SciPi: Scientific Publication Analytics Prototype

Dylan Vassallo, Patrick Bezzina

I. Introduction

In recent years, Scientometrics, the area of research dealing with quantifying and analysing scientific literature, "is at crossroads among social sciences, information science, and advanced computing" according to [1]. Amongst other topics, community detection, collaborative patterns, authorship patterns and citation analysis are commonly addressed.

Using the AMiner [2], [3] and DBLP datasets, this work aims to analyse and discover interesting patterns among communities of scientific entities. This is achieved by addressing the following problems:

1) Find dense communities within the publications network. The aim is to discover dense communities between entities such as authors, papers and their venues/conferences/journals, given a set of keywords or a set of fields of study. The objectives are as follows:
    a) construct a graph which captures the publications network and use an algorithm to detect communities between entities;
    b) devise a metric to gauge the strength of the discovered communities;
    c) evaluate the strength of collaboration between multiple communities given different sets of keywords/fields of study.
2) Track the dynamics in the publications network. The aim is to analyse and visualise how authorship patterns change over time.
    a) evaluate whether joint authorship is common in academic research;
    b) investigate how single and joint authorship changes over time using different metrics (replicating some of the techniques used by [4]);
    c) visualise the results to show differences between single versus joint authorship and how they change over time.
3) Discover and visualise associations between entities. The aim is to find associations between authors to keywords and recommend potential collaborators. The objectives are as follows:
    a) find associations between keywords and authors by finding a relationship between the papers published (using the title) by the author and the defined keywords;
    b) cluster authors by the relationship to the keywords using a graph clustering algorithm and recommend potential collaborations based on the output of the clustering technique

II. Related Research

A. Dense Communities within the Publications Network

Community detection has been studied for several years in different domains [5], [6], [7], especially with the spread of online social networks where users can connect with other users giving rise to complex relationship structures, similarly to the complex relationship between scientific entities collaborating with each other in scientific research. Intuitively one can think of these structures as hierarchical network map or tree (known as dendrogram) where the leaves are the nodes and the branches joining them are edges. [9] examined the problem of real-time community detection in large-scale networks. Although they achieved significant performance improvement when compared to modularity-based algorithms, their paper did no analyse changing behaviours in the community networks and failed to address local changes to the structure without the need for a global update.

[7] provides an innovative algorithm that facilities community detection in large networks by using entropy to measure the information within the network through simulation analysis. They use an established citation network of papers published in *Scientometrics*. The proposed model was lower in complexity when compared to 13 other existing community detection methods and achieved higher accuracy.

[8] tackled the problem of dense communities using graph partitioning or clustering to extract meaningful dense subgraphs. They proposed a dense subgraph extraction approach for three types of graph, undirected, directed and bipartite. Their algorithm utilised cosine similarity of matrix columns and built a hierarchy for the graph vertices while computing partial clustering on them. The clusters represented a dense subgraph. The innovation in this approach (demostrated on a collaboration network) was that the algorithm considered the fact that some nodes did not belong to a community.

B. Dynamics in the Publications Network

Collaborative strength and authorship patterns among professional communities is a topical area of research. [10] investigated different patterns of collaboration based on citations of Harvard University publications (published between 2000 and 2009) and found significant positive correlation between the number of authors and the number of citations. Furthermore, they reported a positive influence of scientific collaboration on research impact and that by increasing the number of institutions collaborating on a project increased the impact of that project on the community. They analysed single versus joint authorship publications and found that only 12% of the publications were single authored.

On tracking publication dynamics, in a case study on agricultural engineers in Nigeria, [4] analysed 589 publications

(published between 2000 and 2010) based on the number of articles published per year, pattern of authorship featured in the NIAE proceedings and the collaborative degree and strength of the authors. Although the dataset was relatively small, their methods of year-wise distribution of article publication and co-authorship of papers, revealed that the research was dominated by a small number of productive authors and that these authors should encourage the low productive ones to contribute more. They also conducted analysis of single authored and joint-authored works and found that single-authored works remained constant during the study period whilst the percentage of joint-authored publications increased gradually by time. Single-authored publications were also significantly less than join-authored ones, which is in line with the findings of [10]. They also measured the average number of authors per paper (between 1.9 and 2.7) and the degree of collaboration of authors which they found to be high.

Similar methods were also used in another bibliometric study conducted by [11], where the researchers analysed authorship trends and collaborative patterns using a dataset of publications published in a Chinese journal between 1996 and early 2008. In this study, the average number of authors per paper was 1.61 which is close to the result obtained in [4], although the degree of collaboration was 0.443 which is quite low when compared to the other paper where they obtained a value of 0.715. This difference between the degree of collaboration could be attributed to the fact that [4] focused on the field of agricultural engineering whilst this paper focused on various fields of library and information science.

[12] used citation analysis to carry out a quantitative bibliometric analysis of the growth of literature in the medical field, specifically on Hepatitis C. They too used year-wise calculations (amongst others) on several thousand cited articles in a medical journal from the period of 2006 to 2010. Their interest was to study trends using the bibliometric indicators, Relative Growth Rate (RGR) and Doubling Time (DT). The former refers to the increase in the number of articles or pages per unit of time, whilst the latter refers to the time required for the number of items to double in number in year 1. They found fluctuating trends using the year- wise analysis of RGR and DT for citations in Hepatitis C research throughout the study period.

On the field of information technology publications, [14] analysed around 18,000 publications from the Library and Information Science Abstracts (LISA). They performed language and year-wise analysis using RGR and DT like the research done by [12] and found that the number of co-authored articles is significantly higher than single authored ones. This result is in line with other research [4] even though the study was conducted on publications from a different field of study.

C. Association and correlation analysis within the Publications Network

Recommending future collaboration between authors is an active area of research and some authors tackle this problem using machine learning techniques. [13] proposed a supervised learning approach to predict links in co-authorship network and presented an approach based on mining the evolution of co-authorship networks. They experimented with real bibliographical data using the bipartite nature of the publication network to enhance the performance of link prediction models. They showed that measuring the likelihood of a link between two nodes that are in the dual graph enhanced prediction precision.

III. Methodology

In this section we describe in detail the design of our system solution, how we implemented it and the challenges encountered along the way.

