Information Retrieval Project Report

Student: Farhan Ali Khan (40287970)
Course: COMP 479 (Information Retrieval)

Introduction:

The objective of this project was to design and implement an end-to-end information retrieval pipeline capable of crawling a real academic repository, extracting and indexing PDF documents, constructing an inverted index, designing queries, forming a custom subset of documents, and finally clustering the resulting document collection using K-Means. The assignment required crawling from Concordia University's Spectrum Open Access Portal, extracting the text of Master's and PhD theses, creating an inverted index, retrieving documents matching the queries *"sustainability"* and *"waste"*, merging the results into *My-collection*, and performing clustering for k = 2, 10, and 20 using scikit-learn.

The project also required respecting the robots.txt standard, implementing a limit on the number of documents downloaded, attributing all tools and sources used, and reporting on the behaviour of the different clusterings. The entire workflow—crawl → extract → index → query → cluster—represents a realistic IR pipeline.

Before starting, I watched the YouTube tutorial *"A Beginner's Guide to Building Web Crawlers using Scrapy"*(https://www.youtube.com/watch?v=mBoX_JCKZTE&t=1697s), which helped me understand the structure of Scrapy spiders, how callbacks work, and how to manage request depth and link filtering. This video was extremely helpful in getting comfortable with the Scrapy framework and understanding how to design a crawl that stays within domain limits and efficiently follows links.

2. Methodology

This section describes each component of the system: crawling, PDF extraction, indexing, query design, and clustering.

2.1 Web Crawling and Robots.txt Compliance

My crawler was implemented in Scrapy (`spectrumSpider.py`). The spider begins crawling at:

https://spectrum.library.concordia.ca/

restricted allowed domains to:

- `spectrum.library.concordia.ca`
- `library.concordia.ca`

This ensures that the crawler remains within the boundaries specified by the assignment and does not drift into external websites.

I verified the robots.txt file at:
`https://spectrum.library.concordia.ca/robots.txt`
and confirmed that crawling the relevant pages is allowed. My crawler respects robots.txt automatically through Scrapy's built-in middleware. I also added code to avoid fetching non-HTML resources unnecessarily and to avoid crawling duplicate links repeatedly.

2.2 Handling PDF Discovery and Crawl Limits

The assignment required implementing a maximum number of PDFs to download. I added a `limit` parameter to the spider's constructor:

```
def __init__(self, limit=300, *args, **kwargs):

    self.max_thesis_pdfs = int(limit)
```

I maintained two sets:

- `visited_pages` (avoids reprocessing HTML)
- `thesis_pdfs` (collected PDFs)

However, a major challenge early on was that the crawler did not stop automatically when the desired number of PDFs was reached. Scrapy's asynchronous execution continues scheduling new requests even after the limit is reached, so I had to implement an explicit check that raises `CloseSpider` as soon as the number of collected PDFs reaches the limit.

This solved the issue, and I successfully produced a dataset of 1000 PDFs, stored in `spectrum_pdfs_1000.json`. This became the foundation of the indexing process.

2.3 Extracting Text from PDFs

After collecting PDF URLs, I downloaded each PDF and extracted its text. For this, I reused the PDF extraction pipeline from Assignment 1, but modified it to handle:

1. PDFs with multiple encodings
2. PDFs containing images or scanned pages
3. Larger documents (90–200 pages)
4. Non-standard metadata

I used PyPDF2 along with regular expressions and fallback extraction logic.
The pipeline:

- streams bytes from the PDF
- extracts raw text
- normalizes whitespace
- lowercases and tokenizes
- strips punctuation and removes stopwords

Each document's token list was then inserted directly into the inverted index.
I avoided storing full PDF contents permanently, as required by the assignment.

2.4 Building the Inverted Index

The inverted index maps:

`term → list of document IDs`

Tokens were stemmed using Porter stemming, ensuring that related variations (e.g., *sustain*, *sustaining*, *sustainability*) fall under the same root term.

The index contains:

- **Unique terms: 92,44**
- **Total documents in index: 998**

The index was saved as `spectrum_index.json` for reuse.

