

Language and core environment

- Python 3.10+
- Conda or venv, requirements.txt (or poetry)
- Jupyter notebooks for experiments, then a src/ package for final code
- Git + GitHub

Data acquisition (international law + human rights)

- OpenAlex (metadata, open access links)
- DOAJ (Directory of Open Access Journals) for open access law journals
- Optional: Crossref for DOI (Digital Object Identifier) metadata completion
- Storage format: JSONL (JSON Lines) for paper records + CSV logs for downloads

PDF ingestion and parsing

- PyMuPDF (fitz) or pdfminer.six for PDF (Portable Document Format) to text
- python-magic or filetype for file validation
- regex for section heuristics and reference heuristics
- Output artifacts:
 - papers.jsonl (paper-level metadata)
 - chunks.jsonl (chunk_id, paper_id, text, offsets, section)

Text processing

- nltk (tokenization) or spaCy (optional, only if needed for better tokenization)
- simple normalization: lowercasing, punctuation rules, stopword list

Keyword retrieval (baseline)

- BM25 (Best Matching 25): rank-bm25 (Python) or a simple in-house BM25
- Optional TF-IDF (Term Frequency–Inverse Document Frequency) baseline: scikit-learn

Semantic retrieval (Word2Vec approach)

- gensim Word2Vec (pretrained GoogleNews vectors or train on your corpus)
- Document or chunk vectors via averaging Word2Vec token vectors
- Similarity: cosine similarity (NumPy)

Vector index (optional acceleration)

- FAISS (Facebook AI Similarity Search) for ANN (Approximate Nearest Neighbor) search
- Fallback (small corpus): brute-force cosine similarity with NumPy

Metadata index and filtering

- SQLite (paper_id, title, authors, year, venue, doi, url, topic tags)
- Optional inverted index for metadata fields: simple Python dicts persisted to JSON
- Pandas for cleaning, deduplication, and stats

Hybrid ranking (BM25 + Word2Vec)

- Score fusion: weighted sum or rank fusion (for example Reciprocal Rank Fusion (RRF))
- Deduplication: DOI match + normalized title-year match

Reranking (optional, if you implement it)

- Lightweight: heuristic rerank using term coverage + entity overlap
- Stronger (if allowed): a cross-encoder reranker from sentence-transformers (requires using a transformer, beyond Word2Vec)

Citation formatting

- bibtextparser or pybtex for BibTeX generation
- Custom formatter for APA (American Psychological Association) and IEEE (Institute of Electrical and Electronics Engineers) in-text strings from stored metadata
- Validation: unit tests that fields match metadata

Backend service (for integration)

- FastAPI (recommended) or Flask for an API (Application Programming Interface)
- Endpoints:
 - /suggest_citations (paragraph → top-k papers + snippets + formatted citations)
 - /health (sanity check)

User interface options

- CLI (Command Line Interface): argparse or typer
- Web demo: Streamlit (fastest) or a simple FastAPI + HTML frontend

Google Docs integration (best for “insert citations”)

- Google Docs Add-on using Google Apps Script
- Sidebar UI (HTML/JS)
- Document editing via DocumentApp methods (cursor insertion, replace selection, append references)

Chrome extension (optional)

- Manifest V3 (MV3)
- Popup UI (HTML/JS)
- Calls your FastAPI endpoint

- For Google Docs: “copy citation to clipboard” is reliable; direct insertion is not reliable compared to a Docs Add-on

Evaluation

- Metrics: Recall@K, Mean Reciprocal Rank (MRR), normalized Discounted Cumulative Gain (nDCG@K)
- Latency: p50 (50th percentile) and p95 (95th percentile) response time
- Human rubric sheet (relevance, correctness, usefulness)
- Testing: pytest (unit/integration/regression)

Deployment (if needed)

- Local: Docker (optional)
- Hosted API: Render / Fly.io / Google Cloud Run (optional)
- Caching: local disk cache (JSON + SQLite) for repeat queries and embeddings