# MMORE: Massive Multimodal Open RAG & Extraction

**Anonymous Authors**[1]

## Abstract

We introduce **MMORE**, an open-source pipeline for **M**assive **M**ultimodal **O**pen **R**etrieval-Augmented Generation and **E**xtraction, designed to ingest, transform, and retrieve knowledge from heterogeneous document formats at scale. MMORE supports more than fifteen file types, including text, tables, images, emails, audio, and video, and processes them into a unified format to enable downstream applications for LLMs. The architecture offers modular, distributed processing, enabling scalable parallelization across CPUs and GPUs. On processing benchmarks, MMORE demonstrates a 3.8-fold speedup over single-node baselines and 40% higher accuracy than Docling on scanned PDFs. The pipeline integrates hybrid dense-sparse retrieval and supports both interactive APIs and batch RAG endpoints. Evaluated on PubMedQA, MMORE-augmented medical LLMs improve biomedical QA accuracy with increasing retrieval depth. MMORE provides a robust, extensible foundation for deploying task-agnostic RAG systems on diverse, real-world multimodal data.

## 1. Introduction

As of 2025, the public web is conservatively estimated to host more than 2.5 trillion PDF documents, alongside petabytes of mixed-modality slide decks, spreadsheets, images, and audiovisual artefacts (CloudFiles). Yet fewer than one percent of these resources are represented in popular machine-learning corpora as they are remain locked behind brittle, heterogeneous formats that frustrate automated parsing at scale. Existing pipelines rely on ad hoc mosaics of format-specific utilities, limiting throughput, reproducibility, and long-term maintainability.

As data-supply forecasts estimate that the pool of high-quality human-generated text could be exhausted by prevailing scaling trends as early as 2026 (Villalobos et al., 2022; 2024), it has become essential to find more format-agnostic preprocessing workflows. Much of this data, particularly in specialized or institutional settings, is unavailable for training but remains crucial for improving the verifiability of LLM outputs through RAG. Hallucinations (OpenAI, 2025) and factual drift (Huang et al., 2025) remain significant challenges, and robust RAG pipelines are increasingly explored as a means to mitigate these issues, thereby reducing the burden of manual validation and better aligning model outputs with trustworthy source material.

To address these limitations, we introduce MMORE an open-source tool for **M**assive **M**ultimodal **O**pen **R**etrieval-Augmented Generation and **E**xtraction, a unified pipeline for scalable extraction, transformation, and retrieval of multimodal data. MMORE supports diverse formats such as documents, presentations, spreadsheets, and multimedia and integrates them into a structured knowledge base, enabling LLMs to access accurate, contextually grounded information via the RAG paradigm.

Designed for modularity and scalability, our pipeline natively supports parallelized processing across multi-node architectures and distributed environments such as Kubernetes clusters. Compared to Docling demonstrates more than 2-fold faster end-to-end processing, while achieving 40% higher layout accuracy on scanned PDFs. In distributed mode, we show that our pipeline processes 720 pages in 185s using four nodes, resulting in 3.8-fold speedup over single-node mode. The results demonstrate MMORE's effectiveness as a scalable, high-accuracy solution for multimodal document processing in real-world deployment.

## 2. Related Work

Large-scale transformation of unstructured documents into structured, machine-readable format has attracted substantial attention. We group prior work into two strands: **(i)** document ingestion and parsing pipelines, and **(ii)** RAG frameworks. To our knowledge, neither line of work simultaneously offers the modality coverage and end-to-end throughput required for industrial- and small-scale multimodal assistants that we target with MMORE.

