Notes

SciGraph:

ExtractPapers.java

Input	Output
▪ `SciGraph-Data/scigraph`*X*`.nt`	▪ `SciGraph-Data/output/sgpapers.txt` ▪ `output/authornames.txt`

In SciGraph, conference papers are stored as book-chapters, as they are published in inproceedings of conferences.
The book-editions referring to proceedings of ISWC-conferences had to be retrieved manually.
This program takes the SciGraph-nt-triples-files as input and filters the relevant papers using the SciGraph-IDs referring to ISWC-related book-editions.
For each of those relevant papers SciGraph-ID, title, publishing year, DOI, and contribution-IDs are retrieved from the nt-triples files.
The retrieved data about papers is stored in the output-file `sgpapers.txt`.
Moreover, the names corresponding to the retrieved contribution-IDs are retrieved and stored in the output-file `authornames.txt`. This file can be used to facilitate the manual retrieval of the duplicate-file needed in the next step.

InputXLSXKorona.java

▪ `SciGraph-Data/scigraph`*X*`.nt` ▪ `output/duplicates.txt` ▪ `SciGraph-Data/output/sgpapers.txt`	▪ `SciGraph-Data/data_filtered/sg_contributors_filtered.nt` ▪ `output/papers.xlsx`

The SG-data referring to contributions is extracted from the SG-files to decrease the data volume.
The duplicates file is retrieved manually. Each line consists of at least two entries, each referring to the same author. If names contained in another than a first entry occur in the contributors-data of SG, they are replaced by the first entry ensuring that all contributions of an author are retrieved and associated to this author in our knowledge graph.
As duplicates of author-names are removed, the resulting contributors-file is stored as output-file `sg_contributors_filtered.nt`.
Furthermore, the file `sgpapers.txt` is converted into an Excel-spreadsheet containing the ID of the paper in our knowledge graph, title, number of authors, year, and names of the authors of the paper. The names of the authors are retrieved from the contributors-file mentioned above.

RetrieveEntities.java

▪ `output/papers.xlsx`	▪ `output/Author.txt` ▪ `output/author-key-map.txt` ▪ `output/Conf.txt` ▪ `output/Conf_matrix.txt` ▪ `output/Auth-Conf_graph.txt`

The Excel-spreadsheet containing data about the relevant papers is used as input to generate lists of different entities in the knowledge graph.
The names of the authors were retrieved from the Excel-spreadsheet and each unique author named was assigned to an ID. The association of ID and name was stored in the output-file author-key-map, while a list with the author-IDs preceded by the number of IDs contained in the kg is stored in the output-file `Author.txt`.
The output-file `Conf.txt` contains the number of conference editions in the kg and the list of those events. As we are only considering papers published in the ISWC-conference series (venue), the generation of a conference-key-map was not necessary. The conferences are identified by an ID consisting of "C" + the year when the event took place.
The first-line of the output-file `Conf_matrix.txt` contains a number depicting the number of rows and columns of the symmetric similarity matrix. Furthermore, the file contains the similarity matrix itself. It contains the similarity-values between each conference. The diagonal values are always 1.0 as every conference is identic to itself and thus has the highest possible similarity value. The

similarity between two conferences is calculated as follows: number of authors who published in both conferences (intersection) / number of authors who published in either of the conferences (union). Finally, the graph-file needed as input for the semEP-tool is generated: The graph-file contains the weighted edges between conference- and author entities. The weights are calculated by dividing the number of papers an author published in a conference by the maximal number of papers published by an author in a conference. The graph edges in the file are preceded by the number of edges in the first line.

