DuckDB Applications in R

February 2nd, 2024
Prof.dr.ir. Arjen P. de Vries
arjen@cs.ru.nl

Radboud University

EFFICIENCY AND SCALABILITY

- IR builds efficient and scalable systems on top of the traditional "ubiquitous" inverted file implementation and/or specialized data structures like the HNSW. The algorithms that make retrieval efficient are "hard-coded" (where most systems today depend on Lucene and FAISS) –

  no data independence

- This is not a problem if all you do is ad-hoc search using a predefined ranking algorithm (e.g., BM25) over a predefined, static collection

  However…

- What if "ranking text" is a core component in a wide variety of tasks, as we see today? Over varying collections, dynamically updating?
- Even in "just" IR experimentation, complexity increases – entities/knowledge bases, whole session retrieval, multi-stage ranking systems using large collections of embeddings, inference using transformers ("neural IR")
- Models (like "Wikipedia") become stale due to data changes over time ("distribution shift")
  - "Wikipedia" becomes "Wikipedia-2014", "Wikipedia-2019", but… ***never*** "Wikipedia-today"

R on a Relational DBMS: “Olddog”

Mühleisen et al., Old Dogs Are Great at New Tricks: Column Stores for R Prototyping (2014)

See also:

- **chriskamphuis.com/2019/03/06/teaching-an-old-dog-a-new-trick.html**
- **github.com/osirrc/olddog-docker**

In DuckDB:

```
PRAGMA create_fts_index
```

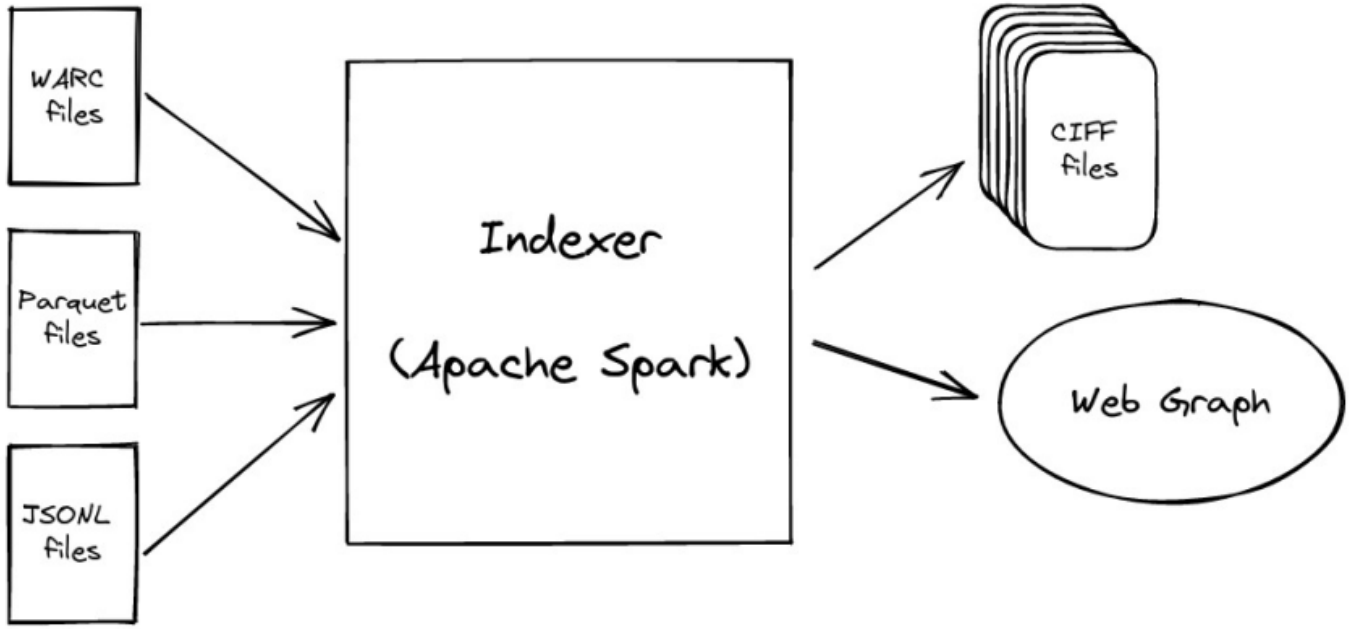
The OWS.EU Indexer

→ Build an Index and a Web Graph from a Crawl

→ Inputs:

- WARC (Crawl output)
- Parquet (Cleaning pipeline output)
- JSONL (IR datasets in TIRA)

