# Docling: An Efficient Open-Source Toolkit for AI-driven Document Conversion

**Nikolaos Livathinos *, Christoph Auer *, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panagiotis Vagenas, Cesar Berrospi, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, Peter W. J. Staar**

IBM Research, Rüschlikon, Switzerland
Please send correspondence to: deepsearch-core@zurich.ibm.com



**Abstract**

We introduce *Docling*, an easy-to-use, self-contained, MIT-licensed, open-source toolkit for document conversion, that can parse several types of popular document formats into a unified, richly structured representation. It is powered by state-of-the-art specialized AI models for layout analysis (DocLayNet) and table structure recognition (TableFormer), and runs efficiently on commodity hardware in a small resource budget. Docling is released as a Python package and can be used as a Python API or as a CLI tool. Docling's modular architecture and efficient document representation make it easy to implement extensions, new features, models, and customizations. Docling has been already integrated in other popular open-source frameworks (e.g., LangChain, LlamaIndex, spaCy), making it a natural fit for the processing of documents and the development of high-end applications. The open-source community has fully engaged in using, promoting, and developing for Docling, which gathered 10k stars on GitHub in less than a month and was reported as the No. 1 trending repository in GitHub worldwide in November 2024.

**Repository** — https://github.com/DS4SD/docling

## 1 Introduction

Converting documents back into a unified machine-processable format has been a major challenge for decades due to their huge variability in formats, weak standardization and printing-optimized characteristic, which often discards structural features and metadata. With the advent of LLMs and popular application patterns such as retrieval-augmented generation (RAG), leveraging the rich content embedded in PDFs, Office documents, and scanned document images has become ever more relevant. In the past decade, several powerful document understanding solutions have emerged on the market, most of which are commercial software, SaaS offerings on hyperscalers (Auer et al. 2022) and most recently, multimodal vision-language models. Typically, they incur a cost (e.g., for licensing or LLM inference) and cannot be run easily on local hardware. Meanwhile, only a handful of different open-source tools cover PDF, MS Word, MS PowerPoint, Images, or HTML conversion, leaving a significant feature and quality gap to proprietary solutions.

With *Docling*, we recently open-sourced a very capable and efficient document conversion tool which builds on the powerful, specialized AI models and datasets for layout analysis and table structure recognition that we developed and presented in the recent past (Livathinos et al. 2021; Pfitzmann et al. 2022; Lysak et al. 2023). Docling is designed as a simple, self-contained Python library with permissive MIT license, running entirely locally on commodity hardware. Its code architecture allows for easy extensibility and addition of new features and models. Since its launch in July 2024, Docling has attracted considerable attention in the AI developer community and ranks top on GitHub's monthly trending repositories with more than 10,000 stars at the time of writing. On October 16, 2024, Docling reached a major milestone with version 2, introducing several new features and concepts, which we outline in this updated technical report, along with details on its architecture, conversion speed benchmarks, and comparisons to other open-source assets.

The following list summarizes the features currently available on Docling:

- Parses common document formats (PDF, Images, MS Office formats, HTML) and exports to Markdown, JSON, and HTML.
- Applies advanced AI for document understanding, including detailed page layout, OCR, reading order, figure extraction, and table structure recognition.
- Establishes a unified `DoclingDocument` data model for rich document representation and operations.
- Provides fully local execution capabilities making it suitable for sensitive data and air-gapped environments.
- Has an ecosystem of plug-and-play integrations with prominent generative AI development frameworks, including LangChain and LlamaIndex.
- Can leverage hardware accelerators such as GPUs.

## 2 State of the Art

Document conversion is a well-established field with numerous solutions already available on the market. These solutions can be categorized along several key dimensions, including open vs. closed source, permissive vs. restrictive licensing, Web APIs vs. local code deployment, susceptibility

*These authors contributed equally.

