# Haystack Overview

## Definition

Haystack is an open-source framework by deepset designed to build production-ready NLP systems based on Large Language Models (LLMs). It is primarily used for creating search systems, question answering (QA), semantic retrieval, and RAG (Retrieval-Augmented Generation) pipelines.

## Framework Architecture

Haystack architecture is modular and pipeline-driven, allowing integration of retrievers, readers, generators, rankers, and custom components. It supports multi-modal document stores and scalable backends.

Key Architecture Components:

-Document Store – Stores and indexes documents (Elasticsearch, FAISS, Weaviate, etc.)  
-Retriever – Pulls relevant documents based on query (sparse/dense)  
-Reader – Extractive QA model that finds precise answers from retrieved docs  
-Generator – Generative model for abstractive answers (e.g., OpenAI, Cohere)  
-Ranker – Ranks retrieved docs based on relevance  
-Pipeline – Orchestrates components in a customizable flow  
-Node – Each processing step in a pipeline

## Core Components

| Component | Description |
| --- | --- |
| DocumentStore | Backend to store documents and embeddings. |
| Retriever | Finds documents relevant to a query (BM25, DPR, etc.). |
| Reader | Extracts the answer span from the text (e.g., BERT, RoBERTa). |
| Generator | Generates answers using language models (e.g., GPT, FLAN-T5). |
| Ranker | Reorders retrieved documents based on semantic relevance. |
| Pipeline | Workflow of connected nodes (Retriever → Reader, etc.). |
| Node | Atomic operation step (can be custom-built). |
| PromptNode | Allows use of prompt-based LLMs for diverse tasks. |

## Pros of Haystack

• Modular Design: Easily build custom pipelines using modular components.

• Multi-Backend Support: Works with Elasticsearch, OpenSearch, FAISS, Weaviate, etc.

• Hybrid Search: Supports both sparse (BM25) and dense retrieval (DPR, SentenceTransformers).

• Flexible Pipelines: Pipeline-based design enables complex workflows.

• Generative + Extractive: Supports both extractive readers and generative LLMs.

• Community and Extensibility: Active open-source community and custom node support.

## Cons of Haystack

• Setup Complexity: Requires understanding of multiple components and dependencies.

• Performance Tuning: Hybrid pipelines may need careful tuning for latency and precision.

• Resource Intensive: Running dense retrievers or generators can be compute-heavy.

• Documentation Gaps: Advanced configurations sometimes lack detailed guidance.

## Alternatives & Similar Tools

• LangChain: Framework for LLM applications, strong tool integration and agent-based workflows.

• RAG by Hugging Face: Pre-built models and utilities for retrieval-augmented generation.

• LlamaIndex: Data framework to connect LLMs with private data; simple interface for document ingestion.

• Semantic Kernel: Microsoft’s orchestration SDK for prompt engineering and semantic pipelines.

• DeepEval: Evaluation tool for RAG pipelines and QA systems.

• Qdrant: Vector DB with built-in semantic search and filtering.

## Summary

| Aspect | Summary |
| --- | --- |
| Use Case | RAG, semantic QA, enterprise search, LLM apps |
| Language | Python |
| Execution Model | Pipeline-based processing |
| Deployment | Docker, Kubernetes, REST API, local |
| Strength | Modular, LLM-ready, backend-agnostic |
| Weakness | Complexity in setup and resource use |