Analyzing, Comparing and Recommending Conferences

Grettel Monteagudo García

Dissertação (Mestrado em Informática). Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, 2016.

Pontifícia Universidade Católica
do Rio de Janeiro

Grettel Monteagudo García

Analyzing, Comparing and Recommending Conferences

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio, as partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Marco Antonio Casanova

Rio de Janeiro
March 2016

Pontifícia Universidade Católica
do Rio de Janeiro

Grettel Monteagudo García

Analyzing, Comparing and Recommending Conferences

Dissertation presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática do Centro Técnico Científico da PUC-Rio, as partial fulfillment of the requirements for the degree of Mestre.

Prof. Marco Antonio Casanova
Advisor
Departamento de Informática – PUC-Rio

Prof. Giseli Rabello Lopes
Departamento de Ciência da Computação – UFRJ

Prof. Bernardo Pereira Nunes
Coordenação Central de Educação a Distância – PUC-Rio

Prof. Marcio da Silveira Carvalho
Coordenador Setorial de Pós-Graduação

Rio de Janeiro, March. 17th, 2016

Grettel Monteagudo García

Graduated in Computer Science from University of Havana (UH), Havana - Cuba in 2012. She joined the Master in Informatics at Pontifical Catholic University of Rio de Janeiro (PUC-Rio) in 2014.

Bibliographic data

Monteagudo García, Grettel

Analyzing, Comparing and Recommending Conferences / Grettel Monteagudo García ; advisor: Marco Antonio Casanova. – 2016.

65 f. : il. (color.) ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2016.

Inclui bibliografia

1. Informática – Teses. 2. Análise de Conferências. 3. Análise Estatística. 4. Análise de Redes Sociais. 5. Comparação de Conferências. 6. Recomendação de Conferências. 7. Dados Interligados. I. Casanova, Marco Antonio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

I would like to say a special thank you to my parents, Sonia and Camilo, for their support and encouragement during all these years of study. To my little sisters, all I do is with the hope to be an example to them. To all my family for their joy.

To Marco Antonio Casanova, the best advisor I could ever ask for. I admire him for their professionalism and dedication to his students.

To PUC-Rio, CNPq and FAPERJ for funding my research.

To all my classmates, professors and staff from the Department of Informatics. Thanks to all for your help and for always being so accommodating.

Abstract

Monteagudo García, Grettel; Casanova, Marco Antonio (Advisor). **Analyzing, Comparing and Recommending Conferences.** Rio de Janeiro, 2016. 65p. MSc. Dissertation – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This dissertation discusses techniques to automatically analyze, compare and recommend conferences, using bibliographic data, outlines an implementation of the proposed techniques and describes experiments with data extracted from a triplified version of the DBLP repository. Conference analysis applies statistical and social network analysis measures to the co-authorship network. The techniques for comparing conferences explore familiar similarity measures, such as the Jaccard similarity coefficient, the Pearson correlation similarity and the cosine similarity, and a new measure, the co-authorship network communities similarity index. These similarity measures are used to create a conference recommendation system based on the Collaborative Filtering strategy. Finally, the work introduces two techniques for recommending conferences to a given prospective author based on the strategy of finding the most related authors in the co-authorship network. The first alternative uses the Katz index, which can be quite costly for large graphs, while the second one adopts an approximation of the Katz index, which proved to be much faster to compute. The experiments suggest that the best performing techniques are: the technique for comparing conferences that uses the new similarity measure based on co-authorship communities; and the conference recommendation technique that explores the most related authors in the co-authorship network.

Keywords

Conference analysis; Statistical Analysis; Social Network Analysis; Conference Comparison; Conference Recommendation; Linked Data.

