Programming 3 Final Assignment

Personal info:
Name: Hendrik Reitsma
Studentnumber: 444264

Introduction

The assignment involves an investigation of the scientific literature using graph theory and parallel processing. The task is to explore the structure of publishing in the scientific world by analyzing the entire PubMed literature database in XML format. The project requires answering several questions such as:

1. How large a group of co-authors does the average publication have?
2. Do authors mostly publish using always the same group of authors?
3. Do authors mainly *reference* papers with other authors with whom they've co-authored papers (including themselves)?
4. What is the distribution in time for citations of papers in general, and for papers with the highest number of citations? Do they differ?
5. Is there a correlation between citations and the number of *keywords* that papers share? I.e. papers which share the same subject cite each other more often.
6. For the most-cited papers (define your own cutoff), is the correlation in shared keywords between them and the papers that cite them different from (5) ?

Methods

1. Parse relevant data from xml files to pickle format using multiprocessing.
2. Read each file into a dask dataframe.
3. Answer questions using the dataframe and python and dask computing functions.

Results

The analysis of the Medline/PubMed dataset led to the following findings:

- The average number of authors per paper was found to be 4.12.
- The average number of references per paper was 32.6, when excluding references with missing data. However, there were also many papers with missing references, averaging 6.9 per paper.
- The most frequently occurring keywords in the dataset were found to be:

  {'Humans': 18761944, 'Female': 8783532, 'Male': 8644766, 'Animals': 6679180, 'Adult': 5033752, 'Middle Aged': 4386880, 'Aged': 3106240, 'Adolescent': 2040688, 'Child': 1697951, 'Rats': 1575268, 'Mice': 1512175, 'Time Factors': 1191547, 'Treatment Outcome': 986498, 'Aged, 80 and over': 924896, 'Child, Preschool': 922645, 'United States': 898204, 'Pregnancy': 881006, 'Young Adult': 863374, 'Retrospective Studies': 844696, 'Risk Factors': 835533, 'Infant': 795383, 'Molecular Sequence Data': 648268, 'Follow-Up Studies': 647908, 'Infant, Newborn': 607524, 'Prospective Studies': 550936}

Discussion

The discussion highlights several problems encountered during the analysis. Parsing XML files was more time-consuming than expected, and the existence of different XML formats led to difficulties in extracting references from some papers. This resulted in a substantial number of papers appearing to have no references, even though they did. Additionally, some calculations took longer than desired, but these were ultimately addressed by modifying the code to a more efficient approach.

Calculating unique authors

To calculate the unique authors first the nunique().compute() methods were used. This resulted in a very slow code:

```
unique_authors = all_dfs['Authors'].nunique().compute()
```

Improvements were made:

```
authors_count =
all_dfs['Authors'].astype(str).str.split(',').apply(len).sum().compute()
```

In the first line of code, Dask has to perform several steps to compute the result. It needs to split up the data into smaller chunks, distribute the chunks to workers, compute the number of unique authors for each chunk, and then combine the results into a final result.

The second line of code splits each author list into a list of individual author names, which Dask can then process in parallel across multiple workers. Once all the lengths of the author lists have been computed, Dask simply sums them up to get the total number of authors, which can be done quickly and efficiently.

Calculating references

When calculating the average amount of references, it was found that many papers did not have references at all. This is likely due to a problem during parsing. Therefore, to calculate an average properly, the papers without references are dropped first. Using .map also led to faster computation times.

Original:

```
avg_refs_len = all_dfs['Refs'].apply(avg_len, meta=meta).mean().compute()
```

Improved:

```
avg_refs_len = all_dfs['Refs'].dropna().map(len, meta=meta).mean().compute()
```

In the original code, a function called avg_len() was applied to each element in the 'Refs' column using the apply() function. However, this method is not efficient because it applies the function to each element separately, which can be slow. In the updated code, the map() function is used instead of apply(). This function applies the len() function to each partition of the series in parallel, which is faster. Using the map() function with the len() function is faster because len() is a built-in function in Python and is optimized for performance.

Conclusion

Based on the analysis of the dataset using multiprocessing and the dask library, the average amount of authors per paper was found to be 4.12, while the average number of references per paper was

32.6 without any missing values and 6.9 with missing values. Additionally, the most common keywords across all papers were identified. These results provide insight into the characteristics of papers in the dataset. Overall, the use of multiprocessing and dask allowed for efficient analysis of the large dataset.

Appendix: Unanswered questions:

How to answer question 3:

1. Load the PubMed dataset into a graph data structure, where each paper is represented as a node in the graph, and each citation between two papers is represented as a directed edge.
2. Filter the graph to include only papers with two or more authors, as we are interested in co-authorship relationships.
3. Use parallel processing techniques to calculate the number of co-authors that each author has in the filtered graph. This can be done by splitting the graph into smaller subgraphs, and calculating the co-authorship counts for each subgraph in parallel.
4. Use the co-authorship counts to calculate the probability that an author will reference a paper written by one of their co-authors, based on the number of co-authors they have in common.
5. Calculate the average probability that an author will reference a paper written by one of their co-authors and compare it to the probability that they will reference a paper written by someone they have not co-authored with.

How to answer question 4:

1. Extract the publication date and the number of citations for each paper in the dataset.
2. Calculate the distribution in time for citations of all papers in the dataset. We can do this by grouping the papers by year of publication and calculating the average number of citations received by papers published in that year. This will give us an idea of how the average citation count changes over time.
3. Identify the papers with the highest number of citations.
4. Calculate the distribution in time for citations of the papers with the highest number of citations. We can do this by grouping the papers by year of publication and calculating the average number of citations received by papers published in that year.

How to answer question 5:

1. Extract a list of unique keywords from the dataset.
2. Create a dictionary where each keyword is associated with a list of the papers that contain it.
3. Compute the number of keywords that each paper shares with other papers in the dataset.
4. Compute the citation count for each paper in the dataset.
5. Compute the correlation between the number of shared keywords and the citation count for each paper in the dataset.

How to answer question 6:

1. Compute the set of shared keywords between each pair of papers, where one paper is in the set of most-cited papers and the other is in the set of papers that cite it.

2. Compute the correlation between the number of shared keywords and the number of citations for each pair of papers.
3. Compare the correlation coefficients from step 5 for pairs where the first paper is in the set of most-cited papers to the coefficients for pairs where the second paper is in the set of most-cited papers.