CSC490 Assignment 2 - Data Processing Pipelines

Team:
Son Nguyen (ID 1009656560), Kyle (ID 1007785229),
Daniel (ID 1008035378), Jinbo (ID 1004821419)
Project: ArXplorer - Semantic Search for arXiv

Part One: Aspirational Datasets

The ideal datasets for creating ArXplorer would include:

Full-text papers from arXiv (LaTeX + PDF + metadata)

Paper(paper_id, title, abstract, full_text, publish_date, categories)

Why ideal: Provides the rich semantic context for embeddings and search.

Paper authorship and institution data

Institution(institution_id, name, country, aliases)

Author(author_id, full_name, institution_id, h_index)

PaperAuthor(paper_id, author_id, author_position)

Why ideal: Enables exploration of academic networks and author disambiguation, also useful for integrating author influence.

Citation network data

Citation(paper_id, cited_paper, citation_context)

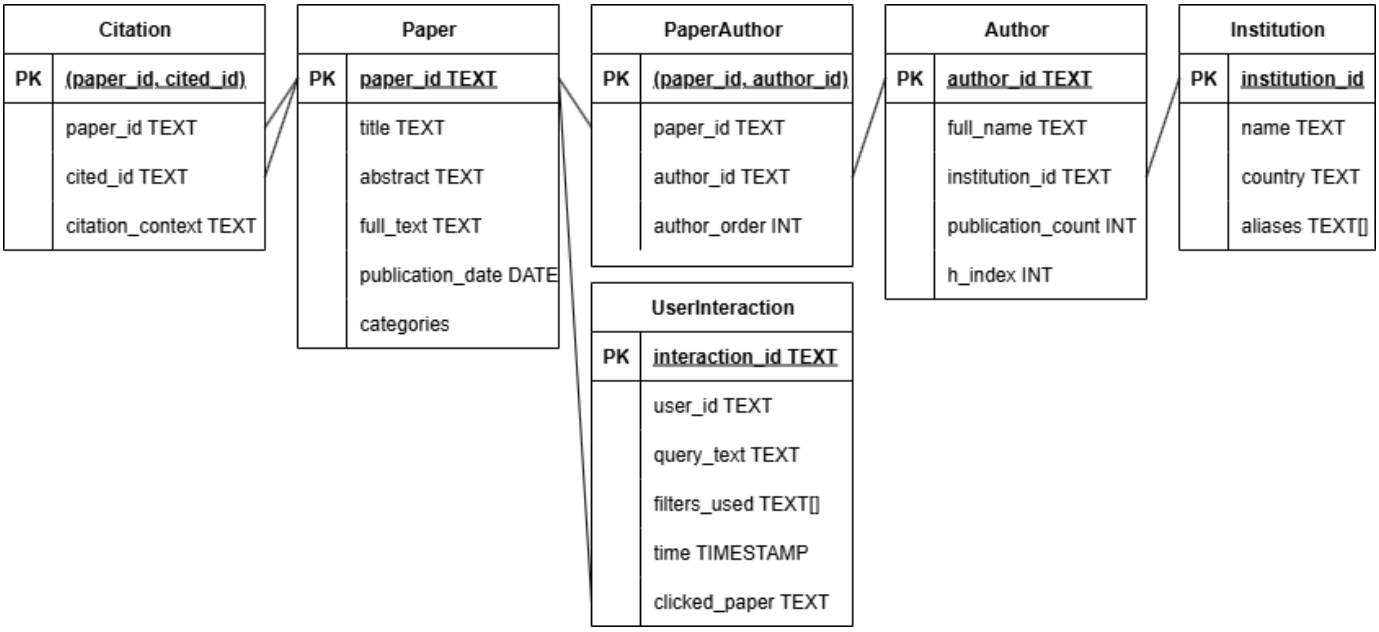
Why ideal: Enhances ranking algorithms by integrating influence and citation context.

User interaction logs

UserInteraction(interaction_id, user_id, query_text, filters_used, time, clicked_paper)

Why ideal: Useful for learning-to-rank models and improving recommendation quality.

Ideal Data Schema Diagram



Part Two: Reality Check

Since full proprietary datasets are hard to access, here are realistic datasets we can use:

Relevant Item	Description	Commentary
Kaggle: arXiv Dataset	Paper metadata dataset of arXiv papers (1.7M entries). Includes arXiv IDs, titles, authors, categories, submission dates, and abstracts.	Convenient starting point for prototyping data pipeline and generating semantic text embeddings of paper metadata.
arXiv Metadata (public OAI API)	Official Open Archives Initiative API. Allows retrieval of arXiv metadata such as paper IDs, titles, authors, abstracts, categories, submission/revision history.	The API is freely available. Allows up-to-date retrieval of paper metadata; unlike Kaggle dataset, allows continuous ingestion of newly published papers. Enables incremental embedding pipelines and real-time updates.
Semantic Scholar Open Research Corpus (S2ORC)	Large-scale dataset with 136M+ academic papers. Includes titles, abstracts, full text (for some), and detailed citation networks.	Augments arXiv data with richer context such as citation graphs. Useful for building hybrid ranking models (semantic relevance + citation authority) and for exploring related-paper discovery beyond arXiv.
FAISS Benchmarks & Sample Datasets	FAISS provides tools and benchmark datasets for large-scale dense vector search. Includes example corpora and performance comparisons of different indexing methods.	Relevant for testing retrieval quality during development. Benchmarks will inform trade-offs between recall, latency, and memory.