My inverted index is only build on 998 pdfs not 1000 because for 2 pdfs access was denied

2.5 Designing Queries and Creating My-Collection

The assignment required retrieving documents matching two queries:

- **“sustainability”**
- **“waste”**

To cover different word variations, I used stem-based matching:

- **For sustainability:** term = `"sustain"`
- **For waste:** terms = `"wast"`, `"wastag"`, `"wasti"`, `"wasteney"`

Updated Statistics (from ~1000 crawled PDFs):

- **Sustainability documents: 134**
- **Waste documents: 43**
- **Intersection: 13**
- **Union: 164**

This union of **164 documents** formed *My-collection*, stored in **my_collection.json**.

2.6 TF–IDF Construction

I constructed a dense TF–IDF matrix where:

- **Rows = 164 documents**
- **Columns = 38,613 vocabulary terms** present in My-collection

TF representation used: **binary**
IDF was computed in two ways:

- **Global DF** (over all **998** crawled Spectrum documents)
- **Local DF** (over the **164-document** subset)

This supports the required global–local DF comparison.

2.7 K-Means Clustering

Using scikit-learn's `KMeans`, I ran clustering for:

- **k = 2**
- **k = 10**
- **k = 20**

For each cluster I output:

- cluster size
- top 50 TF-IDF terms
- example documents

These were generated through `experiments_clustering.py`.

3. Results and Analysis of Cluster Behaviour

This section summarizes the main observations.

3.1 k = 2: Very Broad Themes

With **k = 2**, the clustering naturally split into two major categories:

Cluster A — Sustainability / Social / Community (96 docs)

Top terms included:
`sustain, person, commun, social, engag, offer, question, servic, peopl, benefit, way, resourc`

Description:
Documents discussed sustainability, leadership, education, social development, community involvement, ethics, environmental practices, and reflective/qualitative work.
This reflects the dominant presence of sustainability-related material in My-collection.

Cluster B — Technical / Experimental / Scientific (68 docs)

Top terms included:
`ga, ratio, mm, temperatur, diamet, energi, paramet, variat, experiment, fig`

Description:
These were measurement-heavy scientific papers involving laboratory experiments,

ecology, chemical analysis, fluid behavior, temperature/diameter parameters, and other quantitative scientific domains.

Interpretation:
The separation was extremely stable in both DF modes.
Scientific discipline (social sciences vs. experimental/technical science) is the **primary organizing principle** of the full 164-document My-collection.

3.2 k = 10: Medium-Grained Topic Structure

With **k = 10**, more meaningful topic clusters emerged, such as:

- **Combustion / Detonation / High-pressure shock physics**
  (cluster containing terms like *deton, flame, kpa, stoichiometr, ignit*)
- **Neuroscience and psychopharmacology cluster**
  (*prefront, dopamin, hippocampu, cortex, neurosci*)
- **Ecology, biodiversity, and genetics cluster**
  (*habitat, genotyp, polymorph, microsatellit, plant*)
- **Rooftop airflow, CFD, and pollutant dispersion cluster**
  (*cfd, stathopoulo, turbul, rooftop, wind, aerodynam*)
- **Climate and emissions modeling cluster**
  (*ipcc, clim, aerosol, anthropogen*)
- **Economic, service, and sustainability-management cluster**
  (appearing strongly in Local DF)
- **Upcycling and product design cluster**
  (*upcycl, manufactur, materia*)
- **Narrative, education, experience-based social science cluster**
  (*stori, youth, experienc, feel*)

This medium granularity aligned well with academic subfields found in the 164-document set.

Under **local DF**, the sustainability clusters shifted somewhat toward societal, narrative, and educational vocabulary because terms like *social, group, student, particip, project, stori* became more informative within the smaller subset.

Highly technical clusters (combustion, neuroscience, CFD) remained **almost unchanged** under both DF modes.