**Document Ingestion Pipelines.** GPU-accelerated microservice suites such as *NV-Ingest* (Team, 2024) convert PDFs and office documents into page-level JSON enriched with text blocks, tables, and graphics, and can optionally export embeddings for downstream indexing. *Docling* (Auer et al., 2024) extends the modality set to spreadsheets, and other common formats, but executes primarily on a single node and therefore exhibits limited throughput in production settings. Classical OCR tools like *doctr* (Mindee, 2021) handle

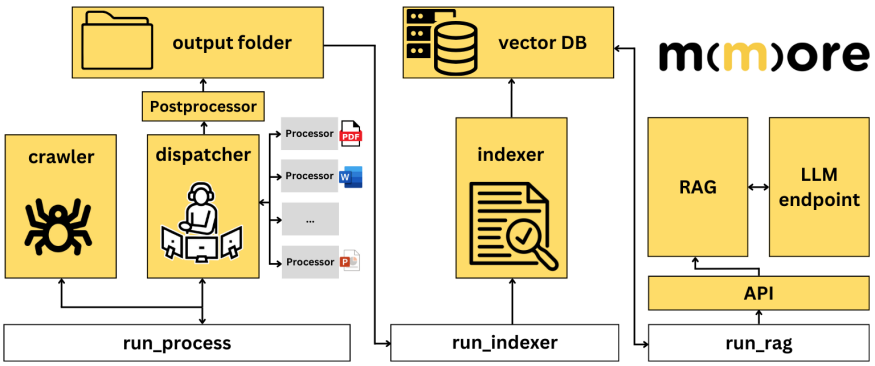


*Figure 1.* The end-to-end pipeline from file-type–specific processing to retrieval-augmented generation (RAG).

text detection and recognition but rely on external systems for layout, embeddings, and indexing. *Surya* (Paruchuri & Team, 2025) adds multilingual OCR and layout analysis but lacks built-in multi-GPU or cluster parallelism. Commercial services such as *LLMWhisperer* (Unstract, 2025) offer similar functionality behind a paywall, which restricts reproducibility and hinders open experimentation. In contrast, MMORE combines extraction, transformation, embedding, and indexing into a single open-source pipeline that natively parallelizes across multi-node, multi-GPU deployments. Moreover, MMORE uniquely handles audiovisual assets, enabling unified RAG over text, images, and time-based media.

**RAG Frameworks.** Open-source libraries such as *LangChain* (Chase, 2022) and *LlamaIndex* (Liu, 2022) provide high-level abstractions for chunking, embedding, retrieval, and prompting. However, they rely on external loaders for modality-specific parsing and give no guidance on efficient high-throughput ingestion. Several recent pipelines, such as *Unstructured.io* (uns, 2025) and Haystack (Pietsch et al., 2019) for document parsing, or *M3IT* (Li et al., 2023) and *OpenFlamingo* (Awadalla et al., 2023) for multimodal model alignment, address specific components of this pipeline. Yet none provide an integrated, open-source framework that supports ingestion, transformation, and retrieval across heterogeneous, real-world file types at scale.

MMORE combines a scalable ingestion layer with a task-agnostic retrieval API, unifying document processing and RAG tools to enable multimodal assistants from raw enterprise data in one library.

## 3. Architecture

MMORE provides an end-to-end platform, enabling users to process large document collections, build retrieval indices, and query LLMs with relevant multimodal content, all within a unified framework, as illustrated in Figure 1.

### 3.1. Processing

At the core of MMORE lies a modular, scalable processing pipeline, designed for efficient, multimodal data extraction. Importantly, MMORE reuses open-source extraction tools such as *Surya* (Paruchuri & Team, 2025) for PDF parsing, *Whisper* (Radford et al., 2023) for audio transcription, and standard Python libraries for office file formats, allowing us to focus on scalable orchestration and integration. A complete list of supported extractors is provided in Appendix A.1. The design prioritizes three main strengths: **(i)** multimodal document processing, **(ii)** extensibility to new file types, and **(iii)** high-throughput distributed execution.

**Multimodal Data Extraction.** The processor module extracts heterogeneous content from documents and standardizes it into a unified JSON-based format, referred to as the *MultimodalSample* (see Appendix A.2). Each sample consists of plain text interleaved with modality placeholders (e.g. images) and a list of the extracted modalities, preserving their type and location. Embedded media are extracted and saved to disk, with placeholder tokens (e.g., `<attachment>`) inserted at the corresponding positions within the text. This design supports downstream tasks that require text with tightly linked visual elements, such as multimodal pre-training or RAG.

