Pre-Processing

Pre-processing
This project requires some additional external library in order to enable access and processing of Excel-files and JSON-objects:
- `commons-collections4-4.1`
- `poi-3.17`
- `xmlbeans-2.6.0`
- `javax.json-1.1.2`

OutputKConversion.java

Program description
Produces an assignment file where each line contains the number, name, and URI of an author. Furthermore, the `Excel`-file containing papers is converted to `.txt` and the authors' URIs in that file are converted to authors' numbers.

Required files:
- `author-key-map.txt`
- `author-list.txt`
- `metis.xlsx`

Output:
- `authorkeys.txt`
- `papers.txt`

Data structure
`static String listpath` Path to Authors-list-file

`Author (String id, String uri, String name)`
Represents an author.

Methods
`ArrayList<Author> getAuthors(String mappath)`
Retrieves the ID and names of the authors from `mappath` and stores them in a list of type `Author`. The URI of the authors is retrieved from the author-list-file when an instance of that class is created.

`void convertExcelToTxt(String excelpath, String txtpath)`
Converts the `Excel`-file corresponding to `excelpath` to a `.txt`-file. The occurring authors' URIs are converted to the corresponding IDs by accessing the list `authors` which was created previously

`void appendtoFile(String path, String content)`
Appends the text given as `content` to the already existing content of the file corresponding to `path`.

`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 are processed
- The names of the output-files are assigned
- A list containing the authors with their IDs, names and URIs is retrieved (`getAuthors()`)
- The obtained authors data is assigned to the authors-keys-output-file-variable
- The obtained assignment is stored in the author-keys-output-file (`writeToNewFile()`)
- The Excel-file is converted to a .txt-file (`convertExcelToTxt()`)

ProcessDBLPData.java

Program description

Filters the information contained in the `.nt`-files from the `DBLP`-data set by eliminating data which is not associated with conferences at all. Furthermore, the remaining data is split into data referring to conference editions (`dblp_confs`) and data referring to publications in the context of conference editions (`dblp_papers`).

Required files:
- `nt-files/`

Output:
- `dblp_papers/`
- `dblp_confs/`

Methods

`void filterFiles(String dirpath, ArrayList<String> ntfiles, String papersdirpath, String confsdirpath)`

The folder corresponding to `dirpath` contains `nt.`-files. The names of the `.nt`-files are contained in `ntfiles`. This method creates empty files with the names stored in `ntfiles` in the directories corresponding to `papersdirpath` and `confsdirpath`. Furthermore, the N-Triples contained in the `.nt`-files are filtered:

- Triples which are not referring to conferences in any way are eliminated
- Triples which refer to a conference edition are stored in a file in the directory corresponding to `confsdirpath`
- Triples which refer to a paper published at a conference are stored in a file in the directory corresponding to `papersdirpath`

The empty files in the directories corresponding to `confsdirpath` and `papersdirpath` are removed (`eraseEmptyFiles()`)

`void eraseEmptyFiles(String directory, ArrayList<String> ntfiles)`

Deletes the empty files in the folder corresponding to `directory` with `ntfiles` being the list of file-names in that directory.

Program Flow (main method)

- Input-files are processed
- The names of the output-directories are assigned
- The `.nt`-files are filtered (`filterfiles()`)

ProcessSemanticScholarData.java

Program description

Reduces the amount of data retrieved from `Semantic Scholar` (`S2`) by filtering data sets with certain values for the attribute `venue`. The venue-names stored in `S2` tend to differ from the ones stored in `DBLP`, thus the relevant names had to be retrieved manual and previous to running this program. Using this program, the used storage is reduced from 87.4 `GB` (complete `S2`-data sets) to 35.9 `MB` (potentially relevant `S2`-data sets).

Required files:
- `s2data/`

Output:
- `s2_papers.txt`

Methods

`boolean filterPapers(ArrayList<String> inputfiles, String[] names, ArrayList<String> exactnames, String outputpath)`

Filters the `S2`-data sets. The paths to the files containing the data are given in the list `inputfiles`. The array `names` contains potential parts of venue-names; any data set with a venue-name containing an element of this list is kept. The list `exactnames` contains venue-names; if they are equal to the venue-name of a data set, this data set is also kept. For the comparison, only alphanumerical characters are considered – in a case-insensitive way – in order to prevent problems with special characters. The data sets which are identified to be potentially relevant are appended to the output-file which corresponds to the path given as `outputpath`.

`void appendToFile(String path, String content)`

Appends the text given as `content` to the file corresponding to `path`. If the file does not exist, a new file with the specific name is created.

