ArXiv Paper Crawling with Neo4j Storage

Matthieu NEAU, Lucas HENNEÇON, Solomon HARVEY

Data Acquisition, Extraction and Storage
Paris Dauphine University

December 13, 2014

What Are We Building?

- **Problem:** Research papers reference other papers. Tracking these connections can reveal valuable insights.
- **Objective:** Build a system that crawls through arXiv papers, extracts their references and metadata, and builds a connected graph of citations and metadata stored in a Neo4j database.

Architecture

1. **PDF Processor**
   - Takes arXiv URL as input
   - Outputs full text from PDF of arXiv article at given URL
2. **Reference Extractor**
   - Takes a paper's text as input
   - Outputs ArXiv IDs of referenced papers
3. **Metadata Extractor**
   - Takes arXiv paper ID as input
   - Outputs paper metadata (title, authors, index, publication year)
4. **Data Pipeline**
   - Connects to a Neo4j database
   - Saves paper metadata and citation relationships to the database

Architecture (Continued)

5. **Recursive arXiv Crawler**
    - Starts with an initial arXiv paper ID
    - Recursively crawls references up to a certain depth
    - Connects to data pipeline and sends it metadata and citation relationships
6. **PapersWithCode Method Crawler**
    - Starts with a method name from PapersWithCode
    - Extracts arXiv IDs for the papers within this method
    - Crawls references from each paper
    - Connects to data pipeline and sends it metadata and citation relationships
7. **Neo4j Graph Database**
    - Stores papers as nodes, metadata in nodes, and citation relationships as edges
    - Yields a connected graph of arXiv papers

PDF Processor

Algorithm 1 PDF Processor

```
1: **Input:** arXiv PDF URL
2: Send an HTTP GET request to the specified URL
3: Store the PDF content in an in-memory binary stream
4: Open the binary stream using pdfplumber
5: Transform PDF content into single text string
6: **Output:** PDF content as single text string
```

Reference Extractor

Algorithm 2 Reference Extractor

```
1: Input: Article text as a single string
2: Use a regular expression to find all occurrences of the pattern
   \(?:arXiv:|abs/\)(\d{4}\.\d{4,5})
3: Extract the reference codes by splitting each match on the colon (:)
   and keeping the second part
4: Store the extracted reference codes in a list
5: Output: List of arXiv reference codes
```

Metadata Extractor

Algorithm 3 Metadata Extractor

```
 1: Input: ArXiv query (e.g., id:1805.08355)
 2: Send a GET request to the arXiv API with the query
 3: if request fails or times out then
 4:     Retry up to 3 times with exponential backoff
 5: end if
 6: Parse the XML response
 7: if no valid entry is found then
 8:     Return default metadata (e.g., "Unknown Title")
 9: else
10:     Extract metadata fields: title, authors, publication date, link
11: end if
12: Output: Metadata as a dictionary
```

Data Pipeline

- **Input:** Metadata of papers (title, authors, index, publication year) and citation relationships
- **Pipeline:**
  1. Connect to the Neo4j database using the provided URI and credentials
  2. Add paper metadata to the database
     - Use `MERGE` to ensure uniqueness based on the paper index
     - Set attributes: title, authors, index, publication year
  3. Add citation relationships
     - Use `MATCH` to find two papers and `MERGE` a `CITES` relationship between them
- **Output:** Papers, their metadata, and citation relationships saved in the Neo4j database

Recursive arXiv Crawler

Algorithm 4 Recursive arXiv Crawler

```
1:  Input: Starting article ID, maximum depth
2:  procedure CrawlArticle(article ID, depth)
3:      if depth exceeds maximum depth or article is visited then
4:          Return
5:      end if
6:      Fetch article metadata and save to database
7:      if depth is less than maximum depth then
8:          Extract references and save relationships to database
9:          for each reference do
10:             Recursively call CrawlArticle(reference, depth + 1)
11:         end for
12:     end if
13: end procedure
14: Output: Metadata and relationships in Neo4j
```

PapersWithCode Method Crawler

Algorithm 5 PapersWithCode Method Crawler

```
1: Input: Method name
2: procedure COMPUTEMETHODGRAPH(method name)
3:     Retrieve method ID from the PapersWithCode website
4:     Fetch paper URLs from the PapersWithCode API
5:     Extract ArXiv IDs for the papers
6:     for each paper do
7:         Extract references from the paper's PDF
8:         Filter references to include only those in the list of ArXiv IDs
9:         Add references, metadata and citation relationships to database
10:    end for
11: end procedure
12: Output: A graph of papers and their citation relationships
```

Neo4j Graph Database

See demo!