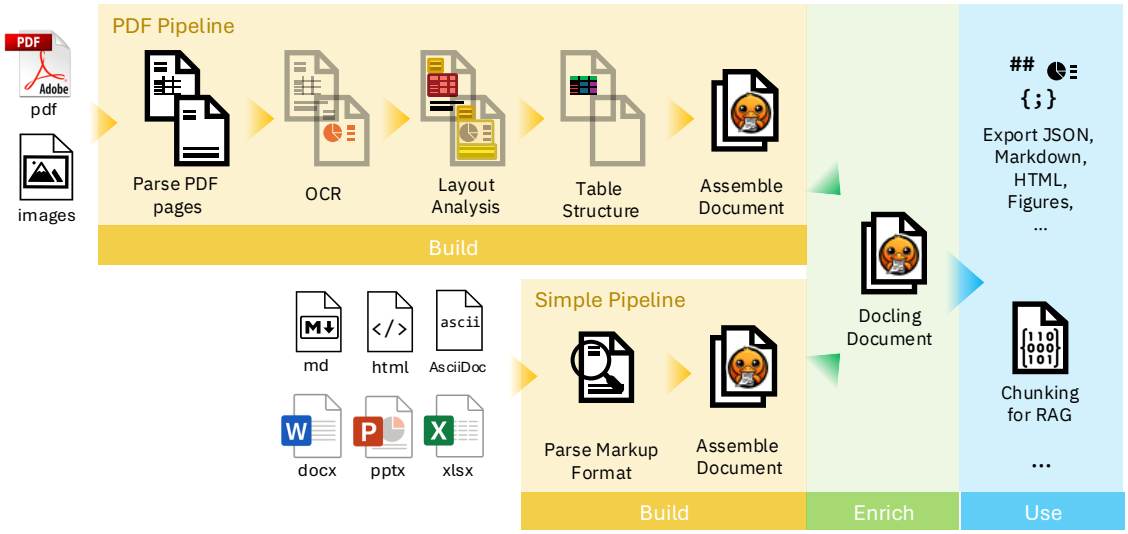


Figure 1: Sketch of Docling's pipelines and usage model. Both PDF pipeline and simple pipeline build up a `DoclingDocument` representation, which can be further enriched. Downstream applications can utilize Docling's API to inspect, export, or chunk the document for various purposes.

to hallucinations, conversion quality, time-to-solution, and compute resource requirements.

The most popular conversion tools today leverage vision-language models (VLMs), which process page images to text and markup directly. Among proprietary solutions, prominent examples include GPT-4o (OpenAI), Claude (Anthropic), and Gemini (Google). In the open-source domain, LLaVA-based models, such as LLaVA-next, are noteworthy. However, all generative AI-based models face two significant challenges. First, they are prone to hallucinations, i.e., their output may contain false information which is not present in the source document — a critical issue when faithful transcription of document content is required. Second, these models demand substantial computational resources, making the conversion process expensive. Consequently, VLM-based tools are typically offered as SaaS, with compute-intensive operations performed remotely in the cloud.

A second category of solutions prioritizes on-premises deployment, either as Web APIs or as libraries. Examples include Adobe Acrobat, Grobid, Marker, MinerU, Unstructured, and others. These solutions often rely on multiple specialized models, such as OCR, layout analysis, and table recognition models. Docling falls into this category, leveraging modular, task-specific models which recover document structures and features only. All text content is taken from the programmatic PDF or transcribed through OCR methods. This design ensures faithful conversion, without the risk of generating false content. However, it necessitates maintaining a diverse set of models for different document components, such as formulas or figures.

Within this category, Docling distinguishes itself through its permissive MIT license, allowing organizations to integrate Docling into their solutions without incurring licensing fees or adopting restrictive licenses (e.g., GPL). Additionally, Docling offers highly accurate, resource-efficient, and fast models, making it well-suited for integration with many standard frameworks.

In summary, Docling stands out as a cost-effective, accurate and transparent open-source library with a permissive license, offering a reliable and flexible solution for document conversion.

## 3 Design and Architecture

Docling is designed in a modular fashion with extensibility in mind, and it builds on three main concepts: pipelines, parser backends, and the `DoclingDocument` data model as its centerpiece (see Figure 1). Pipelines and parser backends share the responsibility of constructing and enriching a `DoclingDocument` representation from any supported input format. The `DoclingDocument` data model with its APIs enable inspection, export, and downstream processing for various applications, such as RAG.

### 3.1 Docling Document

Docling v2 introduces a unified document representation, `DoclingDocument`, as a Pydantic data model that can express various common document features, such as:

- Text, Tables, Pictures, Captions, Lists, and more.
- Document hierarchy with sections and groups.
- Disambiguation between main body and headers, footers (furniture).
- Layout information (i.e., bounding boxes) for all items, if available.
- Provenance information (i.e., page numbers, document origin).