Resumo

Monteagudo García, Grettel; Casanova, Marco Antonio. **Análise, Comparação e Recomendação de Conferências.** Rio de Janeiro, 2016. 65p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação discute técnicas para automaticamente analisar, comparar e recomendar conferências, usando dados bibliográficos. Apresenta uma implementação das técnicas propostas e descreve experimentos com os dados extraídos de uma versão triplificada do repositório DBLP. A análise de conferências baseia-se em medidas estatísticas e medidas para a análises de redes sociais aplicadas à rede de coautoria das conferências. As técnicas para comparar conferências exploram um conjunto de medidas de similaridades como, por exemplo, o coeficiente de similaridade de Jaccard, a similaridade por correlação de Pearson e o Cosseno, além de uma nova medida de similaridade baseada em comunidades de coautores. As medidas para calcular similaridade entre conferências são usadas em um sistema de recomendação baseado na estratégia de filtragem colaborativa. Finalmente, a dissertação introduz duas técnicas para recomendar conferências a um determinado autor, usando uma medida de relação entre autores. A primeira alternativa usa o índice de Katz, que pode ser computacionalmente lento para grandes grafos, enquanto a segunda adota uma aproximação do índice de Katz, que mostrou ser computacionalmente mais eficiente. Os experimentos sugerem que as melhores técnicas são: a técnica de comparação de conferências que utiliza a nova medida de similaridade baseada em comunidades de coautores; e a técnica para recomendação de conferências que explora os autores mais relacionados na rede de coautores.

Palavras-chave

Análise de Conferências; Análise Estatística; Análise de Redes Sociais; Comparação de Conferências; Recomendação de Conferências; Dados Interligados.

Table of Contents

List of Figures

List of Tables

1
Introduction

1.1 Motivation

Periodically, the academic community organizes a large number of conferences, in the most diverse areas, generating a rich set of bibliographic data. Researchers have explored such data to discover topics of interest, find related research groups and estimate the impact of authors and publications (BLANCHARD, 2012; CHEN; SONG; ZHU, 2007; CHEN; ZHANG; VOGELEY, 2009; GASPARINI; KIMURA; PIMENTA, 2013; HENRY et al., 2007).

In particular, Social Network Analysis (SNA) techniques have been applied to investigate co-authorship networks. SNA is a sociological approach for analyzing patterns of relationships and interactions between social actors in order to discover the underlying social structure, such as central nodes that act as hubs, leaders or gatekeepers, highly connected groups and patterns of interactions between groups (WASSERMAN; FAUST, 1994). For example, Zervas et al. (2014) applied SNA metrics to analyze the co-authorship network of the Educational Technology & Society (ETS) Journal. Procópio et al. (2011) conducted a similar analysis for the databases area. Cheong and Corbitt (2009a, 2009b) focused on the Information Systems area, analyzing the Pacific Asia Conference on Information Systems and the Australasian Conference on Information Systems. Recently, Lopes et al. (2015) carried out an extensive analysis of the last ten last editions of the WEBIST conferences, involving authors, publications, conference impact, topics coverage, community analysis, as well as other aspects.

However, replicating such analysis to other publication vehicles or areas can be an arduous task. To address this problem, we propose in this dissertation techniques to automatically analyze conferences, using bibliographic data, implement the proposed techniques and describe experiments with data extracted from a triplified version of the DBLP repository.

In another direction, selecting a good conference or journal in which to publish a new article is very important to researchers and scholars. The choice of the publication venue is usually based on the author's knowledge of the publication venues in his research domain or on matching the conference topics with his paper subject. An author may not be aware of new or other more appropriate publication venues to which his paper could be submitted. To help address this second problem, we propose in this dissertation measures to compare conference and investigate methods to recommend conferences to a given prospective author.

1.2 Contributions

The main contribution of this work is to propose, implement and evaluate techniques to automatically analyze, compare and recommend conferences.

Conference analysis is based on statistical and social network analysis measures, applied to the co-authorship network. The techniques for comparing conferences explore familiar similarity measures, such as the Jaccard similarity coefficient, the Pearson correlation similarity and the cosine similarity. We also propose a new similarity measure, the co-authorship network communities similarity index. These similarity measures are used to create a recommendation system based on the collaborative filtering strategy.

To recommend conferences, we also propose a new algorithm. For a given author, the general strategy of the algorithm is to find the strongest related authors in the co-authorship network and recommend (to the given author) the conferences that they usually publish in. The first version of the algorithm uses the Katz index (KATZ, 1953), an index for measuring relatedness of actors in Social Networks that was later adapted by Nunes et al. (2013) for measuring relatedness in large graphs. Since the Katz index may be very slow to compute for large graphs, we then introduce a second version of the algorithm that adopts an alternative index, which is much more efficient to implement and proved to return similar results.

