Assignment 2: Deduplication and Crawling

Prepared by
Kushagra Gupta
(2025909001)

Under the guidance of
Prof. Anil Nelakanti

Submitted in
partial fulfillment of the requirements
for the course of
CS4.406: Information Retrieval and Extraction

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

(November 2025)

Activity 2.1: Deduplication

Goal: To find and group all records belonging to the same person from a noisy dataset of 5,000 entries.

My Approach: Followed a data-cleaning and linkage pipeline:

1. **Exploratory Data Analysis (EDA):** I first inspected the data and found many problems, including missing values, incorrect data types (like date_of_birth as a number), and typos in text fields (like 'nsw' vs. 'nws').
2. **Data Preprocessing:** I cleaned the entire dataset. I fixed all typos, converted dates and postcodes to standard string formats, and filled in missing values.
3. **Blocking:** To avoid comparing 12.5 million pairs, I used a "blocking" strategy. I grouped records by postcode, assuming matches would be in the same postal area.
4. **Pairwise Comparison:** Within these blocks, I used the recordlinkage library to compare pairs. I scored similarity on names, addresses, and date of birth using the Jaro-Winkler method.
5. **Decision & Clustering:** I set a rule that any pair with a total similarity score of 4.0 (out of 5.0) was a "match." I then used the networkx library to find all connected groups, or "clusters."

Key Libraries Used:

- pandas (for data loading and cleaning)
- recordlinkage & jellyfish (for blocking and comparing)
- networkx (for clustering the matched pairs)
- sklearn (for evaluation)

Results & Insights:

- **Insight:** The soc_sec_id column was the "answer key." The 5,000 records only had 2,291 unique soc_sec_ids. I used this to check my work.
- **Result 1 (Blocking**): My blocking strategy was very effective, reducing the number of comparisons by 99.87% (from 12.5 million to just 16,115).
- **Result 2 (Accuracy):** My algorithm found 1,059 clusters. When I compared this to the 2,291 "true" clusters, I got an Adjusted Rand Score (ARS) of 0.7307. This is a great result, showing my model's groups were very similar to the real ones. The lower cluster count means my model tended to "over-cluster," sometimes grouping two different people who just had very similar information.

Activity 2.2: Crawling

Goal: To crawl a local web server, calculate PageRank, and efficiently update node IDs, submitting all findings to an /evaluate endpoint within a 60-second time limit.

My Approach: I built a single, asynchronous "bot" to handle the new time-based rules.

1. **Asynchronous Bot:** I used asyncio and aiohttp to build a bot that runs two tasks at the same time.

2. **Task 1 (Crawler):** A crawler that runs continuously. It discovers new pages and, most importantly, continuously re-crawls all known pages in a loop. This ensures it's always finding the freshest node_ids.
3. **Task 2 (Evaluator):** A scheduler that runs every 14.5 seconds. It pauses, calculates the PageRank of the graph as it knows it, and POSTs the complete results (page_id, latest_node_id, score) to the /evaluate endpoint.
4. **Parsing:** The server did not have a JSON API, so I used BeautifulSoup to parse the raw HTML and scrape the page data, just as in my initial exploration.

Key Libraries Used:

- asyncio & aiohttp (To build the high-speed, multi-tasking bot)
- beautifulsoup4 (to parse the HTML)
- networkx (to build the graph and run PageRank)

Results & Insights:

- **Insight:** The evaluation server's response was key. It only scored coverage and mse (PageRank accuracy), not avg_staleness. This meant my fast, "round-robin" re-crawl was a perfect strategy.
- **Result:** The bot successfully ran for the 60-second window and submitted 4 valid evaluations (at 14.70s, 29.23s, 43.76s, and 58.27s).
- **Final Solution:** The final evaluation report at 58.27 seconds showed:
    - **Excellent Coverage:** The bot discovered **18 out of 19** total pages (94.7% coverage).
    - **Perfect PageRank:** The bot achieved a Mean Squared Error (MSE) of **2.022e-06**. Since 0.0 is a perfect score, this means the PageRank calculation was extremely accurate.

Appendix

GitHub Repo: All the data, code file(ipynb and py), report.

https://github.com/SpyBeast07/HomeworkIRE/tree/main/deduplication_and_crawling