A. Datasets

In this work we use the AMiner dataset [2], [3] which consists of 154,771,162 papers. Each paper is structured as a JSON object, where each line in the data files (.txt) describe a specific publication. This dataset is sourced from https://aminer.org/open-academic-graph where a detailed description of the data schema is provided. For the purpose of achieving the objectives, from the whole schema we are utilising only these attributes: *id, title, authors.name, venue, year, keywords, fos, lang, doi and publisher*.

Additionally, we are also using the DBLP dataset, compromising only computer science publications, primarily to demonstrate that our solution caters for more than one dataset with a different schema, structure and size. Each paper in this dataset is structured as an XML element and the dataset is sourced from https://dblp.uni-trier.de/xml, where a detailed description of the data schema is also provided.

The size of the AMiner dataset is 38GB whilst the total size of the DBLP dataset is 2.6GB. Utilising both datasets which have a different schema and structure, we will be able to demonstrate that SciPi can handle a variety of data coming from different sources.

B. Solution Design

The solution we choose for addressing the problems described in Section I, is composed of a mix of technologies as illustrated in Fig.1. Apache Kafka is used to push/produce data streams from the datasets described in Section III-A. Then Apache Flink is used to process these data streams based on the requirements specified. We also use Flink batch processing and Flink's Gelly which is a graph-processing API. Also, we use Apache Cassandra database to persist processed data and results.

Data is streamed using Apache Kafka, a distributed messaging system capable of handling high-velocity and high-volume data streams. This technology will allow for building real-time data pipelines which will be later realized/processed in real-time using Flink's streaming API. For the purpose of this project we will only use Kafka's streaming API which allows us to publish a stream of records to Kafka topics, which in our case is the publications found in the datasets. Communication between Kafka and Flink instances is done using a binary protocol over TCP. Another Apache product called Zookeeper

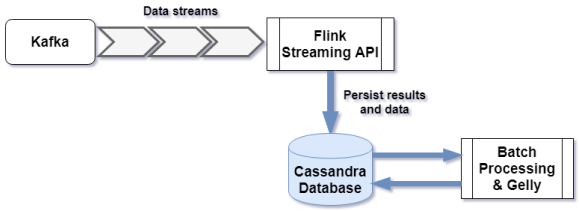

Fig. 1: Overview of the process flow. Kafka is used to push records to the data streams. Flink's streaming API will consume and process data streams in real-time. Cassandra DB will be used to persist results and processed data, while Flink's batch and Gelly API will process the data which is persisted on Cassandra.

is also used alongside Kafka, to monitor instances/brokers via a mechanism called Heartbeat. Amongst other configuration management of the Kafka distributed cluster, it also makes sure that the Kafka cluster is available even if the leader node fails via a mechanism called leader election.

The data streams produced by Kafka brokers are consumed in Flink via the streaming API. The data is realized/processed in real-time and the results/processed data is persisted to Cassandra via a data sink mechanism. The advantages of using Kafka as a producer and Flink as a consumer is mainly attributed to its ability to guarantee exactly once delivery of events, the high rate in throughput and the fact that it eliminates any problems caused by backpressure since Kafka streams do not use a backpressure mechanism.

Cassandra accommodates different data formats such as structured, unstructured or semi-structured and provides very fast and scalable read/write operations which works well with our solution design. It also provides high availability and fault tolerance to the distributed nature of this technology. Refer to the Section III-F for the motivations in choosing Cassandra to persist data.

Flink's batch processing API will be used to process the persisted data found in Cassandra, once the streaming process is done. Flink's Gelly API will also be used to process the persisted data, as some problems which we are dealing with are better suited to be computed via a Graph structure. Since we do not have an infinite stream and some results require for all the data streams to be realized, this part of the process flow is executed once the streaming process is finished. One of the main reasons for choosing Apache Flink over other technologies such as Apache Spark, is because Flink was built from the ground up as a streaming product unlike Spark which simulates streaming by using micro batches. Another reason is the rich graph API found on Flink which provides various different graph algorithms and also provides a graph structure for bipartite graphs.

Finally, the results will be outputted/visualised using python and various libraries together with an interactive Jupyter notebook. An end user will be able to provide an input and the results will be displayed/visualised on the Jupyter notebook.

C. Database Design

In Cassandra, a keyspace called *scipi* is created to hold the results of the stream processing. These are the tables found in the *scipi* keyspace:

1) *scipi.publications*: saves processed publications;
2) *scipi.keywords*: saves keyword count;
3) *scipi.field_study*: holds the field of study count;
4) *scipi.yrwisedist*: holds single-authored versus co-authored publications year wise distribution;
5) *scipi.authorptrn*: persists authorship patterns;
6) *scipi.aap*: holds the average number of authors per paper (AAP);
7) *scipi.hyper_authorship*: holds the count by year for papers with more than 100 authors.

For a more detailed definition of the database schema such as columns, primary keys, clustering keys and so on, a cql script is provided in our project's GitHub repository page which is documented at the end of this paper.

D. Local and Cloud Setup

In this section we provide a brief description of both the local and cloud setup.

1) *Local/Development Setup*
   The development of the solution was done on Ubuntu 18.04.2 using Java for all the streaming and batch processing coding and Python for the visualisations. IntelliJ IDEA was the main development tool for Java whilst for Python we used Jupyter Notebook. As pre-requisities for Apache Flink we set up Maven (version 3.2.5, this was done as Flink was built from source) and Java 8 runtime engine. Apache Flink is built from source.
   As a contribution to the community we have prepared a detailed getting-started guide for Apache Flink using Java by collecting information from various websites and online literature. We made this guide available on GitHub (https://github.com/achmand/flink-java-tutorials) for future work.
   Flink provides a graph processing library, Gelly, to simplify the development of graph analysis. Gelly is available as part of the Flink libraries so we just added as a Maven dependency in our Java code to use it. There was no additional setup to be done.
   All the development and testing of the data flows was done on Ubuntu machines prior to deploying the solution on Amazon Web Services (AWS) Cloud.
2) *Cloud Setup (AWS)*
   A total of ten nodes are deployed on AWS Elastic Compute Cloud (EC2): three nodes for the Zookeeper-Kafka cluster, one node for Kafka tools, four nodes for the Flink cluster and three nodes for Cassandra database cluster. Table I summarises the specification of each node.
   Three Kafka servers (or brokers) are deployed first on three separate EC2 instances. Together they make up the Kafka cluster. ZooKeeper is a separate service which provides highly reliable distributed coordination via leader election and light-weight consistent state storage. Leader election is required in the event that the leading Zookeeper server goes down. Some other functions include locking and keeping track of Kafka brokers