FilterContributionData.java

▪ `SciGraph-Data/scigraph`*X*`.nt` ▪ `SciGraph-Data/sg_bookdata.nt` ▪ `SciGraph-Data/sg_conferencedata.nt`	▪ `SciGraph-Data/data_filtered/sgcontributions.nt` ▪ `SciGraph-Data/data_filtered/sgbooks.nt` ▪ `SciGraph-Data/data_filtered/sg_bookdata_filtered.nt` ▪ `SciGraph-Data/data_filtered/sg_conferencedata_filtered.nt`
The purpose of the next two programs is to reduce the amount of data and retrieve the associations between authors and all conferences (not just ISWC) as this is necessary to generate the authors similarity matrix. To increase the usability of the data retrieved from Springer SciGraph and to decrease the duration of processing the data, data which was not relevant for our cause was filtered out. From the SciGraph-files containing data about book-chapters only triples containing "`hasContribution>`" and "`hasBook>`" were kept and stored in the output-files `sgcontributions.nt` and `sgbooks.nt` respectively. Furthermore, lines of the SG-triples-file containing data about books (like inproceedings) containing `hasConference` were extracted and stored in the output-file `sg_bookdata_filtered.nt`. Likewise, lines of the SG-triples-file corresponding to conferences containing `core/acronym` were extracted and stored in the output-file `sg_conferencedata_filtered.nt`.	

GetAuthorConferences.java

▪ `output/author-key-map.txt` ▪ `SciGraph-Data/data_filtered/sg_contributors_filtered.nt` ▪ `SciGraph-Data/data_filtered/sgcontributions.nt` ▪ `SciGraph-Data/data_filtered/sgbooks.nt` ▪ `SciGraph-Data/data_filtered/sg_bookdata_filtered.nt` ▪ `SciGraph-Data/data_filtered/sg_conferencedata_filtered.nt`	▪ `SciGraph-Data/output/author_publ_all.txt` ▪ `SciGraph-Data/output/author_books_all_filtered.txt` ▪ `SciGraph-Data/output/author_confs.txt`
First, the associations of authors and contribution-IDs are retrieved and stored in the temporary file `author_cont_all.txt`. Next, for each contribution of the authors in the kg, the scigrpah-ID of the corresponding publication is retrieved using the file `sgcontributions.nt`. The associations of authors and publications are stored in the output-file `author_publ_all.txt`. Using the publication-IDs, the corresponding scigraph-book-IDs are retrieved using the file `sgbooks.nt`. The associations of authors and books are stored in a temporary file. Then the temporary file is filtered and duplicate associations of authors and books are removed resulting in the output-file `author_book_all_filtered.txt`. Finally, using the file `sg_bookdata_filtered.txt`, the scigraph-conference-ID for each book in the `author_books_all_filtered.txt`-file is retrieved. Using the file `sg_conferencedata_filtered.txt`, the acronyms corresponding to the scigraph-conference-IDs are retrieved. The association of authors and conferences is stored in the output-file `author_confs.txt`, each line containing the ID of an author in our kg and the acronym of a conference.	

AuthorSimilarityMatrix.java

▪ `output/author-key-map.txt` ▪ `SciGraph-Data/output/authorconfs.txt`	▪ `output/Auth_matrix.txt`

First, a list of authors is retrieved containing the ID of that author in our kg and a list containing the conferences in which this author published and the number of papers they published in this conference. Next, for each of the authors in the list the similarity with the other authors is calculated and stored in a similarity matrix. The similarity of two authors is calculated as follows: The intersection of the number of papers both authors published in the same conference divided by the number of papers published by either of the authors in total (union). The output-file `Auth_matrix.txt` contains the number of rows and columns of the matrix in the first line as well as the matrix itself.

PercentileCalculation.java

▪ output/Conf_matrix.txt ▪ output/Auth_matrix.txt	▪ output/percentiles.txt
This program calculates the threshold-values for both matrix-files. First, the similarity values of each author pair in the matrix are retrieved – as the matrices are symmetric, only half of the values need to be accessed. For the list of similarity values in each matrix, the minimal, maximum, average, and median value are retrieved. Furthermore, the values for the percentiles (i=10, i<=95, i+=5, i=98) are calculated. The resulting values are stored in the output-file `percentiles.txt`.	

GenerateMETISGraph.java

▪ output/Author.txt ▪ output/Auth_matrix.txt ▪ output/Conf.txt ▪ output/Conf_matrix.txt ▪ output/Auth-Conf_graph.txt ▪ *<threshold authors> <threshold conferences>*	▪ relationalsimilarities/simrelations*X*.txt ▪ metis_graphs_output/metis*X*.txt
The number of rows and columns of the relational similarities matrix is the number of edges in the file `Auth-Conf_graph.txt` given as input. Each edge of the graph is compared to another edge of the graph. The similarities of both the authors and both the conferences are retrieved respectively. The similarity values are compared to the corresponding threshold-values. If either of the author or conference similarity value is below the corresponding threshold value, the similarity value of both edges is 0; otherwise the similarity-value is the calculated as the similarity value of the authors multiplied by the similarity value of the conferences. The result is a matrix containing the similarity-value of each of the edges in the graph file. Using the generate similarities-matrix the METIS-graph is created and stored in the output-file metis*X*.txt.	

