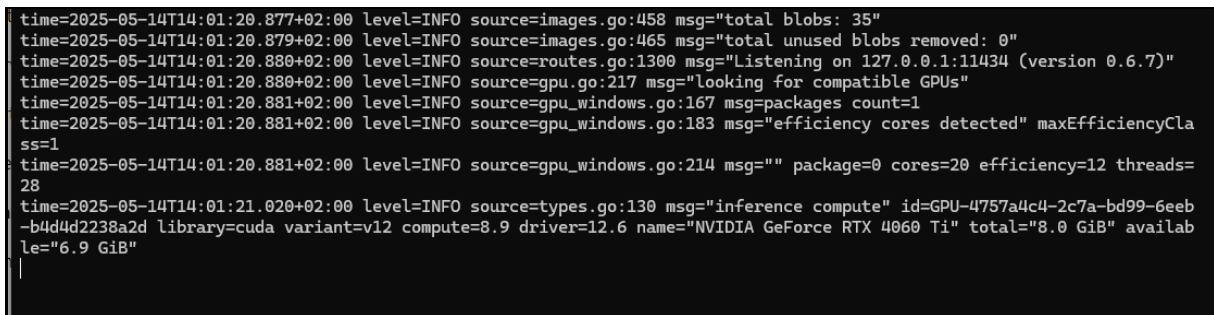
This design allows for simple, modular integration with any LLM or embedding model available through Ollama, and ensures that all operations remain fully local, with no data ever leaving the machine—a critical feature for sensitive or proprietary documents.



```
time=2025-05-14T14:02:41.175+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
[GIN] 2025/05/14 - 14:02:41 | 200 | 27.4094ms | 127.0.0.1 | POST | "/api/embed"
time=2025-05-14T14:02:41.303+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
time=2025-05-14T14:02:41.305+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
[GIN] 2025/05/14 - 14:02:41 | 200 | 21.14ms | 127.0.0.1 | POST | "/api/embed"
time=2025-05-14T14:02:41.463+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
time=2025-05-14T14:02:41.467+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
[GIN] 2025/05/14 - 14:02:41 | 200 | 26.2111ms | 127.0.0.1 | POST | "/api/embed"
time=2025-05-14T14:02:41.627+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
time=2025-05-14T14:02:41.632+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
[GIN] 2025/05/14 - 14:02:41 | 200 | 28.4089ms | 127.0.0.1 | POST | "/api/embed"
time=2025-05-14T14:02:41.745+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
time=2025-05-14T14:02:41.749+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
[GIN] 2025/05/14 - 14:02:41 | 200 | 26.9144ms | 127.0.0.1 | POST | "/api/embed"
time=2025-05-14T14:02:41.892+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
time=2025-05-14T14:02:41.898+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
[GIN] 2025/05/14 - 14:02:41 | 200 | 31.4059ms | 127.0.0.1 | POST | "/api/embed"
time=2025-05-14T14:02:42.087+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
time=2025-05-14T14:02:42.092+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
[GIN] 2025/05/14 - 14:02:42 | 200 | 28.2607ms | 127.0.0.1 | POST | "/api/embed"
time=2025-05-14T14:02:42.221+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
time=2025-05-14T14:02:42.224+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
[GIN] 2025/05/14 - 14:02:42 | 200 | 21.2714ms | 127.0.0.1 | POST | "/api/embed"
time=2025-05-14T14:02:42.414+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
time=2025-05-14T14:02:42.417+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
[GIN] 2025/05/14 - 14:02:42 | 200 | 29.6949ms | 127.0.0.1 | POST | "/api/embed"
time=2025-05-14T14:02:42.555+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
time=2025-05-14T14:02:42.558+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
[GIN] 2025/05/14 - 14:02:42 | 200 | 35.6028ms | 127.0.0.1 | POST | "/api/embed"
time=2025-05-14T14:02:42.684+02:00 level=WARN source=ggml.go:152 msg="key not found" key=general.alignment default=32
```