TABLE I: Specifications of AWS EC2 nodes

Node	Purpose	Type	vCPUs	Memory (GB)	Volume (GB)
1	Kafka-Zk	t2.medium	2	4	200
2	Kafka-Zk	t2.medium	2	4	200
3	Kafka-Zk	t2.medium	2	4	200
4	Kafka tools	t2.small	1	2	8
5	Flink-Zk (Mas)	m4.large	2	8	20
6	Flink-Zk	c4.xlarge	2	8	20
7	Flink-Zk	c4.xlarge	2	8	20
8	Flink-Zk	c4.xlarge	2	8	20
9	Cassandra	t2.xlarge	4	16	200
10	Cassandra	t2.xlarge	4	16	200
11	Cassandra	t2.xlarge	4	16	200

via a heart-beat mechanism to ensure that all brokers are still alive. The Zookeeper leader maintains all the communication with the other two follower servers and the three Kafka brokers. All the tools that are needed in order to administer and monitor the Kafka brokers and Zookeeper servers are set up in the Kafka tools node on another EC2 instance. Figure 2 illustrates the Kafka cluster and web tools node.

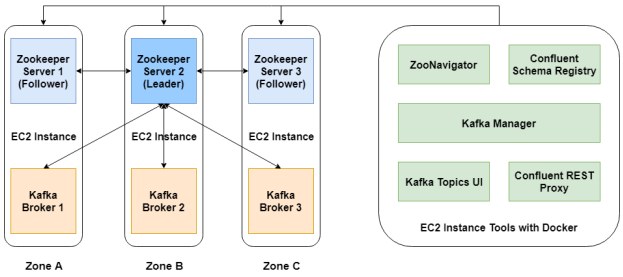

Fig. 2: Cluster with Kafka and Kafka tools EC2 instances. Zone A, B and C are 3 high availability EC2 instances each hosting Zookeeper and Kafka brokers. The Zookeeper setup is made up of one leader and two followers. Another EC2 instance, Tools with Docker, hosts all tools that are needed to control, manage and administer the Kafka brokers and Zookeeper servers.

The Flink cluster is deployed on three EC2 instances using Elastic MapReduce (EMR). Flink expects the cluster to consist of one master node (JobManager) and one or more worker nodes (TaskManagers). The JobManager's primary responsibility is the coordination of all Flink deployments, such as scheduling and appropriate resource allocation. The TaskManagers execute the tasks of a dataflow, and buffer and exchange data streams. The Flink cluster is used for both the streaming and the batch/graph processing. Figure 3 illustrates the Flink cluster.

The Cassandra database cluster is deployed on another three EC2 instances on AWS Cloud with slightly higher specifications than the other two clusters (Table I). In this configuration Cassandra replicates data across the three nodes to ensure reliability and fault tolerance. The Cassandra cluster is set up with a replication factor of three meaning that there are three copies of each row and each copy is on a different node. One of the cluster nodes serves as the seed node. The seed node bootstraps

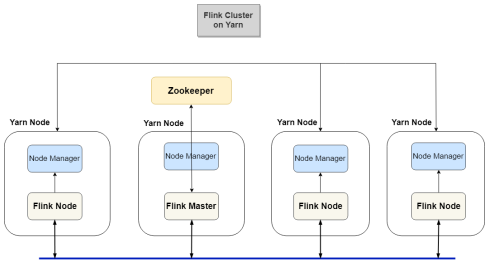

Fig. 3: Flink on Yarn Cluster.

the gossip (or communication) process for new nodes joining the cluster. Refer to Figure 4.

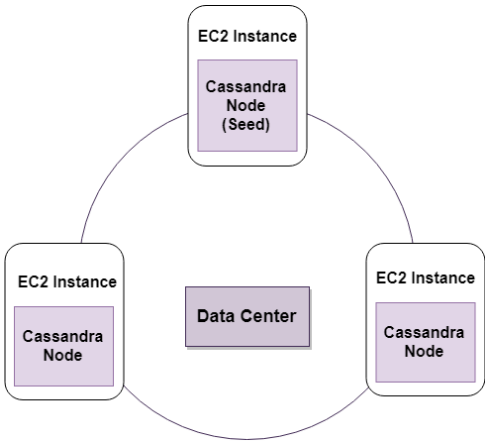

Fig. 4: Cassandra Cluster. Data is replicated across three separate nodes to ensure reliability and fault tolerance.

A representation of how the data flows across the three clusters is shown in Figure 5 starting with the Kafka cluster streaming the data into the Flink cluster. Flink validates and persists data into Cassandra database where it's replicates asynchronously across its three nodes. Data is queried back into the Flink cluster again for further processing and visualisation

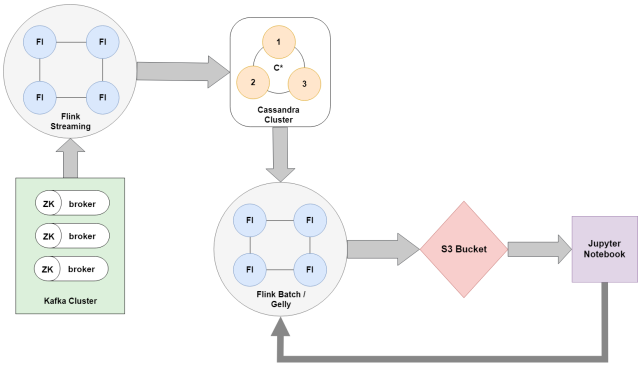

Fig. 5: Process flow between clusters. The arrows indicate the flow of data from one cluster to another.

E. Implementation

In this section we will provide a more in-depth description for the streaming and batch process procedures.

1) *Data Perperation and Pre-processing*
Since the analysis is based mainly on keywords, paper titles, authors and fields of study, the data is filtered so that only clean and complete publication records are persisted in the database. This also facilitates graph construction in the batch processing part. The pre-processing is carried out by Flink prior to persisting the data in the database. The validations are the following: (1) the publication is written in english; (2) the doi field is not null; (3) the publication title is not empty; (4) the publication has a publisher or a venue; (5) the publication has at least one keyword or one field of study; (6) the publication has a valid year; and (7) the publication has at least one author. Any entries that do not satisfy these conditions are not persisted.