DBLP:
KORONA

DBLP & SG: *application of clustering tools*

SciGraph:
METIS2semEP.java

▪ output/Auth-Conf_graph.txt ▪ metis_graphs_output/metis*X*.txt.part*Y*	▪ cluster-files/m*X*/

DBLP & SG:

OutputKConversion.java

▪ output/author-list.txt (*dblp*) ▪ output/author-key-map.txt ▪ output/metis.xlsx (*dblp*) / papers.xlsx (*sg*)	▪ output/authorkeys.txt (*dblp*) ▪ output/papers.txt
For the DBLP-source, the input-files `author-list.txt` containing the dblp-IDs of authors and `author-key-map.txt` containing the IDs and names of authors in our knowledge graph are merged into the output-file `authorkeys.txt` containing all three properties for each author in the kg. For the SG-source this step was omitted, as the relevant data for papers- and venue recommendations was already retrieved and the SciGraph-ID of the authors is not needed anymore. Moreover, the Excel-spreadsheet containing relevant papers is converted into a text-file where the author-names are converted into the author's IDs in our kg.	

ProcessClusters.java

▪ cluster-files/…/ ▪ output/Auth_matrix.txt ▪ *ID-numbers of authors to consider*	▪ similarauthors/….txt ▪ selected-clusters/…/
This program takes the output of the clustering-tools and the numbers of authors to consider as input. For each of the given authors, the relevant cluster-files are reviewed and other authors contained in this file are retrieved as they are considered to be similar. The relevant clusters for each author are copied to a new directory, which is named as the author's number. Clusters related to an author but not containing any other authors are removed as they are not significant. For each author, the similarity-values for his similar authors are retrieved and those associations of similar authors are stored in a new text-file, where three lines always refer to one of the authors given as input to this program. The first line always contains the author's ID in the kg. The second line contains a list of IDs of authors considered to be similar, and finally, the third line contains a list of similarity values between the author stated in the first line and each of the authors in the second line. Example!	

CollaborationFilter.java

▪ selected-clusters/…/ ▪ similarauthors/….txt ▪ SciGraph-Data/output/author_publ_all.txt (*sg*) ▪ DBLP-Data/nt-files (*dblp*)	▪ selected-clusters/cp/…/ ▪ similarauthors/cp/….txt
For the DBLP-source the nt-files are used to check if two authors did co-author before. For the SG-source this is done using the input-file `author_publ_all.txt` as this contains all the publications of each author in the KG stored in the SG-dataset. If the same publication occurs for two authors, they obviously did work together before. The similar-authors-file is filtered and the associations of author-pairs who collaborated before are removed. Moreover, the cluster-files are filtered and similar authors who collaborated before are also removed. Empty clusters and clusters containing only 1 author are erased as well. The remaining associations are considered to be collaboration predictions/recommendations. Example!	

Paper Recommendation

DBLP & SG:

ProcessSemanticScholarData.java

▪ semantic-scholar-data/	▪ s2_papers.txt
The knowledge graph was extended with keywords from another source (semantic scholar). Those keywords were visualized along with paper recommendations although they were not utilized when the paper recommendations were generated. This program extracts papers related to an ISWC-	

event from the enormous amount of data stored in semantic scholar and reduced the data volume significantly to only a few MB.
The major challenge for this step was to retrieve the names of ISWC-events from the semantic scholar source as they use other venue labels than SciGraph and DBLP. This step had to be done manually by looking up some of the papers in our KG in semantic scholar, in this way 50 event-names were retrieved and were then used in this program to extract related papers. The paper-data in semantic scholar is stored as JSON-objects, the relevant objects were retrieved and stored in the output-file `s2_papers.txt`.