**Extensibility.** To facilitate extensibility, we designed a common processor interface that abstracts file-specific handling into modular components. Adding support for a new file type requires only implementing a lightweight subclass, promoting long-term maintainability and community-driven contributions. Each processor needs to define a class that takes a file path as input and outputs a *MultimodalSample*, leveraging the standardized output format across the system. To date, MMORE supports more than 15 file types, including, but not limited to, PDFs, DOCX, PPTX, spreadsheets, media files, emails, and HTML pages.

**Distributed Processing.** MMORE natively supports both intra-node and inter-node parallelization, exploiting all available CPU and GPU resources without requiring manual

configuration from the user. The system is built on top of *Dask* (Dask Development Team, 2016), enabling automatic workload balancing, fault tolerance, and seamless scaling across deployment settings, from standalone machines to large multi-node clusters. This design scales across use cases, from individual researchers to large organizations. To further support both ends of the spectrum, MMORE offers two processing modes: a fast mode for speed and a default mode for accuracy, allowing users to balance performance and fidelity as needed.

### 3.2. RAG

The RAG pipeline is composed of three independent components: **(i)** post-processing, **(ii)** indexing and retrieval, and **(iii)** an integrated RAG service. Each part is modular and can be run independently.

**Post-processing.** This stage filters the extracted text to improve quality for downstream tasks. MMORE exploits the existing *datatrove* (Penedo et al., 2024), a high-throughput filtering library, and includes native support for several post-processing components, including Named Entity Recognition, Chunking, and Tagging.

**Indexing and Retrieval.** Indexing is crucial to RAG performance, as retrieval relies on how documents are represented. MMORE uses a hybrid indexing strategy, storing both sparse and dense embeddings for each document. Sparse representations support lexical matching and improve interpretability, while dense embeddings enable semantic search using neural similarity. This duality allows users to choose or combine retrieval embeddings depending on their downstream task. The retriever is accessible via our integrated RAG system or as a standalone API.

**Integrated RAG system.** The RAG system supports both API-based querying and offline batch processing. In batch mode, users provide a JSONL file containing retrieval queries; the system processes each entry and saves the results to a new JSONL file. Both modes allow customization of the model, prompt template, index source, and other parameters via configuration files or API options.

## 4. Evaluation Setup

We evaluate MMORE's processing and RAG modules independently. Below, we detail our methodology for assessing efficiency, accuracy, and scalability.

### 4.1. Processing

The processing module is evaluated along two axes: efficiency and accuracy, versus *Docling* (Auer et al., 2024) as a baseline due to its popularity and ease of use.

**Efficiency.** We benchmark processing speed using a single A100 80GB. For scalability analysis, we use an 18-page paper and synthetically generate longer documents by duplicating its content to reach 36, 54, 90, to 720 pages. This setup allows us to test throughput for both single-device and distributed processing. The distributed experiments are conducted on a Kubernetes cluster with 1 vs 4 nodes (1 A100 per node) to evaluate parallelization efficiency. To highlight MMORE's strength in handling heterogeneous data, we also evaluate its performance across a diverse set of 19 files, spanning 9 unique file types.

**Accuracy.** To assess text extraction quality, we create a benchmark using public-domain books from Project Gutenberg (Project Gutenberg, 1971) by pairing PDF inputs with their corresponding plain-text ground truths. We select two contrasting cases: "The Blue Castle" (a clean, digital-friendly PDF) and "The Great Gatsby" (a scanned, image-based file). Each document is truncated to 50k characters to ensure computational feasibility, particularly for metrics like Levenshtein distance. We report standard metrics: BLEU (Papinesi, 2002) for n-gram overlap, ROUGE-L (Lin, 2004), and character error rate (CER) (Navarro, 2001). Metric formulations are provided in the Appendix A.3

### 4.2. RAG

To evaluate our RAG pipeline, we focus on the PubMedQA benchmark (Jin et al., 2019), a biomedical question-answering task. We construct a retrieval corpus by indexing all PubMed abstracts and conclusions into a dense vector database using MMORE. At inference time, the top-$k$ most relevant documents are retrieved using a similarity search and prepended to the original question as context for the language model. We experiment with both Meditron3-8B and Meditron3-70B (Sallinen et al., 2025), evaluating how different values of $k$ affect downstream accuracy. This setup isolates the effect of retrieval depth on performance within a consistent biomedical knowledge source.