2) *Stream Process Flow*
Figure 6 is an illustration of the process flow of the data stream from consumption to persistence in the Cassandra database. The data was streamed into two different topics found on the Kafka cluster. One named *oag* and the other named *dblp*, one to hold the data coming from AMiner and the other coming from DBLP. Records coming from AMiner where streamed using the console producer, where a bash script was written to go through all text files and streamed line by line (each line is a specific publication). On the other hand, the DBLP dataset required more work as it was not possible to stream each element using the native console producer since its in XML format. For this a JAR (scipikafka) was created to parse the XML into JSON and stream each record to the Kafka brokers.
The JSON strings hitting Kafka are passed to the Flink stream. These strings are mapped to instances of publication object. After that, Flink processes and validates the object and its data sink emits the publication object to the Cassandra database. Next we map our publication instance to Tuple<str, int>containing the keyword and its occurrence within the current stream. The occurrence count for keyword is then persisted to the database using data sink in table *scipi.keywords*. The field of study occurrence count is done in the same way. The table for the field of study counts is *scipi.fields_study*. These results for both keywords and fields of study will be later processed in the batch part of the implementation and displayed on the Jupyter Notebook. The reason for this is to provide the end user with some options for the input which will be described in a later stage.
The next step is the preparation of the year-wise publication distribution, the authorship patterns and the average author per paper. To produce the year-wise distribution we adopt the same approach as [4], [10] and [15]. The results are stored in table *scipi.yrwisedist*. These results will help us to tackle problem 2 described in the Section I and we will be to compare the results to the ones obtained by [4] and [10]. For this procedure a JAR called *scipi_stream.jar* was implemented to facilitate the requirements to tackle our problems mainly dynamics in the publications network. Any parameters passed as input to this JAR are well documented in the *ScipiStream.java* which can be found on our repository page.

3) *Batch Process Flow*
In this section several JARS were implemented to tackle the problems described in Section I, mainly it focuses on problem 1 and 2.
Since Gelly does not support stream processing this part was implemented once the data has finished streaming. This was also required to gather precise results as we need to process all the data which was streamed to gather our results and to be able to compare with other studies. All these steps which will be described are fired up from the Jupyter Notebook via bash scripts where the user is able to input keywords/fields of study and execute each JAR (results are also shown, once processing is done).
First off, the community detection problem is fired once the user defines a set of keywords or fields of study. The JAR file scipi_community.jar is executed. In this implementation the graph algorithm called CommunityDetection which is provided by Gelly is executed to detect various communities. This algorithm is based on the research done by [9]. To apply this algorithm the publications data is loaded from Cassandra and mapped to edges and vertices. Only publications which map the user input are considered, for example if a set of keywords are passed, only publications which contain that keyword/s are processed.
Several processes such as setting an initial label to the vertices and connecting each vertex based on a relationship was required to be able to feed this graph to the algorithm. The number of iterations until the algorithm terminates was passed as an input which was set to 30 and a delta of 0.5, both of which are required by the algorithm. Once the algorithm terminates, we filter out dense community by counting the labels and accepting only communities which have several vertices with each label. Each label denotes a community. The threshold which we set to define a dense community was set to 1000.
The result is multiple subgraphs (made up of authors, papers, venues and publishers) which belong to different communities based on the user input. We also applied a weighted average based on the number of dense communities found, given a certain input. This will allow us to quantify how strong the collaboration between this community (made up of multiple communities) is. Lastly a sample of the subgraphs (top 3 dense communities) is taken and displayed as a graph network on the Jupyter Notebook.
After this another user input is required (set of keywords) to fire up the second JAR (again through a bash script) created called scipi_association.jar which deals with problem 3. This implementation also uses Gelly the graph API provided by Flink. In the first bit of the process the publications are again retrieved from Cassandra. To find the association between the author

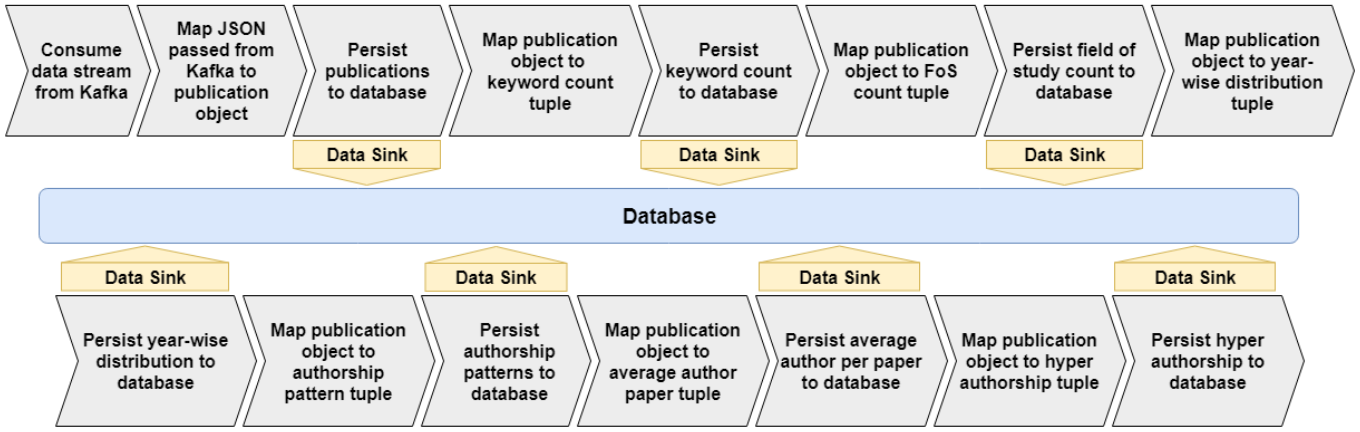

Fig. 6: Steam process flow from consumption to persistence

and keyword we applied a modification of the cosine similarity between the user defined input and to the titles of any publications published by that author. If the score is greater than a threshold (set as 0.3) we create an association between the author and keyword. A count is applied to the number of associations to the keyword and filtered again by another threshold to find strong associations. A sample of the result is saved to an S3 bucket and displayed to the Jupyter Notebook.

The second part of this implementation deals with finding potential collaborators. To do so a bipartite graph is created with two types of vertices, one for the author and the other for keywords. An edge is created if the author used the keyword/s passed as input in one of his/hers publication/s. The number of instances the author used that keyword is computed (using reduce function and setting this count to the weight of the edge) and then it is filtered by a threshold to create a bipartite graph with a strong relationship. After doing this we will extract a normal Graph structure where each vertex is an author and there is an edge between them if both authors were connected to the same keyword (using top projection in Graph API). This procedure can be visualised in Figure 7. By doing so we clustered the authors based on the keywords, and we recommend a potential collaboration if there is an edge between any two authors. A sample of the result is saved to S3 bucket and displayed into the Jupyter notebook.