RetrieveData.java

▪ `output/author-key-map.txt` *(sg)* ▪ `output/authorkeys.txt` *(dblp)* ▪ `output/papers.txt` ▪ `SciGraph-Data/output/sgpapers.txt` *(sg)* ▪ `s2_papers.txt`	▪ `paperrecommendations/authors.txt` ▪ `paperrecommendations/papers_keywords.txt`

First, the name and ID of each author is retrieved as well as a list of the IDs of papers which were published by each author. This list of authors is stored in the output-file `authors.txt`.
Next, the semantic scholar-ID and DOI of the papers in the knowledge graph are retrieved using the file `s2_papers.txt`. For the DBLP-source, the papers could only be matched by title and year. Whereas we were able to retrieve DOIs for all the papers from the SG-source by iterating through the SG-nt-files again and matching DOIs using the SG-IDs of the papers in our KG. For the SG-source we used the DOI to match the papers in our KG to papers from the semantic scholar source.
As the keywords of papers are not contained in the downloaded Semantic Scholar data, but can be accessed online using the S2-API, requests for each of the papers in the KG were made using the papers' S2-IDs, and if this request was not successful, another request was done using the papers' DOIs. The requests delivered results in forms of JSON-objects which contained the property "topic". The values for this property were stored in our KG as keywords.
We were able to retrieve keywords for most of the papers in our KG this way.
The file papers.txt given as input was enriched with the paper's keywords and stored in the output-file `papers_keywords.txt`.
Example:

PNr	[Title]	Year	[Keyword1]	[Keyword2]	[Keyword3]	…
A2	A3	A4				

PaperRecommendation.java

▪ `paperrecommendations/authors.txt` ▪ `paperrecommendations/papers_keywords.txt` ▪ `similarauthors/….txt` ▪ *`suffix`*	▪ `p_recommendations_`*`suffix`*`.txt` ▪ `p_recommendations_`*`suffix`*`.js`

This program retrieves recommendations for the authors referred to in the similar-authors-file.
First, all the authors occurring in the similar-authors-file are stored in a list. For each author ID, name, a list of published papers (paper-number) and a list of similar authors (author-ID and sim.-value) are stored.
Then, a list with all papers in the KG is retrieved. For each paper number, year, title, a list of authors (IDs) and a list of keywords is stored.
(Write more with an example. Assuming, we have an author A, who should receive recommendations…")
Then the main step happens and the paper recommendations are generated. For this, all the papers co-authored by similar authors of an author A are retrieved and the sum of all similarity-values is calculated as a maximum-value. Then, the authors of all the retrieved papers who are also similar authors are stored in a list (i.e., if a paper was written by three authors similar to author A, these three similarity values are summed up). The similarity values of those authors are summed up and to normalize the result divided by the maximum value. The result is the similarity-value of the paper and the author A. Obviously, the papers written by author A himself are removed from the list of possible recommendations.

Following, the list of papers containing possible recommendations is sorted by year and then by its similarity value. Finally, the recommendations-list is reduced to containing maximal 5 papers per year.
The resulting associations between authors and papers recommended to them are stored in the output-file `p_recommendations_`*`suffix`*`.txt` whereas this is also converted into JS-objects stored in the output-file `p_recommendations_`*`suffix`*`.js` which can be used for visualization.
(Explain suffix, take from visualization)

Venue Recommendation

DBLP:

ProcessDBLPData.java

▪ `DBLP-Data/nt-files/`	▪ `DBLP-Data/dblp_papers/` ▪ `DBLP-Data/dblp_confs/`

This program extracts nt-triples related to conferences and stores them in the directory `dblp_confs` and extracts nt-triples related to papers and stores them in the directory `dblp_papers`. Thus, the total volume of the nt-files is reduced and the processing time of the data is decreased.