3.3 k = 20: Fine-Grained and Sometimes Over-Specific

With **k = 20**, there were several coherent clusters, but many became very small (sizes 1–3).
Examples include:

- **18th-century literature / gender analysis cluster**
  (e.g., *Lady Mary* documents, prose/poetry analysis)
- **Neuroscience behavioural-rats / pharmacology cluster**
  (dopamine, prefrontal cortex, ventral striatum work)
- **PCR / biodiversity / molecular ecology cluster**
  (microsatellites, genotyping, ecology)
- **Librarianship and scholarly communication cluster**
  (terms like *librari, disciplinari, scholarship, profession*)
- **CFD / pollutant dispersion micro-clusters**
  (narrow wind-tunnel or rooftop emission papers)
- **Upcycling / manufacturing cluster**
  (remained stable even at high k)

While this level is over-fragmented because 20 is too large for 164 documents, it still captures the internal topical variety of My-collection.

3.4 Global vs Local DF (Bonus Comparison)

Global DF (998 documents):

- Emphasizes terms that are rare across the full Spectrum crawl.
- Strongly boosts **highly specialized scientific vocabulary**, such as:
  - combustion physics (`deton, kpa, combust, stoichiometr`)
  - neuroscience terms (`dopamin, hippocampu, prefront, cortex`)
  - CFD and dispersion terms (`cfd, stathopoulo, turbul, rooftop`)
- As a result, engineering and biology clusters become extremely sharp and stable.

Local DF (164 documents):

- Emphasizes terms rare only within My-collection.
- Shifts sustainability clusters toward:
  - **social** terms (`voic, stori, youth, particip`)
  - **economic/service** terms (`econom, cost, servic, busi`)
  - **education/narrative** terms (`learn, bring, feel`)
- Causes more fine-grained subgroups within sustainability topics.

In summary:

- **Global DF = emphasizes disciplinary rarity (more scientific weighting)**
- **Local DF = emphasizes internal distinctions within sustainability/social themes**

Technical clusters remained almost identical under both modes, demonstrating the dominance and uniqueness of scientific vocabulary.

Note:

User-Controlled Crawl Limit via runSpider.py

To make the crawling process easier to test and to ensure that the crawler always respects the assignment's requirement for a configurable upper bound on the number of PDFs, I created a small wrapper script called `runSpider.py`. When executed, the script prompts the user for how many PDFs they want to download (e.g., 50, 100, or 300), validates the input, and then launches the Scrapy spider with the corresponding `limit` parameter.

Resources and Acknowledgements

To complete this project, a variety of technical, academic, and online resources were consulted to better understand web crawling, PDF text extraction, inverted indexing, document-term matrix construction, and clustering methodology.

Primary Resources

- *Introduction to Information Retrieval* (Manning, Raghavan & Schütze, 2008) chapters on indexing, compression, and vector-space representations.
- Concordia University Spectrum Open Access Portal source of all crawled PDF documents.
- Scikit-learn documentation for K-Means clustering and vectorization concepts.

Supplementary Learning

- YouTube tutorial: *Building a Web Crawler with Scrapy* concepts of spiders, pipelines, and response parsing.
- YouTube: *SPIMI Algorithm Explained Step-by-Step* understanding block-based index construction.

- BeautifulSoup documentation for HTML parsing and link extraction.
- Articles on PDF text extraction using PyPDF2 and `pdfminer.six`.
- GeeksforGeeks resources on tokenization, normalization, and stopword filtering.
- COMP 479 lecture notes on indexing, DAAT/TAAT, clustering, cosine similarity, and document representation.

Tools and Technical Environment

- Python 3.9 running in Visual Studio Code.
- Scrapy framework for crawling and respecting robots.txt.
- BeautifulSoup for HTML structure analysis.
- pdfminer for extracting raw text from Spectrum PDFs.
- NLTK for tokenization, stemming, and stopword removal.
- Scikit-learn for TF-IDF vectorization and K-Means clustering.
- JSON for storing the final inverted index and metadata.

Acknowledgement

All crawling, text extraction, indexing, querying, and clustering code was written independently by the author.
External resources were consulted strictly for conceptual understanding, debugging assistance, and reinforcement of theoretical material.
ChatGPT was used as a study assistant to clarify IR concepts, verify Python logic, and help refine the structure and clarity of this report.