An additional JAR file was also implemented which do not deal with any problem being tackled. This JAR file called scipi_topics.jar take a number as an input and filters out any keyword/field of study with an occurrence greater than that input (previously processed in the streaming part). The result is then saved to a S3 bucket and displayed in the notebook. This helps the end user to know what options (mostly used) is more likely to get interesting results in the procedures previously discussed.

All input parameters required by each JAR file are documented in each class which is also found in this projects repository page. All results described above were displayed on the Jupyter notebook.

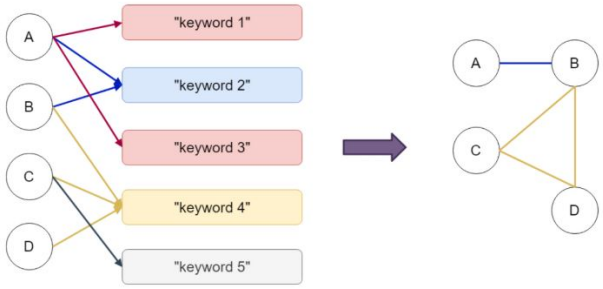

Fig. 7: Potential collaboration. A, B, C and D represent authors. Based on their association with different keywords we can generate collaboration patterns between the authors themselves.

F. Challenges

These are the main challenges encountered while working on the assignment.

- **Choosing the right database engine.** Our original design suggested the used of MongoDB since this was used in the study unit practical session. Both databases can store structured and unstructured data, however as we progressed in the implementation of the solution a number of issues started to surface which lead us to consider alternatives. In the end we opted for Cassandra. The motivations were:
  - Data sink mechanism. One of the issues we encountered with MongoDB was the data sink there was no native Data Sink implemented for Flink. Although Flink allows to write custom data sinks this did not allow for records to not be persisted more than one time if something fails (may have more than one record refereeing to the same instance if failing occurs while stream processing).
  - Query language. Cassandra query language syntax is almost identical to the standard SQL syntax for querying, inserting and updating records. MongoDB

query language syntax is proprietary and unconventional, for instance, *find()* is used instead of *select*.

- Part of Apache suite. Cassandra is maintained by Apache and therefore it integrates neatly with Kafka-Flink-Gelly stack since these are also Apache initiatives.
  In terms of product support, active community and online documentation, both Cassandra and MongoDB are both extensively covered.

- **Recommending potential collaborations** proved to be too computationally heavy when constructing the vertices and the edges. To overcome the performance problem we made use of bipartite graph and bipartite edges. Like this we managed to reduce computational complexity. By recreating the same functionality of the bipartite graph with a normal graph yielded to a huge improvement in performance (mostly due to bottleneck in GroupReduce function). In the end the projection method in the bipartite graph data structure was used and achieved the same result in a less than a minute rather than 20 minutes (when tested locally using a sample of the data set).
- **Finding associations between authors and keywords.** We wanted to find a measure of similarity between keyword and one or more titles which could lead us to infer potential associations between that keyword and the authors of the similar titles. At first we considered the use of Jaccard distance and Cosine distance similarity measures. Eventually we opted to use a modification of the Cosine similarity which is quite a simple/naive approach.

IV. Results

In this section we elaborate on the results obtained using the proposed solution.

In order to have an indication what input values can be supplied to retrive a good visual a count on keywords and fields of study is done. Table III and table II give top 10 and top 4 keywords and fields-of-study counts in descending order.

TABLE II: Top 4 fields of study

Field	Count
medicine	5478
biology	5065
physics	3963
chemistry	3876

A. Community Detection

To detect dence communities we construct graphs for a give set of keywords and fields of study. Figure 8 and figure 9 show two samples of 200 connected entities from the top 3 dense communities (shown in different colours blue, green and red) using as input bioinformatics and computer science in the first sample and medicine and physics in the second sample. Different communities are shown in different colours according to the label of the vertex. The strength between the communities for the first sample (figure 8) using a weighted average approach is 2.37. The community strength for the second sample (figure 9) is 36.81. Since we are only showing a sample for ease of visualisation the positioning of the lables has no bearing on the strength measure.

TABLE III: Top 10 keywords

Keyword	Count
computer science	2338883
bioinformatics	283518
genetics	245483
biomedical research	229531
kinetics	221368
enzyme	180261
spectrum	178058
null	163435
mathematical model	163121
indexation	135471

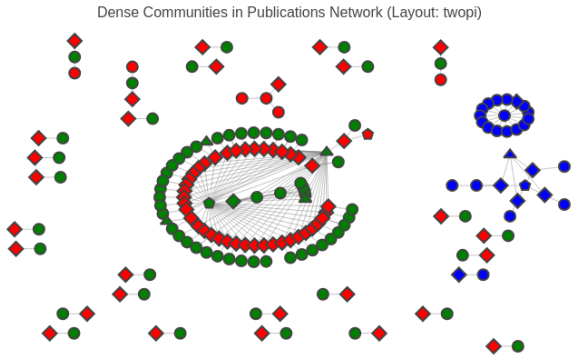

Fig. 8: Dense community of entities for the keywords bioinformatics and computer science. This graph is based on a sample of 200 entities where circles signify authors, venues are denoted by a triange, publishers are shown as pentagons and publications are displayed as a diamond. The layout used for visualisation is twopi.

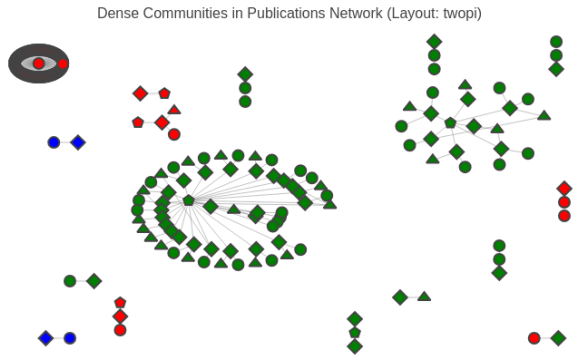

Fig. 9: Dense community based on a sample of 200 edges where circles signify authors, venues are denoted by a triange, publishers are shown as pentagons and publications are visualised by a diamond. The layout used for visualisation is twopi.