DBLP & SG:

RetrieveData.java

▪ `output/authorkeys.txt` (*dblp*) ▪ `similarauthors/….txt` (*dblp*) ▪ `DBLP-Data/dblp_papers/` (*dblp*) ▪ `DBLP-Data/dblp_confs/` (*dblp*) ▪ `SciGraph-Data/data_filtered/author_books_all_filtered.txt` (*sg*) ▪ `SciGraph-Data/data_filtered/sg_bookdata_filtered.txt` (*sg*) ▪ `SciGraph-Data/data_filtered/sg_conferencedata_filtered.txt` (*sg*)	▪ `venuerecommendations/…/authorconfs.txt` (*dblp*) ▪ `venuerecommendations/…/venues.txt` (*dblp*) ▪ `venuerecommendations/…/conferences.txt` (*dblp*) ▪ `venuerecommendations/conferences.txt` (*sg*) ▪ `venuerecommendations/venues.txt` (*sg*) ▪ `venuerecommendations/authorconfs.txt` (*sg*)

For the **DBLP-source** this program needs to be run for every different similar-authors-file, as it always only considers the authors occurring in this file. Processing the conferences-data for all authors in the kg would be too time-consuming.
First, using the input-file `authorkeys.txt`, the kg-ID and DBLP-URI of each author in the kg are retrieved.
Next, the DBLP-conference-URIs are retrieved for every author of the kg occurring in the similar-authors-file by iterating through the DBLP-nt-triples-files related to papers (`dblp_papers/`) and filtering out all publication-URIs of this author: the subject of each triple with a predicate containing "`#authoredBy`" and the author's URI as object equals one of those publication-URIs. For each retrieved publication, the conference-URI is retrieved: the publication-URI is the subject and the predicate contains "`#publishedAsPartOf`" so the object is the conference-URI we want to retrieve. Every retrieved conference is also added to another list.
The result of the first two steps is a list with associations of author-IDs and lists of conference-URIs for each author. Furthermore, we have a list containing all conferences assigned to any author. This list is used to generate an kg-ID for each of these conferences consisting of "CE" + and a number.
Using the DBLP-URI of each conference, additional properties of this conference are retrieved by iterating through the DBLP-nt-triples-files related to conferences (`dblp_confs/`): when the subject of a triple equals the conference URI, the object of the triple is

- the title when the predicate contains "`#title`"
- the venue when the predicate contains "`#publishedInSeries`" or "`#publishedInSeriesVolume`"
- the year when the predicate contains "`#yearOfPublication`"

Thus, each of the conferences is stored in our kg with an ID, its DBLP-URI, its title, its year, and the venue it was part of. The list with conferences and their properties is stored in the output-file `conferences.txt`. For each of the conference-URIs in the author conferences associations, the corresponding conference-kg-ID is retrieved from the conferences list and the associations of author-IDs and conference-IDs is stored in the output-file `authorconfs.txt`.
By accessing the list of conferences stored in the kg, the corresponding venue-URIs are retrieved. Each venue is added to a list containing all occurring venues. For each venue, an ID consisting of "V" + a number is added. The name of each venue (conference-series) is retrieved by querying the DBLP-API with JSON-requests (as this information is not contained in the DBLP dump-file). All in all, we have the following properties for each venue: kg-ID, DBLP-URI, and name. A list of conferences related to a venue is retrieved using the list of conferences. This data corresponding to venues is stored in the output-file `venues.txt`.
For the **SG-source** it only needs to be run once, as we are retrieving the data for all authors in our knowledge graph and not for a specific similar-authors-file. This is because our kg based on the SG-source is considerably smaller than the data set of DBLP we use.
Using the input file `author_books_all_filtered.txt` a list with the authors in the KG is retrieved. Each author (kg-ID) is associated with a list of books (SG-IDs) in which they published. Accessing the input-file `sg_bookdata_filtered.txt`, the SG-ID of the conferences associated with retrieved books are retrieved. When retrieving the conference-IDs, other properties of this event are looked up as well: using the SG-ID, for each conference, the event name, year, and the ID of the conference's venue (conference series) are retrieved as well. Furthermore, the conference is assigned an ID in the kg and added to the output-file `conferences.txt`. The retrieved venue is (if not already existing) added to a list of venues and assigned an ID in the kg. Also, other properties of the venue are retrieved: name and acronym. The venues are stored in the output-file `venues.txt`.
As the associations of authors and books and the associations of books and conferences were retrieved earlier, the associations of authors and conferences can now be derived. The IDs of all authors in the kg along with a list of conferences in which they published is stored in the output-file `authorconfs.txt`.