→ Partitioned on metadata like language, date, “category”, *etc.*



→ Output as CIFF files, and Parquet files for nodes and edges of the Web Graph

https://opencode.it4i.eu/openwebsearcheu-public/spark-indexer/
… but… active development in https://opencode.it4i.eu/openwebsearcheu/wp3/indexing/spark-indexer/

“JUST” AN EXAMPLE INDEXING JOB

- Indexed the CommonCrawl (April 2021) on our local Spark cluster
  - 70TB in WARCs, 1.4TB after extracting clean text and links
- Index size: 467GB
  - Total of 220M documents
  - Partitioned into 50,000 CIFF files
- Web graph size: ~62GB
  - Includes the **anchor text**
- Elapsed time: 106h
  - 55h (just over 50%) for parsing WARC files and extracting co

Anchor Text
is the secret door to
Web Search
Effectiveness!

Feel free to forget everything you
heard about PageRank

DuckDB helps organize the Parquet data smoothly, except for one little thing…
https://github.com/duckdb/duckdb/discussions/10346#discussioncomment-8258298

```
Can't combine FILE_SIZE_BYTES and PARTITION_BY for COPY
```

ONGOING WORK: LOADING CIFF INDICES IN DUCKDB

- CIFF uses Protobuf to represent the inverted file;
  storing the exact same information that the FTS extension would also create when it builds an index
- Goal:
  - Use DuckDB for the Parquet metadata *and* the inverted file
- Approach:
  - Use Google's Protobuf to read the index
  - Use Python to hand out term identifiers and undo gap compression
  - Use PyArrow to ingest the data into DuckDB

- Status:
  - DuckDB database 6x as large
  - Does not automatically detect the natural opportunity to use PFOR-Delta for the postings

https://github.com/arjenpdevries/CIFF2DuckDB

MOVE PYTHON TRANSFORMATION TO SQL?

- Getting document identifiers from the gap-encoded ones:

```
select rowid, termid, docid, docid - lag(docid,1,0)
over (partition by termid order by rowid)
from postings
order by rowid;
```

- Back to gap-encoded document identifiers:

```
select rowid, termid, docid, sum(docid)
over (partition by termid order by rowid)
from postings
order by rowid;
```

- Beautiful SQL!
  - But… no effect on the selection of the compression algorithm (yet)

GEESEDB: A PYTHON GRAPH ENGINE FOR EXPLORATION AND SEARCH

- GeeseDB in an easy to install, self contained Python package. It contains topics and relevance judgements out-of-the-box (yet to do: integrate with MacAvaney et al.'s `ir_datasets`)
- First stage retrieval is directly supported out-of-the-box
  - Load your data and create a first stage ranker in only a few lines of code
- Data is served in a format suited for later retrieval stages (see "DuckDB slide" ahead)
- Easy data exploration is supported through both SQL and **a graph query language based on Cypher**

CYPHER VS. SQL

```
1  MATCH (d:docs)-[]-(:authors)-[]-(d2:docs)
2  WHERE d.collection_id = "96ab542e"
3  RETURN DISTINCT d2.collection_id
```

Figure 4: An example cypher query that finds all documents that were written by the same author that wrote the document with the collecion_id "96ab542e"

SIMILARITY SEARCH IN HIGH DIMENSIONS

Key operation in modern AI solutions

Occurs in many places in modern AI applications, whenever the input data is represented using “embeddings”

