Practical Guide: Chunking in Data Intake

# 1) Core Building Blocks

Tokenization / sizing (don’t guess—measure):  
- tiktoken (OpenAI token counts) or transformers tokenizers (BERT/T5/GPT-2).  
- Why: chunk sizes should be in tokens, not characters.  
  
Chunkers (production-friendly):  
- langchain-text-splitters: RecursiveCharacterTextSplitter, TokenTextSplitter, MarkdownHeaderTextSplitter, HTMLHeaderTextSplitter.  
- llama-index “Node Parsers”: good structural splitters for PDFs/HTML/Markdown with metadata inheritance.  
- Sentence segmentation for semantic boundaries: spacy (robust) or pysbd (rule-based, fast).  
  
Evaluation for RAG/retrieval:  
- OpenSearch/Elastic Ranking Evaluation API (/\_rank\_eval) → nDCG, recall.  
- ragas (QA-style RAG eval: context precision/recall, faithfulness).  
- trec\_eval mindset: Recall@k, MRR, nDCG@k.  
- Observability while tuning: TruLens, Arize Phoenix, or LangSmith.  
  
Vector + keyword backends:  
- OpenSearch k-NN / Elastic vectors (hybrid = BM25 + ANN).  
- FAISS if you want local ANN while you tune (fast experiments).

# 2) Extractors by Data Type

|  |  |  |
| --- | --- | --- |
| Data type | Primary extractor | Notes on boundaries |
| PDF | pymupdf, pdfminer.six, unstructured | Pages → headings → paragraphs. Keep page#, section. |
| Office (docx, pptx, xlsx) | python-docx, python-pptx, openpyxl, unstructured | Word: headings/paragraphs; Slides: per-shape text with title. |
| HTML | trafilatura, readabilipy, unstructured | Use H1–H6/ul/ol/p; preserve anchors/ids. |
| Markdown/ReST | markdown-it-py, langchain MD header splitter | Split by headers and lists; good metadata. |
| JSON/Logs | ijson, pandas | Chunk by record (object/line); logs by time window or N-lines. |
| Email | mailparser | Chunk by header, body, quoted blocks, attachments. |
| Audio/Video | whisperx, faster-whisper | Chunk by speaker turn or time windows + overlap. |
| Images/PDF scans | pytesseract, paddleocr + layoutparser | Chunk by detected blocks (paragraph/region). |

# 3) Practical Defaults That Work

- BM25 (keyword): paragraph/section chunks, no overlap, 200–1,000 words.  
- Vectors (RAG): token-based 300–700 tokens with 10–20% overlap. Keep titles and breadcrumbs in metadata.  
- Transcripts: per speaker turn; long turns re-chunked to 400–600 tokens with overlap.  
- Code/configs: function-level or file sections; avoid splitting signatures/blocks.

# 4) Evaluation

- Use FAISS or OpenSearch to test different chunk sizes.  
- Metrics: Recall@k, nDCG@k, MRR.  
- For BM25/Hybrid in OpenSearch: use Ranking Evaluation API (\_rank\_eval).

# 5) Corpus-specific Recipes

- Contracts/policies: 400–600 tokens, 15% overlap, boundaries at headings/clauses. Keep section\_path metadata.  
- Wikis/Docs: header-aware splitting → 350–550 tokens, 10–15% overlap.  
- Transcripts: per speaker turn; if >700 tokens, split to 400–600 tokens with 50–100 token overlap; store speaker, timestamp.  
- Logs/JSON: chunk by record/time window (1–5 minutes per chunk). No overlap.  
- HTML: clean boilerplate, chunk by DOM headers then token windows.

# 6) Installation Checklist

Extractors: pymupdf, pdfminer.six, unstructured, trafilatura, python-docx, python-pptx, openpyxl, pytesseract, paddleocr  
Chunking + NLP: spacy, pysbd, tiktoken, langchain-text-splitters, llama-index  
Embeddings + ANN: sentence-transformers, faiss-cpu  
Backends: opensearch-py, elasticsearch  
Evaluation: ragas, trulens, arize-phoenix

# 7) Key Takeaways

- No universal best chunk size; always grid-search size × overlap and measure recall/nDCG on real queries.  
- Use tokens, not characters, for chunk sizes.  
- Overlap improves RAG but increases cost; cap at 10–20%.  
- Metadata (titles, headings, speaker, timestamp) is cheap and critical for hybrid retrieval.