VenueRecommendation.java

▪ `venuerecommendations/…/authorconfs.txt` (*dblp*) ▪ `venuerecommendations/…/venues.txt` (*dblp*) ▪ `venuerecommendations/…/conferences.txt` (*dblp*) ▪ `venuerecommendations/conferences.txt` (*sg*) ▪ `venuerecommendations/venues.txt` (*sg*) ▪ `venuerecommendations/authorconfs.txt` (*sg*) ▪ `similarauthors/….txt` ▪ `output/authorkeys` (*dblp*) ▪ `output/author-key-map` (*sg*) ▪ *suffix*	▪ `v_recommendations_`*suffix*`.txt` ▪ `v_recommendations_`*suffix*`.js`

First, the input-file `conferences.txt` is processed and a list containing the relevant conferences with kg-id, title, corresponding venue-ID, and year is created.
Next, the input-file `venues.txt` is processed and a list containing the relevant venues with kg-id, name, and a list of related conference-kg-IDs is created.
The handling of authors is similar to the proceeding for paper recommendations, all the authors occurring in the similar-authors-file are stored in a list. For each authorID, name, a list of conferences, and a list of similar authors (author-ID and sim.-value) are stored.
Following, the main step is done and the venue recommendations are generated: all venues corresponding to conferences in which similar authors of an author A published (and not author A hiself) are retrieved and the sum of all similarity-values (of authors) is calculated as a maximum-value. For each venue, the weights of authors who are similar to author A and who published in this conference are summed up. To normalize the result, the sum of those weights is divided by the maximum value. The result is the similarity-value of the venue and the author A.
Next, the list of venues containing possible recommendations is sorted by similarity-value.

The resulting associations between authors and venues recommended to them are stored in the output-file `v_recommendations_`*`suffix`*`.txt` whereas this data is also converted into JS-objects and enriched with information about the corresponding conferences and stored in the output-file `v_recommendations_`*`suffix`*`.js` which can be used for visualization.
Describe / Explain Output-files

Visualization

ListToJS.java

▪ `visualizationdata/selectauhor.txt`	▪ `visualizationdata/selectauthor.js`

The input-file `selectauthor.txt` was created manually and contains the IDs and names of the authors we want to visualized recommendations for. This program converts the list into a JavaScript-object and stores it in the output-file `selectauthor.js`. It is important, that the filename is “selectauthor” as the generated object has the same name and the implemented visualization uses this object to display the select-list. (Screenshot)

ClustersToJS.java

▪ `selected-clusters/…/` (*unfiltered*) ▪ `select-clusters/cp/…/` (*filtered*) ▪ `output/authorkeys.txt` (*dblp*) ▪ `output/author-key-map.txt` (*sg*) ▪ *suffix*	▪ `network_`*`suffix`*`.js`

This program converts the cluster-files given as input into JavaScript-objects in such a way that they can be used with the visualization tool.
The author-key-files are used to retrieve the authors’ names. Then a file is generated containing one JavaScript-object for each author related to one of the sub-directories in the selected-clusters-directory. Each JS-object contains the author’s ID and a list of the clusters related to him. Each cluster contains a set of nodes (either an author or a conference) and a set of links (always connecting and author and a conference). The nodes are subdivided into three groups; group 1: the author to receive recommendations, group 2: authors who are recommended to the author of group 1, group 3: conferences. The weight for each link is stored as well.
Effectively, when using the unfiltered clusters, the output of the clustering-tools is visualized, when using the filtered clusters, the recommendations are shown.
The suffix indicates the tool, the filtering and the percentile used. Thus, i.e., `s85` indicates the unfiltered output of `semEP` with percentile 85, whereas `cpm90` indicates the output of `METIS` using percentile 90 and having collaborations filtered out.
Example!
Move explanation of suffix to paper recommendation and make a reference here to his section.

Web-Application

The visualization shows author / paper / venue recommendations for 4 different percentile values and offers the possibility to filter out already existing (in this data source) collaborations. The visualization consists of a web-application. The visualization was applied to both data-sources. The recommendation and author data was plugged in by storing the corresponding .js-files in the directory “Visualization/data”. → constraints of names / look at code / add screen-shots of visualization

When preparing the data for visualization, UTF-character-sequences are removed from the data retrieved from the SG-source. This was not necessary for the DBLP-source as it does not contain such sequences.