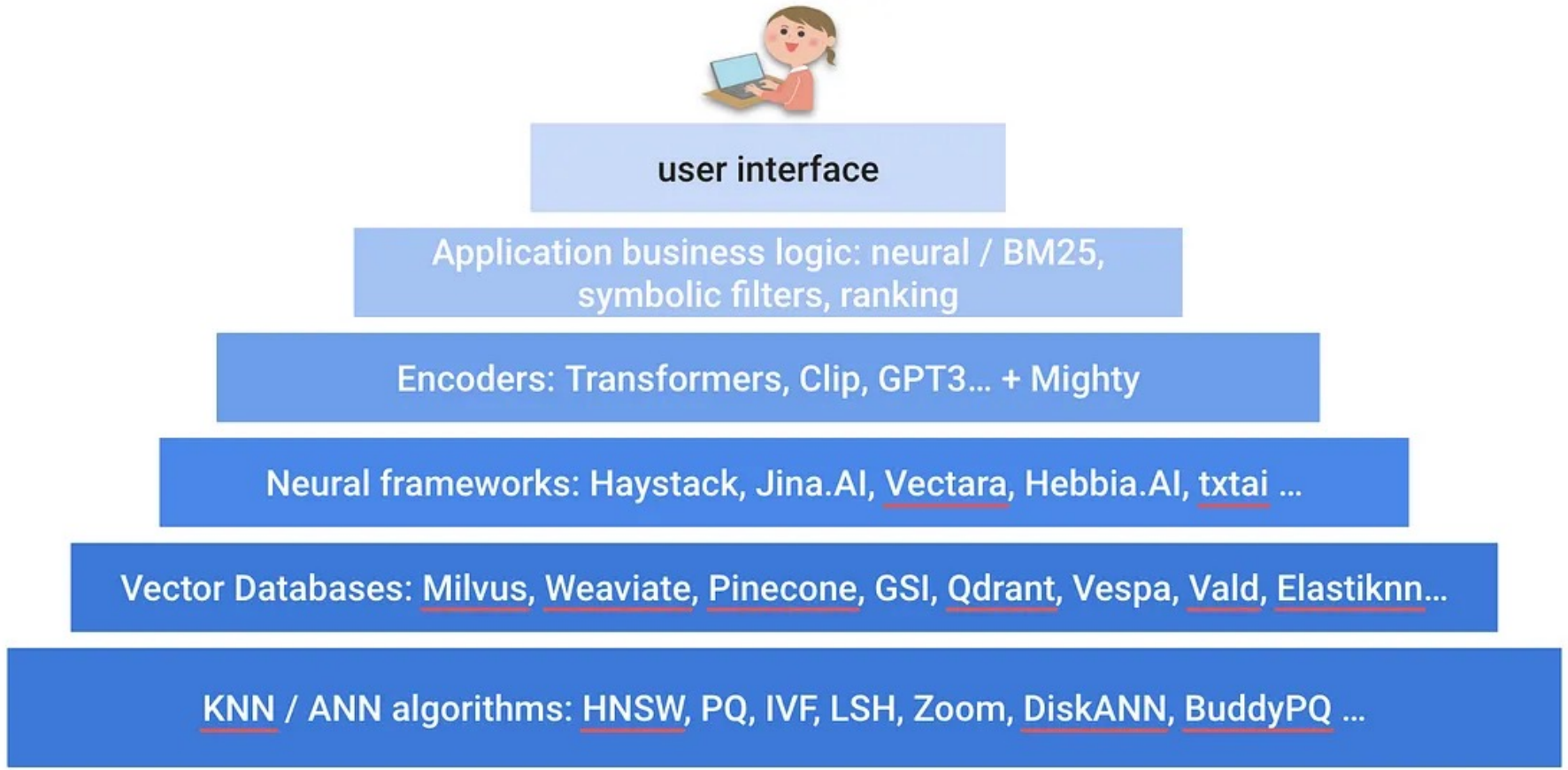
Embeddings can be constructed in many different ways – where the classic example is Word2Vec

Great way to learn about W2V: WEVI: Word Embedding Visual Inspector

Context-aware embeddings are drawn from large language models like BERT

https://dmitry-kan.medium.com/neural-search-frameworks-a-head-to-head-comparison-976aa6662d20

L□BRAR□ES, VECTOR DATABASES, FRAMEWORKS, ...



VECTOR DATABASES FUTURE?

- Personal prediction:
  - Eventually, vector databases will be “swallowed” by general purpose database solutions
  - Unclear however what the data model will be! Will relational win again?
- Real solution:
  - Revive and upgrade BOND (Efficient k-NN Search on Vertically Decomposed Data, SIGMOD 2002)
- Quicker "solution":
  - Run a FAISS index together with a DuckDB database
  - Use an extension couple the embeddings-in-the-index together with the data-in-the-database
  - FAISS limiting HNSW scope using `IDselectorBatch/Array/Bitmap` from DuckDB

In development, repo not yet public... but coming real soon!
https://github.com/arjenpdevries/faiss

ENTITY LINKING

MMEAD

- MMEAD is a specification for entity links for MSMARCO
  - JSON specification for sharing and using entity links
  - Pretrained Wikipedia2Vec embeddings
  - Python library to use both resources (entity links and embeddings) easily

- MS Marco V1 and V2, passage and doc, tagged with entity annotations
  - RE(B)L
  - BLINK *(only V1 passage completed right now)*

- Ready to use!

DuckDB

Chris Kamphuis, Aileen Lin, Siwen Yang, Jimmy Lin, Arjen de Vries and Faegheh Hasibi. MMEAD: MS MARCO Entity Annotations and Disambiguations. In SIGIR 2023.
https://github.com/informagi/MMEAD