Part Three: Data-Processing Pipelines

Data Schema

Application Layer Data Schema:

- Author(name : str, affiliation : Optional[str], email : Optional[str])
- ArXivPaper(arxiv_id : str, title : str, abstract : str, authors : List[Author], categories : List[str], submitted_dates : datetime, updated_date: Optional[datetime], doi: Optional[str], journal_ref: Optional[str], comments: Optional[str])
- ProcessedPaper(arxiv_id: str, cleaned_title: str, cleaned_abstract: str, extracted_keywords: List[str], citation_count: int, references: List[str], full_text_url: Optional[str], word_count: int, language: str, readability_score: Optional[float])
- PaperEmbedding(arxiv_id: str, title_embedding: List[float], abstract_embedding: List[float], combined_embedding: List[float], model_name: str, model_version: str, embedding_dimension: int, created_at: datetime)
- SearchQuery(query_id: str, raw_query: str, processed_query: str, query_embedding: List[float], filters: Dict[str, Any], user_id: Optional[str], timestamp: datetime)
- SearchResult(arxiv_id: str, relevance_score: float, similarity_score: float, boost_factors: Dict[str, float])

Storage Layer Data Schema:

- papers(<u>arxiv_id</u>, title, abstract, authors JSON, categories JSON, submitted_dates, updated_date, doi, journal_ref, comments, status, processed_at, error_message, created_at, updated_at)
- processed_papers(<u>arxiv_id</u>, cleaned_title, cleaned_abstract, extracted_keywords, citation_count, references, full_text_url, word_count, language, readability_score, created_at)
- embeddings(<u>arxiv_id</u>, title_embedding, abstract_embedding, combined_embedding, model_name, model_version, embedding_dimension, created_at)
- search_queries(<u>query_id</u>, raw_query, processed_query, query_embedding, filters, user_id, timestamp)
- Foreign Key Constraints
  - processed_papers(arxiv_id)$\subseteq$ papers(arxiv_id)
  - embeddings(arxiv_id) $\subseteq$ papers(arxiv_id)

Pipeline Description

→ **Data Ingestion (kaggle_loader.py)**

- Load paper in JSON format from kaggle arxiv dataset.
- Parse authors, categories, and dates into standard format.

→ **Data Cleaning (pipeline.py::TextProcessor)**

- Regex normalization for titles and abstracts.
- Tokenize text (NLTK) and compute word counts.
- Compute TF-IDF top-K keywords per paper (scikit-learn).
- Compute simple readability metric per paper (number of words).

→ **Embedding Generation (pipeline.py::EmbeddingGenerator)**

- Compute embeddings (‘allenai/scibert_scivocab_uncased‘ via transformers + PyTorch)

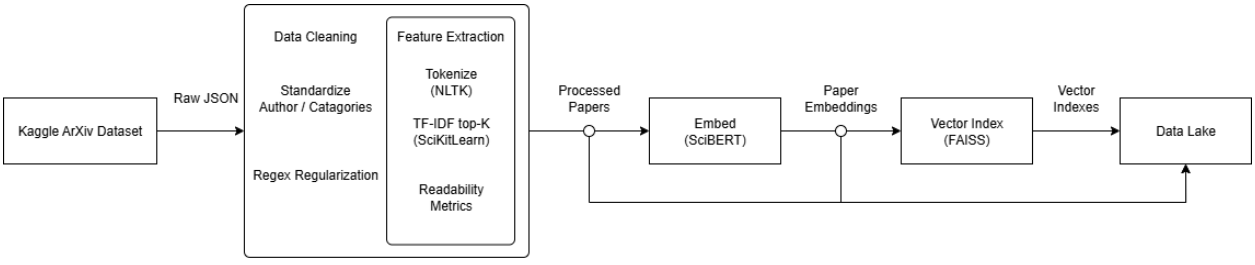
→ **Vector Index (pipeline.py::VectorIndexer)**

- Index paper embeddings using FAISS for fast similairty search.
- Inner-product similarity (IndexFlatIP) or IVF (IndexIVFFlat) per config.

→ **Data Lake (and Data Warehouse):** output_dir/

- processed/ processed_papers_<timestamp>.json + raw_papers_<timestamp>.json
- embeddings/ embeddings_batch_<batch>_<timestamp>.json
- index/ papers_index_<timestamp>.index (+ <index>_ids.json saved by indexer)
- stats/ dataset_analysis.json, processing_summary.json
- SQL DDL strings available for relational database storage (schemas.py)

Pipeline Diagram



Pipeline Use Cases

The data processing pipeline is designed to run on-demand using CLI commands. The pipeline processes the data in a batch mode, offering several execution modes controlled by user-defined flags:

Analyze Only: Generates basic dataset statistics (e.g., category distributions, paper counts) without processing the data further. Used for dataset exploration and quality checks.

Process Papers: Cleans and extracts features like TF-IDF keywords, word counts, and readability scores. Suitable for preparing the data for research corpus creation or feature extraction.

Generate Embeddings: Converts cleaned titles and abstracts into embeddings using SciBERT, enabling semantic similarity tasks like search and clustering.

Build FAISS Index: Builds a vector index for fast similarity searches on the generated embeddings. Essential for enabling semantic search and paper recommendation systems.

Overall, the pipeline will be used to generate the vector database for semantic search. It will create the vector indexes, which can then be queried with different search queries to retrieve relevant papers based on their semantic content.

Next Steps

- Explore citation data (if added later).
- Scale up embeddings or test advanced transformer variants.
- Expand front-end features for filtering and exploration.
- Explore real-time data ingestion with arXiv API (partially implemented).