B. Dynamics in authorship patterns

To track the dynamics in authorship we first evaluate the degree of joint-authorship in the network being examined. We

find that 60% of the publications are joint-authored by a team of up to 3 authors. A further one-third of the publications is co-authored by 4 to 6 persons as shown in 10 The highest number of co-authored publications are written by 2 individuals. Table IV shows the number of publications co-authored and their cumulative percentage of the total publications. The percentage of total co-authored publications decays exponentially as the team of writers increases. It can be observed also that only 16% of the publications are written by a single person. Thus the results are indicative of a strong authorship pattern.

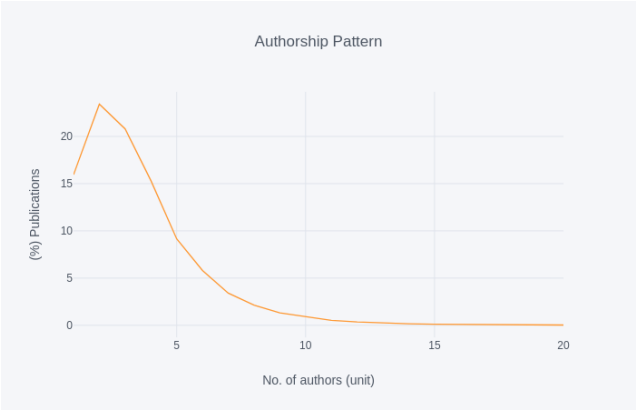

Fig. 10: Authorship pattern

TABLE IV: Authorship pattern

No. of Authors (Unit)	No. of Publications	No. of Authors	% of Total Publications	Cumulative % of Total Publications
1	3219689	3219686	15.95	15.95
2	4727571	9455142	23.42	39.37
3	4200224	12600672	20.80	60.17
4	3093436	12373744	15.32	75.49
5	1848747	9243735	9.16	84.65
6	1172453	7034718	5.81	90.46
7	689197	4824379	3.41	93.87
8	432432	3459456	2.14	96.01
9	265016	2385144	1.31	97.32
10	179884	1798840	0.89	98.21
11	107315	1180465	0.53	98.74
12	72631	871572	0.36	99.10
13	46247	601211	0.23	99.33
14	32011	448154	0.16	99.49
15	22442	336630	0.11	99.60
16	15960	255360	0.08	99.68
17	11519	195823	0.06	99.74
18	8745	157410	0.04	99.78
19	6684	126996	0.03	99.81
20	5641	112820	0.03	99.84
21+	32299	1728898	0.16	100.00

We next compute a year-wise distribution of co-authorship and find that during the period 2000 and 2019, whilst single-authored papers remain stable with minimum variation along the years, there was a continuous increase in joint-authored papers up to 2010. Thereafter joint-authorship started to decline again and consistently reduce in popularity as illustrated in Figure 11. In figure 12 one can observe again that the percentage of co-authored publications by far out-number the single-authored ones consistently over whole period and that they are diverging. Table V summarises our findings.

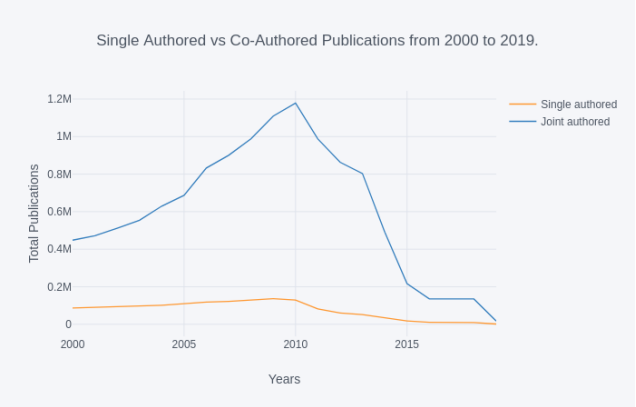

Fig. 11: Single-authorship vs. joint-authorship

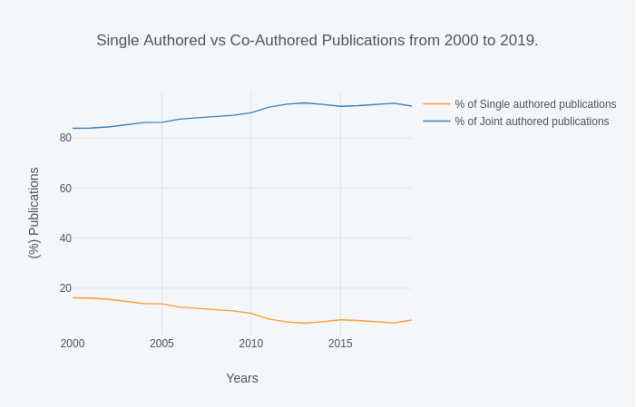

Fig. 12: Single-authorship vs. joint-authorship percentage

To follow the calculations done by [4] we compute the average number of authors per paper during the same period. We observe that the highest average number of authors per paper is 4.48 in the year 2012 and lowest is 3.53 in 2000. There was a consistent increase in co-authored papers up to 2012 then went down in the following years. This results is consistent with [10] who showed that the average author per paper grew expontially up to the year 2010. Although in 2010 there was a surge in the total number of co-authored papers and the total number of authorship, the average was not the highest. The highest average achieved in 2012 was perhaps due to the reduction in the total number of papers published. See illustration in figure 13 and table VI.

As a further analysis on the dynamics of authorship patterns over time we attempt to find papers that have more than 100 authors following [**?**]. Once again we see that there is a high number of papers in 2010 that fall under this category. Papers attracting such a large audience of writers seem to follow the same trend of increasing consistently to around the year 2010 then decaying after that year (figure 14).