The dissertation includes the description of a Web-based application that enables users to interactively analyze and compare a set of conferences. The

application uses a triplified version of DBLP[1], which follows the Linked Data (LD) principles (BERNERS-LEE, 2006), along the lines of (BATISTA; LÓSCIO, 2013; LOPES et al., 2015). The original DBLP repository[2] stores Computer Science bibliographic data for more than 4,500 conferences and 1,500 journals (as of early 2016).

The experiments indicate that the best performances are obtained by: (1) the technique for comparing conferences using the new co-authorship network community similarity index; and (2) the conference recommendation technique that explores the co-authorship network and adopts an approximation of the Katz index. These two techniques are therefore the major contributions of this paper.

1.3 Dissertation Structure

This dissertation is structured as follows. Chapter 2 presents the basic concepts and summarizes related work. Chapter **Error! Reference source not found.** covers the automatic analysis of a single conference. Chapter 0 introduces metrics and algorithms to compare and recommend conferences. Chapter 5 describes the implementation of the system to analyze conference. Chapter 6 details an evaluation of the proposed algorithms. Finally, Chapter 7 contains the conclusions and proposes future work.

[1] http://dblp.l3s.de/dblp++.php
[2] http://dblp.uni-trier.de/

2
Background

This chapter provides an overview of the main concepts related to this dissertation. Section 2.1 covers social network analysis techniques. Section 2.2 addresses recommendation systems based in collaborative filtering. Section **Error! Reference source not found.** presents classical similarity, rating and statistical measures used for analyzing, comparing and recommending conferences. Finally, Section 2.4 presents additional related work.

2.1 Social Network Analysis

2.1.1. Definition of Social Network

Consider a social network represented as a graph $G = (N, E)$, where N is the set of nodes, where $n_i \in N$ represents an actor of the network, and E is the set of edges, where $e_i \in E$ represents a relational tie between a pair of actors.

2.1.2. Definition of a Social Network for Co-authorship Relation Between Authors

We may define several social networks with the bibliographic data of a conference. In the context of this dissertation, to analyze and recommend conferences, we use a specific type of the social network over conference data: the *co-authorship network.*

A *co-authorship network* consists of a collection of researchers each of whom is connected to other researchers if they have co-authored one or more papers. Formally, a *co-authorship network* is defined as an undirected and unweighted graph $G = (N, E)$, where N is the set of authors such that $n_i \in N$ represents an author of the network and E is such that $e_{i,j} \in E$ represents that the author *i* and the author *j* have co-authored one or more papers.

2.1.3. Measures to Analyze a Social Network

Informally, the following measures are adopted to analyze a social network.

The *density* of G is defined as the number of the edges of G divided by the maximum number of edges G can have. The maximal density is 1 (for complete graphs) and the minimal density is 0.

The density for undirected graphs is defined as :

$$D = \frac{2|E|}{|N|(|N|-1)} \quad (1)$$

and, the density for directed graphs is defined as :

$$D = \frac{|E|}{|N|(|N|-1)} \quad (2)$$

The *modularity* of G (BLONDEL et al., 2008; NEWMAN; GIRVAN, 2004) is used to measure the degree of clustering of G. Intuitively, if G consists of distinct closed subgroups or *communities*, then G has a larger modularity. Modularity is measured by comparing the number of edges inside communities with the number of edges in the whole graph. The number of communities is detected based on the modularity. Modularity can be defined as:

$$Q = \sum_i (e_{ii} - a_i^2) \quad (3)$$

where e_{ij} is the number of edges connecting nodes from community i to nodes from community j and $a_i = \sum_j e_{ij}$ is the number of edges with at least one node from community i.

The *clustering coefficient* of a node n in G is defined as the number of existing edges between the direct neighbors of n divided by the total number of possible edges directly connecting all neighbors of n.

The *average clustering coefficient* is a measure of the degree to which nodes in a graph tend to cluster together (connectivity of neighbors). It is defined as the average of the clustering coefficients of all the nodes in the graph:

$$\bar{C} = \frac{\sum_{i=1}^{|N|} C_i}{|N|} \quad (4)$$

where N is the set of nodes and C_i is the clustering coefficient of a node n_i.

The *diameter* of G is associated with graph distance. It is defined as the maximum value among all shortest paths between two nodes of G (i.e., the longest distance between any pair of nodes belonging to G).