Figure 4

As shown in **Figure 4**, Ollama logs confirm that the application successfully sends multiple POST requests during embedding computation. All requests are answered with HTTP 200 and complete in ~20–30ms per chunk.



```
time=2025-05-14T14:01:20.877+02:00 level=INFO source=images.go:458 msg="total blobs: 35"
time=2025-05-14T14:01:20.879+02:00 level=INFO source=images.go:465 msg="total unused blobs removed: 0"
time=2025-05-14T14:01:20.880+02:00 level=INFO source=routes.go:1300 msg="Listening on 127.0.0.1:11434 (version 0.6.7)"
time=2025-05-14T14:01:20.880+02:00 level=INFO source=gpu.go:217 msg="Looking for compatible GPUs"
time=2025-05-14T14:01:20.881+02:00 level=INFO source=gpu_windows.go:167 msg="packages count=1"
time=2025-05-14T14:01:20.881+02:00 level=INFO source=gpu_windows.go:183 msg="efficiency cores detected" maxEfficiencyCores=1
time=2025-05-14T14:01:20.881+02:00 level=INFO source=gpu_windows.go:214 msg="" package=0 cores=20 efficiency=12 threads=28
time=2025-05-14T14:01:21.020+02:00 level=INFO source=types.go:130 msg="inference compute" id=GPU-4757a4c4-2c7a-bd99-6eeb-b4d4d2238a2d library=cuda variant=v12 compute=8.9 driver=12.6 name="NVIDIA GeForce RTX 4060 Ti" total="8.0 GiB" available="6.9 GiB"
```

Figure 5

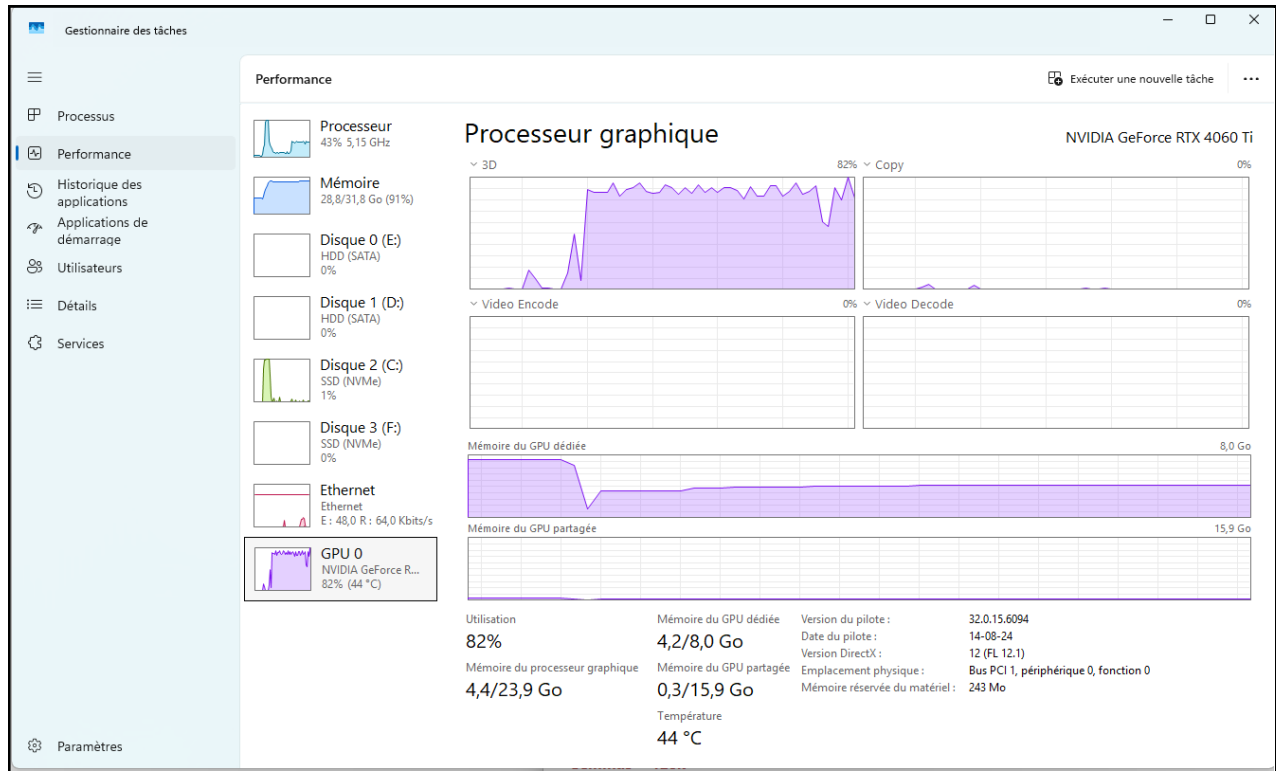


Figure 6

When running on a high-performance PC equipped with an NVIDIA RTX 4060 Ti, the application benefits from hardware acceleration. Ollama automatically detects available GPUs at startup (see Figure 5) and offloads the embedding model onto the GPU. As shown in Figure 6, GPU usage spikes to over 90% during vector computation, while the CPU remains moderately active, demonstrating effective parallelism and resource utilization. The GPU memory footprint is also optimized, using only 4GB of dedicated memory out of the 8GB available.