Program Flow (main method)

- Input-directory is processed
- A list of files in the input-directory is retrieved
- An array of venue-name-parts for filtering is defined
- A list of complete venue-names for filtering is defined
- The path of the output-file is assigned
- The `S2`-data sets given in the retrieved files are filtered (`filterPapers()`) and the remaining data sets are stored in the output-file (`appendToFile()`)

ProcessClusters.java

Program description

Filters the clusters produced by `SemEP` or `METIS` according to the authors' numbers given as input and stores the corresponding clusters in subdirectories. Furthermore, a file is produced which contains a list of similar authors and weights for each of the authors given as input.

Input:
- `cluster-files/`
- `Auth_matrix.txt`
- N_1 N_2 N_3 N_4

Output:
- `selectedpredictions/`
- `similarauthors.txt`

Data structure

`static String authorsmatrixpath` Path to authors-matrix-file
`static String clusterdirpath` Path to clusters-directory
`static ArrayList<String> clusterfiles` List of filenames in clusters-directory
`static ArrayList<Author> authors` List of authors regarded for filtering
`static String newdir` Path of output-root-directory

`Author (String id, ArrayList<SimAuthor> simauthors)`
Represents an author. The list `simauthors` contains authors who are regarded as being similar to this author.

`SimAuthor (String id, double weight)`
Represents an author who is similar to another author where `weight` refers to the similarity value of both authors.

Methods

`boolean containsAuthor(String filepath, String author)`
Checks if `author` is contained in the file corresponding to `filepath` and if there is more than one author referred to in the file.

`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 are processed
- The list with names of all files in the cluster-directory is retrieved
- The list with authors is created corresponding to the authors' numbers given as input
  - When a new instance of `Author` is created, the cluster files corresponding to this author are retrieved
- The similar authors-file is created and stored (`writeToNewFile()`)

CollaborationFilter.java

Program description

Search the clusters in a subdirectory corresponding to a specific author for collaborations with other authors in each cluster and removes the nodes of authors who collaborated before. Thus, the result is a collaboration recommendations-network as only the association of authors remains who did not collaborate before. Moreover, authors who collaborated before are removed from the Similar-Authors-file.

Input:
- `selectedclusters/`
- `similarauthors.txt`
- `authorkeys.txt`
- `nt-files/`

Output:
- `filteredclusters/`
- `similarauthors_new.txt`

Data structure

`static ArrayList<String> ntfiles` List with paths to the files contained in the nt-directory given as input.

`static String authorskeypath` Path to the Authors-Key-file

`static ArrayList<Collaboration> colist` List with retrieved collaboration-values

`ClusterDirectory (String name, String root, ArrayList<String> filenames)`

Represents a directory corresponding to a specific author and contains files each representing a cluster corresponding to this author. The attribute `name` refers to the directory's name and is always the number contained in the author's ID (*i.e., 138 for author A138*). The attribute `root` refers to the name (or path) of the directory which contains this directory. The list `filenames` contains the names of all relevant cluster-files in this directory.

`Author (String id, String uri, String name)`

Represents an author. The ID is given when the object is instantiated; URI and name are retrieved from the Authors-Key-file.

`Collaboration (String author1, String author2, boolean value)`

Represents the association of two authors. The attribute `value` indicates whether those authors collaborated previously, or not.

Methods

`boolean collaborationExists(Author a1, Author a2)`

Checks if the authors represented by `a1` and `a2` collaborated previously, or not. If this collaboration was checked before, the result is stored in the list `colist` and can be retrieved from there; otherwise the collaboration is checked by iterating through the nt-files.

`void filterSimAuthors(String path)`
Creates a copy of the Similar-Authors-file containing only associations of authors who did not collaborate before.

`void filterClusters(ClusterDirectory cd, String outputdir)`
Filters the clusters corresponding to a specific author (associated with directory `cd`) by removing the nodes which refer to authors who have collaborated before with this author. The newly created directory is stored in the root-directory referring to `outputdir`.

`void cleanClusters(ClusterDirectory cd, String outputdir)`
Removes the files from the output directory which are empty or contain only nodes referring to the author who corresponds to the directory `cd`.

`void appendToFile(String path, String content)`
Appends the text given as `content` to the file corresponding to `path`. If the file does not exist, a new file with the specific name is created.

Program Flow (main method)

- Input-data is processed
- A list with paths to all files in the nt-directory is retrieved
- A list with all sub-folders of the cluster-directory is retrieved
- The Similar-Authors-file is filtered to retrieve only authors who did not collaborate before (`filterSimAuthors()`)
- The name of the output-directory is assigned and the output-root-directory is created
- Each cluster-directory is filtered in order to retrieve predictions (`filterClusters()`) the resulting new directory with the corresponding cluster-files is stored in the output-directory
- Meaningless cluster-files are removed from each newly created sub-directory (`cleanClusters()`)