The *giant component* of G is the largest connected component of G.

The *giant coefficient* of G is defined as the number of nodes of the giant component of G divided by the total number of nodes of G.

$$GC = \frac{|N'|}{|N|} \quad (5)$$

where $|N'|$ is the number of nodes in the giant component and $|N|$ is the number of nodes in the entire graph.

Finally, the *semantic connectivity score* (NUNES et al., 2013) is based on the Katz index and measures the relatedness of actors in social networks. The semantic connectivity score $SCS_e(n_i, n_j)$ between a pair of nodes (n_i, n_j) of G is defined as:

$$SCS_e(n_i, n_j) = \sum_{l=1}^{T} \beta^l \cdot \left| paths_{<n_i,n_j>}^{<l>} \right| \quad (6)$$

where $|paths_{<n_i,n_j>}^{<l>}|$ is the number of paths of length l between n_i and n_j, T is the maximum length of paths considered and $0 < \beta \leq 1$ is a positive damping factor. The damping actor β^l is responsible for exponentially penalizing longer paths. The smaller this factor is the smaller the contribution to the final score of longer paths will be.

2.2 Collaborative Filtering

In a recommendation system, there are two classes of entities, which we shall refer to as users and items. Users have preferences for certain items, and these preferences must be extracted from the data (LESKOVEC; RAJARAMAN; ULLMAN, 2014). In the context of this dissertation we need to compute the preference of the authors (users) by the conferences (items). The data itself is represented as a utility matrix giving, for each author-conference pair, a value that represents what is known about the degree of preference of that author for that

conference to publish his work. An unknown rating implies that we have no explicit information about the author's preference for the conference.

The goal of a recommendation system is to predict the blanks in the utility matrix. The first step of a collaborative filtering algorithm is to obtain the authors history profile, which can be represented as the utility matrix. The second step is to calculate the similarity between authors or conferences and to find their nearest neighbors (most similar). The last step is to calculate the conference rating.

To measure similarity in the problem of recommending items to users from rows or columns in the utility matrix, one can use the Jaccard distance, the cosine distance or the Pearson correlation coefficient between vectors, presented in their classical definition in Section 2.3.1. The *k* nearest neighbors are the *k* most similar conferences. The rating is computed by a weighted average of the ratings by the neighbors.

2.3 Classical Similarity, Rating and Statistical Measures

2.3.1. Similarity Measures

The *Jaccard similarity coefficient* (JACCARD, 1901) is used to compare the similarity and diversity of sample sets. It is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (7)$$

The *Pearson's correlation coefficient* (LEE RODGERS; NICEWANDER, 1988), often denoted by the letter *r*, measures the strength and direction of the linear correlation between two variables *X* and *Y*. It is defined as the covariance of the variables divided by the product of their standard deviations to measure their dependence:

$$r = \frac{\sum_{i=1}^{N}(x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{X})^2 \sum_{i=1}^{N}(y_i - \bar{Y})^2}} \quad (8)$$

The *r* value lies between +1 and -1 and indicates the degree of linear dependence between *X* and *Y*; *r* = 1 indicates a total positive correlation between the two variables and *r* = -1 indicates a total negative (inverse) correlation.

The *cosine similarity* is a measure of similarity between two vectors of an inner product space, defined as the cosine of the angle between the vectors. Cosine similarity is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude:

$$sim(x,y) = cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{\|\vec{x}\| \times \|\vec{y}\|} \quad (9)$$

The cosine similarity value lies between 0 and 1 in the context of recommendation systems, where the vectors have values greater than 0 and the angle between them is always 0° and 90°.

2.3.2. Rating Measures

In collaborative filtering, the value of ratings user *x* gives to item *s* is defined as an aggregation of the ratings of the neighbors. There exists several aggregation functions, but the most commonly used is the following:

$$CF = \frac{\sum_{y\in S_x}(r_{y,i})sim(x,y)}{\sum_{y\in S_x} sim(x,y)} \quad (10)$$

where S_x indicates the nearest neighbors, $r_{y,i}$ is the rating of item *i* given by user *y*. In Section 4.2.1 we contextualize this formula to rate the preferences of an author to a conference and propose an algorithm to recommend conferences based on this rating.