## 5. Results

### 5.1. Processing

**Efficiency.** Figure 2 shows comparison of *Docling* and MMORE. On short documents (36 pages) *Docling* is marginally faster than MMORE (default). The difference disappears at 90 pages and shifts in favor of MMORE beyond 180 pages, where our pipeline scales almost linearly while *Docling* slows down super-linearly. The fast mode, which omits OCR, delivers an additional speed-up of roughly two to three times. Running the default pipeline on four nodes achieves a 3.8-fold reduction in latency compared to the single-node baseline, surpassing even the single-node fast mode and clearly demonstrating the efficiency and scalability of the distributed execution in MMORE. It is also worth mentioning that the batch size is user-configurable.

| Method | *The Blue Castle (digital PDF)* | | | *The Great Gatsby (scanned images)* | | |
|---|---|---|---|---|---|---|
| | BLEU↑ | ROUGE-L↑ | CER↓ | BLEU↑ | ROUGE-L↑ | CER↓ |
| MMORE | 0.8608 | 0.9940 | 0.0241 | **0.7973** | **0.9813** | **0.0295** |
| MMORE (fast) | 0.8639 | **0.9963** | 0.0206 | 0.0000 | 0.0000 | 1.0000 |
| Docling | **0.8643** | 0.9959 | **0.0199** | 0.5451 | 0.6582 | 0.5518 |

*Table 1.* Accuracy evaluation on two Project Gutenberg books: "*The Blue Castle*" and "*The Great Gatsby*".

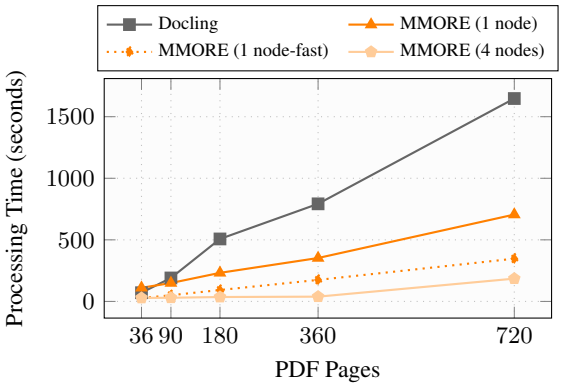


*Figure 2.* Processing time vs. PDF length for *Docling* and MMORE. MMORE (1 node-fast) disables OCR for performance, and MMORE (4 nodes) uses distributed processing.

The experiments presented here used a conservative default, leaving around 65GB of the 80GB GPU unused. This highlights the potential for further optimization, as users can adjust the configuration to fully exploit available hardware resources. Table 2 further illustrates the performance advantage of MMORE across multiple file types. In default mode, MMORE reduces the total processing time by 45.48% compared to *Docling*, with the fast mode achieving an even more pronounced improvement of 155.38%.

| Metric | Docling | MMORE default | MMORE fast |
|---|---|---|---|
| Total Time (s) | 522.98 | 358.93 | 204.57 |
| Num. of Unsupported Files | 5 | 0 | 0 |
| Relative Efficiency | baseline | +45.48% | +155.38% |

*Table 2.* Processing speeds for 9 unique file types - PDF, DOCX, EML, MD, MP4, MP3, PPTX, TXT, XLSX (19 files in total).