Interactive Exploration and Semantic Question Answering

The final part of the system focuses on user-facing interaction with the digitized documents through a dedicated "Ask my Data" interface. This tab allows users to ask natural language questions and receive contextual answers derived from the indexed scientific books. The interface has been designed to provide full control over the query process, while ensuring an intuitive and dynamic user experience.

As shown in Figure 7, the user can configure several advanced options before submitting a question:

- **Model selection:** choose among locally deployed models like *Mistral*, *DeepSeek*, *Gemma*, or *Mixtral*.
- **Top N chunks:** define how many document segments (chunks) should be retrieved and forwarded as context to the LLM.
- **Language:** set the preferred output language of the answer (French, English, or Dutch).

- **Token limit:** specify the maximum number of tokens to allow the LLM to generate.
- **Target book:** limit the search to a specific document or query across the entire library.

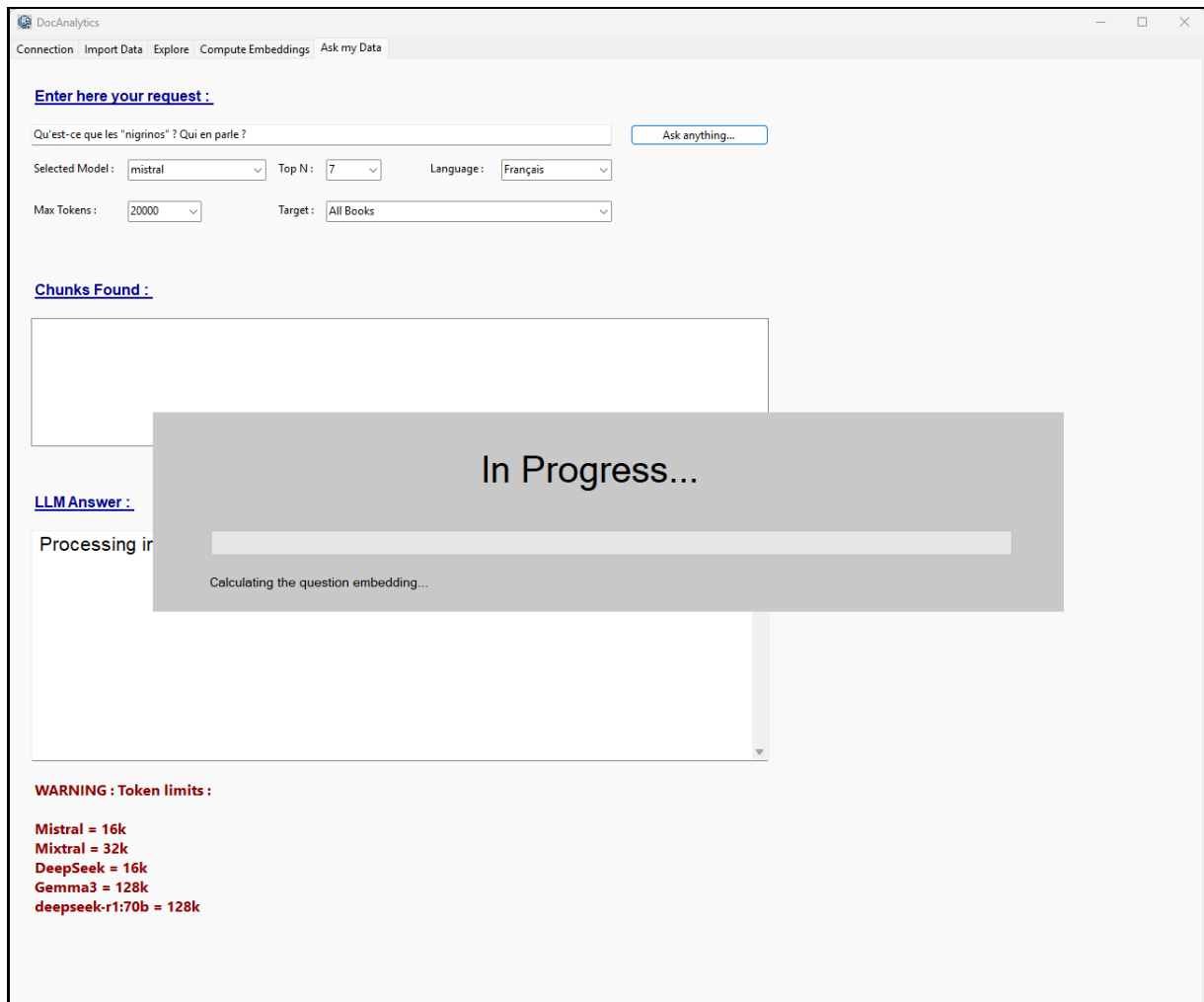


Figure 7

When a query is submitted, the system displays a progress overlay (**Figure 7**), computes the embedding of the question, retrieves the most semantically similar text segments using cosine similarity, and sends them as structured context to the selected LLM. Responses are streamed live and appended incrementally to the LLM output panel (**Figure 9**).

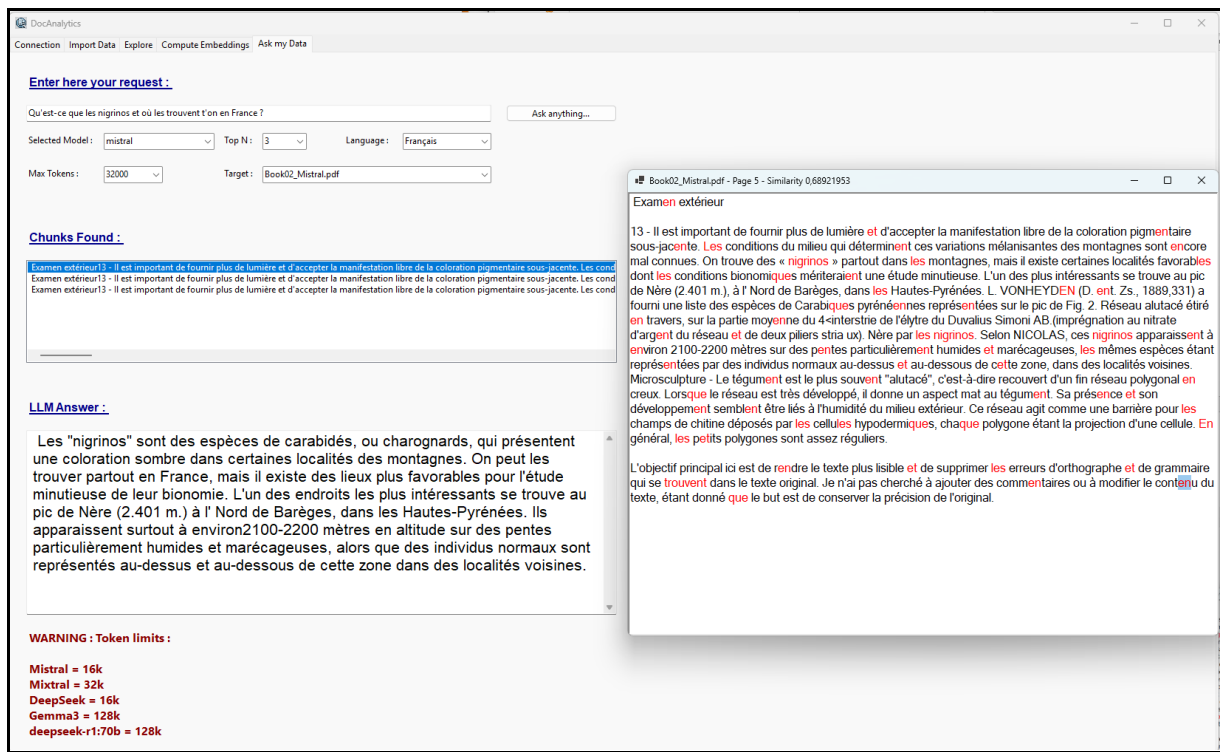


Figure 8

Each retrieved chunk is shown in a scrollable list, and users can click on any chunk to display the full page from which it was extracted, as well as the exact similarity score that determined its relevance. This allows users to verify the precision and traceability of the semantic search pipeline.