2.3.3. Statistical Measures

The *standard deviation* is a measure of the dispersion of a set of data from its mean. It is defined as the square root of the variance:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2} \quad (11)$$

where μ is the mean of the variable x.

The *Lorenz curve* (GINI, 1912) represents the cumulative distribution of a probability density function. Such a function is built as a ranking of the members

of the population disposed of in ascending order of the amount being studied. The percentage of individuals is plotted on the x-axis and the percentage of the variable values on the y-axis. The distribution is perfectly equalitarian when every individual has the same variable value; a 45-degree line represents the perfect equality. On the other hand, the perfectly unequal distribution is the one in which only one individual has all the variable value.

The *Gini coefficient* (GINI, 1912) is a measure of the statistical dispersion indicating the inequality among values of a frequency distribution. It is graphically represented as the area between the perfect equality line and the observed Lorenz curve.

The *Robin Hood index* (HOOVER, 1941), also called *Hoover index*, is used to measure the fraction of the total variable values that must be redistributed over the population to become a uniform distribution. It is graphically represented as the longest vertical distance between the Lorenz curve and the perfect equality line.

2.4 Related Work

2.4.1. Conference Analysis

Henry et al. (2007) analyzed a group of the four major conferences in the field of Human-Computer Interaction (HCI). The authors discovered many global and local patterns using only article metadata, such as authors, keywords and year. The paper presented in (BLANCHARD, 2012) is the result of a ten-year analysis of paper production in Intelligent Tutoring Systems (ITS) and Artificial Intelligence in Education (AIED) conferences and shows that Western, Educated, Industrialized, Rich, and Democratic bias observed in psychology may be influencing AIED research. Chen, Zhang, and Vogeley (2009) proposed an extension of the contemporary co-citation network analysis, to identify co-citation clusters of cited references. Intuitively, the authors synthesize thematic contexts in which these clusters are cited and trace how the research focus evolves over time. Gasparini, Kimura and Pimenta (2013) presented a visual exploration of the field of Human Computer Interaction in Brazil from a fifteen-year analysis of paper production in the Brazilian Symposium on Human Factors in Computing Systems

(IHC). Such analysis helped getting insights from the data and identifying topics evolution, central authors and institutions, and important trends. Chen, Song and Zhu (2007) opened a wide range of opportunities for research agendas and trends in ER conferences.

Zervas et al. (2014) applied social network analysis (SNA) metrics to analyzing the co-authorship network of the Educational Technology & Society (ETS) Journal. Procópio, Laender and Moro (2011) did a similar analysis for the Databases field. Regarding Information Systems fields, Cheong and Corbitt (2009a, 2009b) analyzed the Pacific Asia Conference on Information Systems and the Australasian Conference on Information Systems.

Recently, Lopes et al. (2015) carried out an extensive analysis of the WEBIST conferences, involving authors, publications, conference impact, topics coverage, community analysis and other aspects.

Linked Data principles to publish conference data was used by (BATISTA; LÓSCIO, 2013; LOPES et al., 2015).

As mentioned in Section 1, to replicate the previous analyses would be a laborious task. Hence, this dissertation proposes a system that combines and proposes improvements to the previous work on conference analysis thereby facilitates automatic conference analysis. The work reported here naturally extends to any type of publication venue for which data about the papers or articles, authors, etc. are available.

2.4.2. Conference Recommendation Methods

Luong et al. (2012) proposed and compared three recommendation methods for conferences and proved that Method 3 has the best accuracy. To define the methods, they recursively collected the co-authors of the co-authors, until a network of 3 levels deep was created, in a set of the more important co-authors.