C. Associations between authors and keywords

A potential association between keyword and author(s) is found by using a cosine similarity algorithm (adopted from

TABLE V: Year-wise distribution of single- and multi-authored papers

Year	Single-authored	Joint-authored	Total Publications	% of Single-authored	% of Joint-authored
2000	86249	448118	534367	16.14	83.85
2001	90129	471487	561616	16.05	83.95
2002	94597	512134	606731	15.59	84.41
2003	96109	554191	560300	14.78	85.22
2004	100867	629399	730266	13.81	86.19
2005	109305	686819	796124	13.73	86.27
2006	118505	832702	951207	12.46	87.54
2007	122116	900335	1022451	11.94	88.06
2008	129304	988485	1117789	11.57	88.43
2009	136749	1109378	1246127	10.97	89.03
2010	129421	1178156	1307577	9.90	90.10
2011	82226	987056	1069282	7.69	92.31
2012	59815	863501	923316	6.48	93.52
2013	51322	803504	854826	6.00	94.00
2014	34838	491244	526082	6.62	93.38
2015	17236	217027	234263	7.36	92.64
2016	10313	135047	145360	7.09	92.91
2017	9662	136587	146249	6.61	93.39
2018	8805	134835	143640	6.13	93.87
2019	1334	16986	18320	7.28	92.72

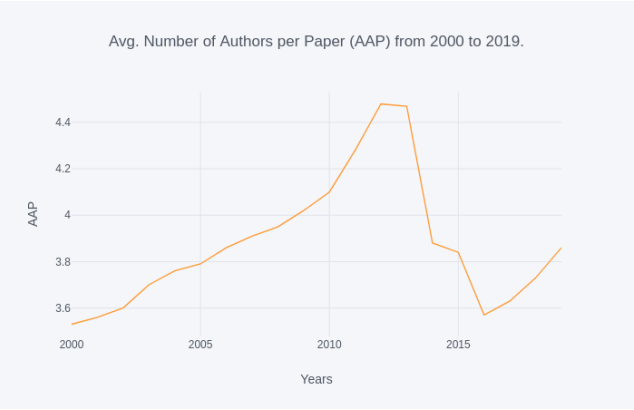

Fig. 13: Average number of authors per paper.

Fig. 14: Hyper authorship

https://github.com/tdebatty/java-string-similarity) between keyword and paper title. The authors whose connected titles produce the highest similarity score are then associated to the keyword. Figure 15 shows the resulting association network between an author and a number of keywords. In addition based on the authors' associations with keywords collaboration patterns between authors can be generated. A graph depicting the potenial collaborators of a researcher is shown in figure 16. As an example we generated the potential collaborators of the author A. Gabriella Wernicke (figure 17).

TABLE VI: Average number of authors per paper

Year	Total no. of papers (P)	Total no. of authorship (A)	Average no. of authors per paper (AAP = A/P)
2000	534367	1885533	3.53
2001	561616	1998139	3.56
2002	606731	2186265	3.60
2003	650300	2403064	3.70
2004	730266	2745228	3.76
2005	796124	3020370	3.79
2006	951207	3672885	3.86
2007	1022451	3997230	3.91
2008	1117789	4418572	3.95
2009	1246127	5015096	4.02
2010	1307577	5362812	4.10
2011	1069282	4581463	4.28
2012	923316	4132890	4.48
2013	854826	3817244	4.47
2014	526082	2039555	3.88
2015	234263	899476	3.84
2016	145360	518670	3.57
2017	146249	531486	3.63
2018	143640	536295	3.73
2019	18320	70785	3.86

V. Conclusion and Future Work

Finally we present our conclusion and the potential future improvements on our implementation.

A. Conclusion

In the study we used the AMiner and DBLP datasets and attempted to discover dense communities by constructing a

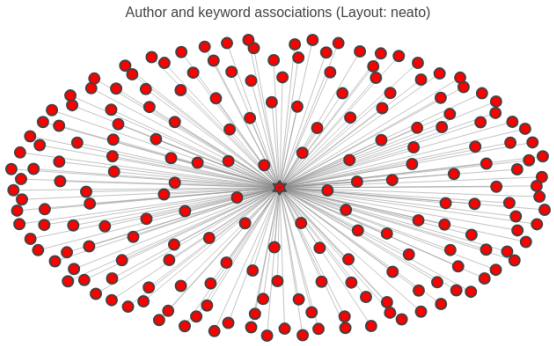

Fig. 15: Associations between authur and keywords. The keyword is depicted as a hexagram in the centre and the authors are shown as circles with edges connecting them to the keyword. They are of the same colour because they have the same label. The layout used for this graph is neato.

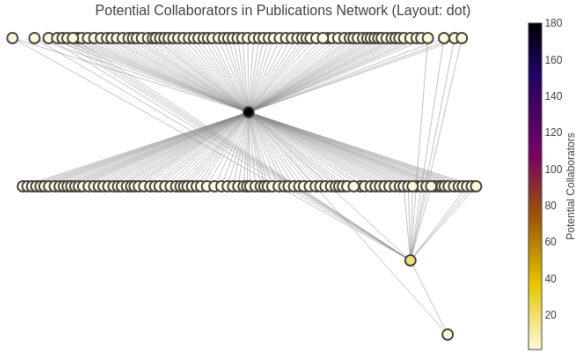

Fig. 16: Potential collaboration between authors

Author	Potential Collaborators	No. Collaborators
a gabriella wernicke	tetsuo ohta | theodore a stern | edwin r bickerstaff | glenn e m maguire | clovis a silva | xueji zhang | fei gu | j b gurney smith | simon lammy | jie huang | s smith | tarun varma | jing li | gang chen | edward r t tiekink | jane roberts | ping sun | m milanov | alla lapidus | wilfrid mills | itasu ninomiya | morton ann gernsbacher | hoongkun fun | jung min lee | wei wu | andreas g ladurner | bhupesh parashar | thavendran govender | richard d feinman | muhammad nawaz tahir | baochang cai | miriam land | michelle r munson | silvia manzanero | heather l van epps | wei li | jennifer l ellis | mehmet akkurt | mariadhas valan arasu | hui lu | hao liu | francisco azuaje | lei sun | naomi laventhal | v todorova | nicole lebrasseur | lynne goodwin | hendrik g kruger | gkhan amaz | serpil yaylaci | mustafa serinken | min li | leonidas stamatatos | leonard g parsons | seik weng ng | g georgiev | byeng wha son | mitch leslie | daniel b reeves | benjamin blondel | william a wells | sophia r newcomer | david bruce | jan fang cheng | natalia ivanova | xin zhou | hong dae choi | uk lee | maria carolina hardoy | marianne haapea | yanping chen | charlotte m deane | t garrett horder | lars r furenlid | gordon li | uday c ghoshal | yuan wang | qinyong hu | arshed a quyyumi | suchada	180

Fig. 17: Potential collaboration between authors

graph to detect communities between entities. By allowing the user to interact with the system by entering a set of keywords or fields of study, we evaluate the strength of the collaboration between communities using an weighted average method.