As illustrated in **Figure 8**, the answers generated by the LLM are not generic; they are clearly grounded in the scientific content of the documents. For instance, questions like “*Que sais-tu du fouet de la série ombiliquée d’un Aphaenops ?*” or “*Qu’est-ce que les nigrinos et où les trouve-t-on en France ?*” yield accurate, context-aware answers that reference entomological concepts and even specific species and locations described in the original works.

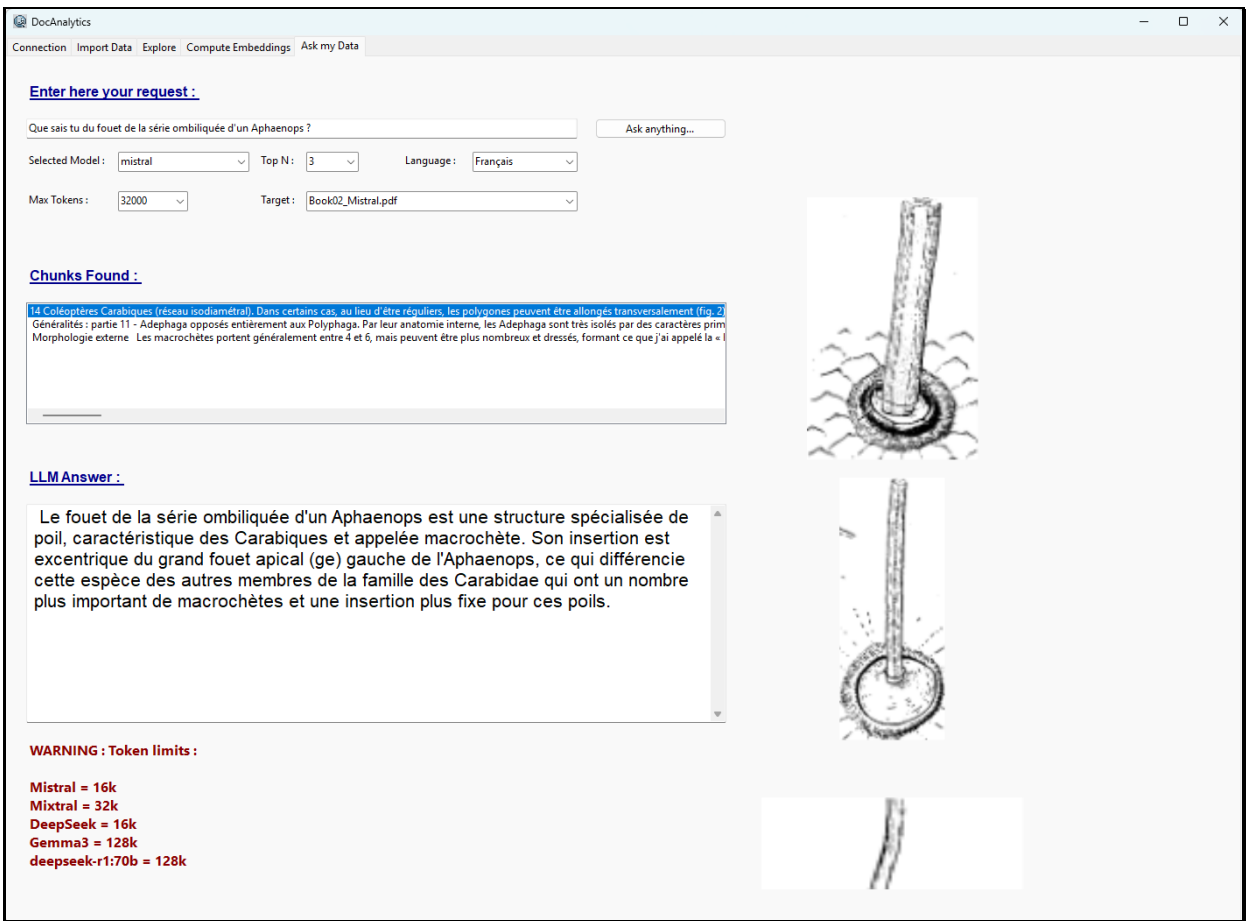


Figure 9

Moreover, when pages include scientific illustrations, such as anatomical sketches or structural diagrams, these are automatically displayed in a vertical scroll panel. Each image is clickable and opens in a zoomed popup viewer, allowing users to explore the visual content in detail alongside the explanatory text. This component of the system showcases the fusion of semantic AI with classical academic publishing—allowing users to interact meaningfully with scanned books using modern language models, without losing access to the visual and textual richness of the original documents.

SQL Data model