Method 1 - Most frequent conference: This method weights the candidate conference values simply by computing the total number of papers published in the conference by the authors in a set of the more important co-authors.

$$freq\_CONF_i = \sum_{j=1}^{N} num\_paper_{i,j} \quad (12)$$

Method 2 - Most frequent conference normalized by author: This method computes the weight of a conference *i* as the sum of each author's probability of publishing in conference *i*.

$$nfreq\_CONF_i = \sum_{j=1}^{N} \frac{num\_paper_{i,j}}{total\_paper_j} \quad (13)$$

Method 3 - Method 2 incorporating network topology: Both the previous methods treat all authors equally, ignoring the strength of the connections between the authors in the network. Method 3 weights the contributions of each co-author by the number of papers they have co-authored with the main author.

$$coauthor\_CONF_i = \sum_{m=1}^{N} coauthors\_w_{i,m} \quad (14)$$

where *N* is main author(s) of the test paper, $coauthors\_w_{i,m}$ is the co-authors' conference weight between the main author *m* and her co-authors in the network.

$$coauthors\_w_{i,m} = \sum_{k=1}^{CoA} \left(nfreq\_CONF_{i,m} + nfreq\_CONF_{i,k}\right) * w\_CoA_{k,m} \quad (15)$$

where CoA is a co-author(s) of the main author *m* who have published respectively at conference *i*, and $w\_CoA_{k,m}$ is the number of times a main author *m* has co-authorships with another member *k* in the network.

In this dissertation, we propose two conference recommendation strategies based on the social network analysis of the co-authorship network, but adopting a measure of the strength of the connections between the authors in the network which is computed differently from **Method 3**. We first propose to estimate the relatedness of actors in a social network by using a semantic connectivity score (NUNES et al., 2013), based on the Katz index (KATZ, 1953). This score, introduced in Section 2.1, takes into account the number of paths between two node of the networks and the length of this paths. Then, we propose a second score that approximates the Katz index and that uses the shortest path between two nodes. In addition to these two strategies, we also propose to construct a utility matrix and to implement a recommendation system based on collaborative filtering using this matrix.

3
Conference Analysis

This chapter presents the main ideas behind the construction of an automatic conference analysis system. Section 3.1 describes the semantic database schema adopted. A key point of this section is to discuss how a triplified version of the data should be created. Section 3.2 shows how to query the dataset to get the necessary data for the analysis itself.

3.1 Semantic Database Schema

3.1.1. Description of the Schema

The automatic conference analysis system adopts a database designed according to the Linked Data principles. The first and the only non-automatic step of the analysis system is the creation of an *RDF dataset* that contains the conference data. This step is only necessary if the conference data is no readily available in RDF, such as the triplified version of DBLP. The DBLP++ dataset is an enhancement of DBLP, with additional keywords and abstracts as available on public Web pages; this dataset is constantly updated and available at http://dblp.l3s.de/dblp++.php. It houses data about all publications in the Computer Science area, but for this dissertation we only use a segment of the dataset which includes information about bibliographic data of academic conference. We have the data from more than 4,000 conferences.

An RDF dataset *T* is a set of RDF triples. A *triple* (*s, p, v*) in *T* defines the value *v* of a property *p* of a resource *s* (HARRIS; SEABORNE, 2012).

The RDF Schema vocabulary provides a data-modelling vocabulary for RDF data. RDF Schema is an extension of the basic RDF vocabulary (BRICKLEY; GUHA, 2014).

For the construction of the RDF dataset with conference data, we should use the RDF schema described by the Figure 1. An instance of a conference should be of type *swrc:Conference*, and the label is the name of the conference. An instance of a conference edition should be of type *swrc:Proceedings*, where *swrc:series* property relates the edition with the conference. For an edition, we also specified the name (label) and the year. Instances of type *foaf:Agent* are authors; the label of the instance is the name of the author. Finally, instances of *swrc:InProceedings* represent conference publications; the authors of a publication are specified with the *dc:creator* property. To establish the conference of the paper and the edition of the conference, we use *swrc:series* and *dcterms:part* properties, respectively. The property *dc:subject* should have as value the keywords of the publication separated by semicolons and rdf:label is a title of the publication.

