Paper Recommendation

Pre-processing
This project requires an additional external library in order to enable access and processing of JSON-objects:
- `javax.json-1.1.2`

RetrieveData.java

Program description
Retrieves the keywords for each paper from the `S2`-data and converts the file `papers.txt` containing the association of Papers to Authors into a file containing the association Authors to Papers.

Required files:
- `authorkeys.txt`
- `papers.txt`
- `s2_papers.txt`

Output:
- `authors.txt`
- `papers_keywords.txt`

Data structure
`static String paperspath` Path to papers-file
`static String s2datapath` Path to Semantic Scholar-data-file

`Author (String id, String name, ArrayList<Integer> papers)`
Represents an author with the corresponding ID, name and a list containing the numbers of the papers published by this author. The list of papers is retrieved from the papers-file.

`Paper(int number, int year, String title, ArrayList<String> keywords, ArrayList<String> authors)`
Represents a paper. The list `authors` contains the IDs of the authors who published this paper.
The keywords are gathered by retrieving the `S2`-ID and DOI of the `S2`-data-file and querying the `S2`-API.

Methods
`ArrayList<Author> getAuthors(String keypath)`
Returns a list of authors included in the authors-keys-file.

`ArrayList<Paper> getPapers()`
Returns a list of papers included in the papers-file. For each paper, the number, title, year, and list of authors are retrieved.

`void writeToNewFile(String path, String content)`
Creates a new file using the given `path` and stores the text given as `String content`.

Program Flow (main method)

- The input files are processed
- Names of the output files are defined
- A list with all authors is retrieved (`getAuthors()`)
- The authors-file is created and stored (`writeToNewFile()`)
- A list of papers with their keywords is retrieved (`getPapers()`)
- The papers-keywords-file is created and stored (`writeToNewFile()`)

PaperRecommendation.java

Program description

Produces a `.txt`-file containing paper recommendations and recommendation weights corresponding to the data given in the similar-authors-file.
Furthermore, a `.js`-file is produced containing `JSON`-objects each containing the paper recommendations for an author.

Input:
- `authors.txt`
- `papers_keywords.txt`
- `similarauthors.txt`
- *`suffix`* `(i.e., s95)`

Output:
- `p_recommendations_`*`suffix`*`.txt`
- `p_recommendations_`*`suffix`*`.js`

Data structure

`static String authorspath` Path to authors-file
`static String suffix` Suffix for output-files and produced `JSON`-object-names
`static String recommendationsjsfile` Content of `.js`-output-file
`static String recommendationsfile` Content of `.txt`-output-file
`static HashMap<String, Author> authorslist` Reference list with relevant authors
`static HashMap<Integer, Paper> paperslist` Reference list with all papers

`Author (String id, String name, ArrayList<Integer> papers, ArrayList<String[]> simauthors)`
Represents an author. The list `papers` contains the numbers of papers which the author published. The list of papers is retrieved from the authors-file.
Each entry in the list `simauthors` refers to a similar author and is represented by an array with 2 entries; the first entry contains the similar author's ID, the second entry contains the similarity weight of both authors.

`Paper (int number, int year, String title, ArrayList<String> authors, ArrayList<String> keywords)`
Represents a paper. The list `authors` contains the IDs of the authors who published this paper.

Methods

`String getAuthorName (String id)`
Retrieves the name of the author corresponding to `id`.

`double getSimValue(String author1, String author2)`
Returns the similarity value of the authors corresponding to the IDs `author1` and `author2`. If either of those authors is not contained in the authors-similarity-file, `-1.0` is returned.

`void retrieveRecommendations(ArrayList<String> authors)`
Computes paper recommendations for each author represented in the list `authors` and sorts the recommendations by year and similarity value (`sortRecommendations()`).
The content of the output-text-file is assigned containing the recommended papers and corresponding similarity values for each author.

Furthermore, the output-js-file is assigned containing a JSON-object for each of the authors in authors. Each JSON-object is named according to the represented author and the suffix given as input and contains the following information:

```
PRecAxsuffix = { “id”: “Ax”, “numrecommendations”: …,
  “papers”: [
      { “number”: …, “year”: …, “title”: ”…”, “weight”: z.y,
        “authors”: [
            { “id”: “…”, “name”: “…”, “weight”: x.yz,}, …
        ],
        “topics”: [
            { “keyword”: “…”}, …
        ]
      } …
  ]
};
```

ArrayList<Paper> sortRecommendations(ArrayList<Double[]> papers)
Sorts the list papers given as input by year and similarity value and returns the sorted list. The list papers contains paper recommendations; each item of the list consists of an array with two entries: a paper-number and a similarity value.

void writeToNewFile(String path, String content)
Creates a new file using the given path and stores the text given as String content.

Program Flow (main method)

- Input-files and the suffix given as input are processed
- Names of the output files are defined
- The authors whose similar authors are listed in the similar-authors-file are retrieved and stored in the authors-reference-list
- The similar authors for other authors occurring in the similar-authors-file are retrieved and those other authors are then also added to the authors-reference-list
- The papers-reference-list is retrieved
- The paper recommendations are retrieved and assigned to the output-files (retrieveRecommendations())
- The .txt-recommendations-file is stored (writeToNewFile())
- The .js-recommendations-file is stored (writeToNewFile())