**Accuracy.** Table 1 reports BLEU, ROUGE-L, and CER on two Project Gutenberg titles. On the digitally formatted "Blue Castle" book, all three systems achieve near-perfect scores, with *Docling* attaining the lowest CER (1.99%); however, differences remain negligible. The scanned version of "The Great Gatsby", an image-based document requiring OCR, provides a greater challenge. Here, MMORE fast predictably fails, as it omits OCR entirely. In contrast, MMORE default maintains high extraction fidelity, clearly outperforming *Docling*, whose CER of 55% indicates significant OCR errors. Although these results demonstrate the accuracy of our pipeline, further benchmarking on a larger and more diverse set of documents is necessary to robustly validate its generalization capabilities.

### 5.2. RAG

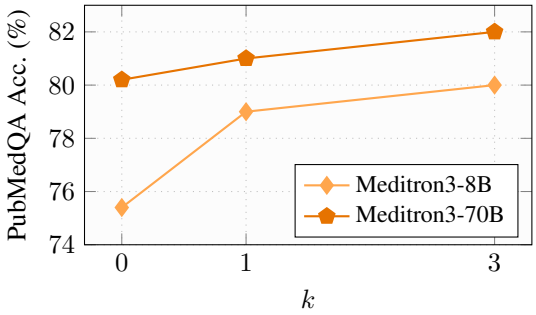


*Figure 3.* Effect of retrieved documents ($k$) on PubMedQA accuracy for Meditron models using MMORE's built-in RAG.

To evaluate RAG performance, we test the Meditron-3 model family with various RAG configurations on the PubMedQA benchmark. Figure 3 shows that both Meditron-3[8B] and Meditron-3[70B] (Sallinen et al., 2025) consistently improve accuracy with RAG, especially as the number of retrieved documents $k$ increases. These results demonstrate that our RAG pipeline effectively injects domain-specific context at inference time, improving answer accuracy.

## 6. Conclusion

MMORE is a scalable, open-source pipeline for retrieval-augmented generation over diverse, real-world data. It supports more than 15 file types, including PDFs, spreadsheets, images, audio, and video, and enables structured, high-throughput integration into LLM workflows.

Our results show that MMORE outperforms *Docling* in both speed and layout fidelity, particularly in OCR-heavy documents, and improves biomedical QA accuracy on PubMedQA via efficient RAG pipelines.

Built for extensibility and deployment at scale, MMORE provides a flexible foundation for verifiable, multimodal LLM applications. Future work will expand support for multilingual retrieval, audiovisual alignment, and federated processing in privacy-sensitive settings.

## References


Unstructured: Open-source pre-processing tools for unstructured data. https://github.com/Unstructured-IO/unstructured, 2025. Accessed: 2025-05-26.

Auer, C., Lysak, M., Nassar, A., Dolfi, M., Livathinos, N., Vagenas, P., Ramis, C. B., Omenetti, M., Lindlbauer, F., Dinkla, K., et al. Docling technical report. *arXiv preprint arXiv:2408.09869*, 2024.

Awadalla, A., Gao, I., Gardner, J., Hessel, J., Hanafy, Y., Zhu, W., Marathe, K., Bitton, Y., Gadre, S., Sagawa, S., et al. Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*, 2023.

Chase, H. LangChain, October 2022. URL https://github.com/langchain-ai/langchain.

CloudFiles. How many files are there in the world? URL https://www.cloudfiles.io/blog/how-many-files-are-there-in-the-world.

Dask Development Team. *Dask: Library for dynamic task scheduling*, 2016. URL http://dask.pydata.org.

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

Jin, Q., Dhingra, B., Liu, Z., Cohen, W., and Lu, X. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2567–2577, 2019.

Li, L., Yin, Y., Li, S., Chen, L., Wang, P., Ren, S., Li, M., Yang, Y., Xu, J., Sun, X., Kong, L., and Liu, Q. M$^3$it: A large-scale dataset towards multi-modal multilingual instruction tuning. *arXiv preprint arXiv:2306.04387*, 2023.

Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. pp. 74–81, 2004.

Liu, J. LlamaIndex, 11 2022. URL https://github.com/jerryjliu/llama_index.

Mindee. doctr: Document text recognition. https://github.com/mindee/doctr, 2021.

Navarro, G. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

OpenAI. Openai o3 and o4-mini system card. https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf, 2025. Accessed: 2025-05-23.