Next we addressed co-authorship trends and how they changed over time adopting the same techniques as in previous studies, namely, year-wise distribution of single-authored and joint-authored publications, and average author per published paper. We observed the same trends as previous studies even though in different domains. Joint-authorship grew exponentially up to 2010 and then started to decline. The average author per paper was highest in 2012.

TABLE VII: Paper writing goes hyper. Number of papers that have more than 100 authors.

Year	No. of publications
2000	79
2001	62
2002	108
2003	96
2004	172
2005	161
2006	181
2007	207
2008	174
2009	229
2010	230
2011	149
2012	170
2013	44
2014	38
2015	8
2016	1
2017	1
2018	2

Finally we find, using a similarity approach, associations between keywords and authors. We visualise clusters of authors by their relationship to keyword using a graph clustering algorithm.

Some of the strong points in this study are: (1) we adopt techniques found in previous studies in different domains to track dynamics of co-authorship over time. Our results are in line with previous studies; (2) We analysed hyper authorship (papers with over 100 authors) and found that hyper authorship increased up to 2010 and then started to decline. However there were years where the numbers were relatively very low which could mean that there is missing data in our dataset.

As weak points in our approach: (1) we idenitified entities by name which means that authors with same initials could be referred to the same person; (2) the community detection algorithm is optimised when compared to other algorithms as decribed in [9] however if a change needs to be done in network it would require a recomputation of the whole graph structure.

B. Future Work

Although with the implemented solution we managed to achieve most of the objectives outlined in this assignment, a number of improvements can be identified to render it more comprehensive. These are:

1) **Gelly streaming model.** An alternative approach to our batch processing of graphs is Gelly Streaming. Online graph streaming API's, such as Gelly Streaming, is an active area of research and a lightweight distributed graph streaming model is available but it is still in experimental stage (https://github.com/vasia/gelly-streaming). It would be interesting to augment our solution with a graph streaming approach and compare analytics.

2) **Graph persistence.** The system is calculating and rendering the vertices and edges of the graphs on the fly thus taxing the overall performance of the system. Persisting the graphs in a graph database management system such as Neo4j, would potentially make the system work faster and offer a better user experience.
3) **More sophisticated similarity metrics.** In our model we utilised the Cosine similarity metric to find any similarity between keyword and title. An enhancement in this area would be to adopt more sophisticated distance metrics such as those investigated in [16]. FlinkML provides a way to implement any custom distance or similarity metric, by implementing the *DistanceMetric* trait, to improve the accuracy of the similarity approach.
4) **Accept publications with different languages.** Many papers are written in many languages other than English. In our approach we focused on English language publications. An improvement on this work would be the handling of publications with different languages and the analysis of any language patterns between authors.
5) **Real-time visualisations.** Finally the way visualisations are rendered could be improved by implementing a real-time framework that can enable faster processing of graphs which is more suitable for big data processing such as Kibana and Elastisearch.
6) **Better identification of entities** A better way to identify different entities should be explored since relying on just the entity name is not scaleable.

All source code for this work can be found in: https://github.com/achmand/SciPi.

REFERENCES

[1] L. Leydesdorff and S. Milojevic, *Scientometrics*, In J.D.Wright, M. Lynch, et al. (Eds.), The International encyclopedia of social and behavioral sciences, 2nd Ed., Sec 8.5:Science and Technolgoy Studies, Subsection 85030, Elsevier, 2015

[2] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang and Z. Su, *ArnetMiner: Extraction and mining of academic social networks*, In Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'2008), pp.990-998.

[3] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J.(P.) Hsu and K. Wang, *An overview of Microsoft Academic Service (MAS) and applications*, In Proceedings of the 24th International Conference on World Wide Web (WWW'15 Companion). ACM, New York, NY, USA, pp.243-246, 2015.

[4] J.O. Oyeniyi and T.P. Olaifa, *Collaborative strength and pattern of authorship among agricultural engineers in Nigeria: A case study of 2000-2010 NIAE proceedings*, International Journal of Library and Information Science, vol.4, no.6, pp.115-120, 2012

[5] G.W Flake, S. Lawrence, C.L. Giles, F.M. Coetzee, *Self-organization and identification of web communities*, Computer, vol.35(3), pp.66-70, 2002.

[6] E. Ravasz, A.L. Somera, D.A. Mongru, Z.N. Oltvai and A.-L. Barabasi, *Hierarchical organization of modularity in metabolic networks*, Science, vol.297(5586), pp.1551-1555, 2002.

[7] Y. Li, G. Zhang, Y. Feng and C. Wu, *An entropy-based social network community detecting method and its application to scientometrics*, Scientometrics, vol.102(1), pp.1003-1017, 2015.

[8] J. Chen and Y. Saad, *Dense subgraph extraction with application to community detection*, IEEE Transactions on Knowledge and Data Engineering, vol. 24(7), pp.1216-1230, 2012.

[9] I.X.Y. Leung, P. Hui, P. Lio and J. Crowcroft, *Towards real-time community detection in large networks*, Physical Review E, vol.79, Article ID 066107, 2009.

[10] A. Gazni and F. Didegah, *Investigating different types of research collaboration and citation impact: a case study of Harvard University's publications*, Scientometrics, vol.87, no.4, pp.251-265, 2011.

[11] A. Perianes-Rodriguez, C. Olmeda-Gomez and F. Moya-Anegon, *Detecting, identifying and visualizing research groups in co-authorship networks*, Scientometrics, vol.82(2), pp.307-319, 2010.

[12] J. Ramakrishnan and K. Thavamani, *Growth of literature in the field of Hepatitic-C*, Library Philosophy and Practice, May 2013, pp.1-23, 2013.

[13] N. Benchettara, R. Kanawati and C. Rouveirol, *A supervised machine learning link prediction approach for academic collaboration recommendation*, Proceedings of the Fourth ACM Conference on Recommender Systems, pp.253-256, 2010

[14] V. Khaparde and P. Shubhangi, *Authorship pattern and degree of collaboration in information technology*, Journal of Computer Science and Information Technology, June 2013, pp.46-54, 2013.

[15] K. Thavamani, *Authorship and Collaborative Patterns in the Chinese Librarianship: an International Electronic Journal, 1996-2013*, Chinese Librarianship: and International Electronic Journal, June 2014(37), pp.1-14, 2014.

[16] D. Kutra, A. Parikh, A. Ramdas and J. Xiang, *Algorithms for graph similarity and subgraph matching*, Proc. Ecol. Inference Conf. 2011.