Papinesi, K. Bleu: A method for automatic evaluation of machine translation. pp. 311–318, 2002.

Paruchuri, V. and Team, D. Surya: A lightweight document ocr and analysis toolkit. https://github.com/VikParuchuri/surya, 2025. GitHub repository.

Penedo, G., Kydlíček, H., Cappelli, A., Sasko, M., and Wolf, T. Datatrove: large scale data processing, 2024. URL https://github.com/huggingface/datatrove.

Pietsch, M., Möller, T., Kostic, B., Risch, J., Pippi, M., Jobanputra, M., Zanzottera, S., Cerza, S., Blagojevic, V., Stadelmann, T., Soni, T., and Lee, S. Haystack: the end-to-end nlp framework for pragmatic builders, 2019. URL https://github.com/deepset-ai/haystack. Accessed: 2025-05-26.

Project Gutenberg. Project gutenberg. https://www.gutenberg.org/, 1971. Accessed: 2025-05-26.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pp. 28492–28518. PMLR, 2023.

Sallinen, A., Solergibert, A.-J., Zhang, M., Boyé, G., Dupont-Roc, M., Theimer-Lienhard, X., Boisson, E., Bernath, B., Hadhri, H., Tran, A., et al. Llama-3-meditron: An open-weight suite of medical llms based on llama-3.1. In *Workshop on Large Language Models and Generative AI for Health at AAAI 2025*, 2025.

Team, N. I. D. *NVIDIA Ingest: An accelerated pipeline for document ingestion*, 2024. URL https://github.com/NVIDIA/nv-ingest.

Unstract. Llmwhisperer. https://unstract.com/llmwhisperer/, 2025. Accessed: 2025-05-25.

Villalobos, P., Ho, A., Sevilla, J., Besiroglu, T., Heim, L., and Hobbhahn, M. Will we run out of data? limits of LLM scaling based on human-generated data. *arXiv preprint arXiv:2211.04325*, 2022. doi: 10.48550/arXiv.2211.04325. URL https://arxiv.org/abs/2211.04325.

Villalobos, P., Ho, A., Sevilla, J., Besiroglu, T., Heim, L., and Hobbhahn, M. Will we run out of data? limits of LLM scaling based on human-generated data (v2, 2024

update). *arXiv preprint arXiv:2211.04325v2*, 2024. doi: 10.48550/arXiv.2211.04325. URL https://arxiv.org/abs/2211.04325v2. Version 2, revised 4 June 2024.

# A. Appendix

## A.1. Document Ingestion

To better situate MMORE within the ecosystem of document ingestion systems, Table 3 presents a fine-grained comparison with two representative alternatives: *Docling* and *NV-Ingest* (part of NeMo Retriever). We evaluate them across modality support, indexing capabilities, and RAG integration. Green cells indicate native support, while grey cells denote the absence of the corresponding capability.

| Feature | Docling | NV-Ingest[1] | MMORE |
|---|---|---|---|
| *Supported Modalities* | | | |
| PDF | ✓ | ✓ | ✓ |
| DOCX | ✓ | ✓ | ✓ |
| PPTX | ✓ | ✓ | ✓ |
| XLSX / spreadsheets | ✓ | | ✓ |
| TXT | ✓ | ✓ | ✓ |
| HTML | ✓ | | ✓ |
| Markdown | ✓ | | ✓ |
| CSV | ✓ | | ✓ |
| Images (PNG/JPEG/SVG/TIFF/BMP) | ✓ | ✓ | ✓ |
| Audio | | | ✓ |
| Video | | | ✓ |
| EML | | | ✓ |
| *Indexing & Embedding* | | | |
| Native engine included | | ✓ | ✓ |
| LangChain / LlamaIndex connector | ✓ | ✓ | ✓ |
| *RAG* | | | |
| Built–in RAG pipeline | | | ✓ |
| Plugin–based RAG | ✓ | ✓ | |
| Open–Source license | MIT | Apache 2.0 | Apache 2.0 |

*Table 3.* Fine-grained comparison of Docling, NV-Ingest, and MMORE document-ingestion pipelines. Green cells indicate native support; grey cells indicate absence of the capability.

MMORE supports a wide range of file formats through modular extractors. For each supported type, we define a *default mode* prioritizing accuracy and a *fast mode* optimized for speed. When no alternative tool is available, the fast mode is left unspecified (–). A complete list of tools used per file type is shown in Table 4.

| File Type | Default Mode Tool(s) | Fast Mode Tool(s) |
|---|---|---|
| DOCX | `python-docx` for text and image extraction | – |
| MD | `markdown` for text, `markdownify` for HTML conversion | – |
| PPTX | `python-pptx` for text and image extraction | – |
| XLSX | `openpyxl` for table and text extraction | – |
| TXT | Python built-in `open()` | – |
| EML | Python built-in `email` module | – |
| Audio/Video (MP4, MP3, etc.) | `moviepy` for frames, `whisper-large-v3-turbo` for transcription | `whisper-tiny` |
| PDF | `marker-pdf` for OCR/structured data | `PyMuPDF` |
| HTML | `BeautifulSoup` | – |

*Table 4.* Overview of supported file types and extraction tools in MMORE. Full URLs are included in the project documentation.

## A.2. Multimodal Sample

The format provides a standardized representation for processed documents, combining extracted text with references to non-text elements. As shown in the example, the "text" field contains the document's content with `<attachment>` placeholders (which are configurable) marking modality locations, while the modalities array contains all embedded objects with their types and storage paths.

[1] NeMo Retriever Documentation

**Format Example:**

```
{
  "text": "A report containing a cool image <attachment> and a chart <attachment>...",
  "modalities": [
    {
      "type": "image",
      "value": "chart_url_2.png"
    },
    {
      "type": "image",
      "value": "chart_url_1.png"
    }
  ]
}
```

*The standardized format for document processing.*

### A.3. Processing Accuracy - Metrics

To quantify extraction accuracy, we used a combination of machine translation, summarization and string similarity metrics. Their definitions are given below.

**BLEU Score (bilingual evaluation understudy)** (Papinesi, 2002): The BLEU score evaluates the overlap between the n-grams (sequences of words of length $n$) between the extracted text and the ground truth. It is defined as:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{1}$$

where $p_n$ is the precision for n-grams of length $n$, ranging from [1 to 4], $w_n$ are the weights (uniform), and brevity penalty (BP), given by:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ \exp\left(1 - \frac{r}{c}\right) & \text{if } c \leq r \end{cases} \tag{2}$$

Here, $c$ is the length of the candidate (extracted) text, and $r$ is the length of the reference (ground truth). BLEU considers how much of the extracted text matches the reference text in terms of word sequences, while also penalizing outputs that are too short.

**ROUGE-L (recall-oriented understudy for gisting evaluation)** (Lin, 2004): ROUGE-L measures the quality of the extracted text using the longest common subsequence (LCS) between the extracted text and the ground truth. The LCS is the longest sequence of words appearing in the same order in both texts (though not necessarily consecutively). ROUGE-L is calculated as:

$$\text{ROUGE-L} = F_{\text{measure}} = \frac{(1 + \beta^2) \cdot \text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \tag{3}$$

where $\beta$ is a weighting factor (set to 1 for equal weighting), and:

$$\begin{aligned} \text{Precision} &= \frac{\text{LCS}}{\text{Length of Extracted Text}}, \\ \text{Recall} &= \frac{\text{LCS}}{\text{Length of Ground Truth}}. \end{aligned} \tag{4}$$

**Levenshtein distance - character error rate (CER)** (Navarro, 2001): Given two strings, $s_1$ (extracted text) and $s_2$ (ground truth), the Levenshtein distance $d(s_1, s_2)$ measures the minimum number of insertions, deletions, or substitutions required to transform $s_1$ into $s_2$. We normalize this distance over the length of the ground truth and is defined as:

$$\text{CER} = \frac{d(s_1, s_2)}{|s_1|} \tag{5}$$