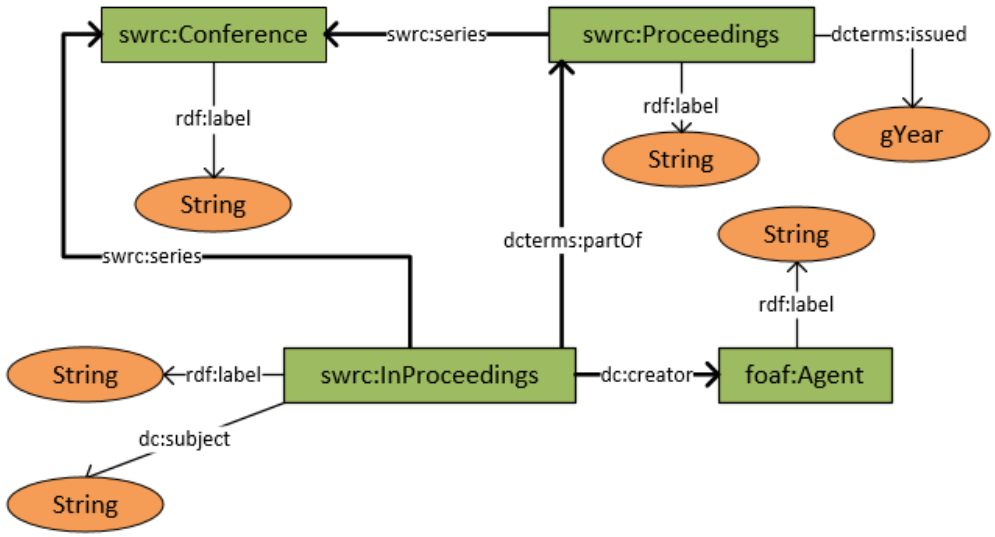

Figure 1 RDF schema for Conference Analysis

3.1.2. Example

Once the schema is defined, one must populate it, as illustrated in what follows.

Suppose that the user wants to create the 3D-GIS conference. He then defines the label of the conference and indicates that the new object is of type conference:

```
<url/resource/conferences/3dgis> rdfs:label
     "3D-GIS"^^xsd:string.
<ulr/resource/conferences/3dgis> rdf:type swrc:Conference.
```

Now, suppose that he wants to create the 2006 edition of the 3D-GIS conference. He proceeds as follows:

```
<url/resource/publications/conf/3dgis/2006> dcterms:issued
     "2006"^^xsd:gYear.
<url/resource/publications/conf/3dgis/2006> swrc:series
     <url/resource/conferences/3dgis>.
<url/resource/publications/conf/3dgis/2006> rdfs:label
     "Innovations in 3D Geo Information Systems, First
International Workshop on 3D Geoinformation, 7-8 August, 2006,
Kuala Lumpur, Malaysia"^^xsd:string.
<url/resource/publications/conf/3dgis/2006> dc:publisher
     "Springer"^^xsd:string.
<url/resource/publications/conf/3dgis/2006> rdf:type
     swrc:Proceedings.
```

To create instances to represent the authors “Mohammed Yaziz Ahmad” and “Mokhtar Azizi Mohd Din”, he add the label of the conference and indicates that the new objects are of the type Agent:

```
<url/resource/authors/Mohammed_Yaziz_Ahmad> rdfs:label
     "Mohammed Yaziz Ahmad"^^xsd:string
<url/resource/authors/Mohammed_Yaziz_Ahmad> rdf:type
     foaf:Agent
```

```
<url/resource/authors/Mohammed_Yaziz_Ahmad> rdfs:label
     "Mokhtar Azizi Mohd Din "^^xsd:string
<url/resource/authors/Mohammed_Yaziz_Ahmad> rdf:type
     foaf:Agent
```

To create the paper “Integration of GIS and Digital Photogrammetry in Building Space Analysis” in the conference 3D-GIS with authors Mohammed Yaziz Ahmad and Mokhtar Azizi Mohd Din, he writes:

```
<url/resource/publications/conf/3dgis/DinA06> rdfs:label
     "Integration of GIS and Digital Photogrammetry in Building
Space Analysis."^^xsd:string
<url/resource/publications/conf/3dgis/DinA06> swrc:series
     <url/resource/conferences/3dgis>
<url/resource/publications/conf/3dgis/DinA06> dc:creator
     <url/resource/authors/Mokhtar_Azizi_Mohd_Din>
<url/resource/publications/conf/3dgis/DinA06> dc:creator
     <url/resource/authors/Mohammed_Yaziz_Ahmad>
<url/resource/publications/conf/3dgis/DinA06> dcterms:partOf
     <url/resource/publications/conf/3dgis/2006>
<url/resource/publications/conf/3dgis/DinA06> dc:subject
     "Digital photogrammetry; GIS; spatial data; building usage;
space utilization"^^xsd:string
<url/resource/publications/conf/3dgis/DinA06> rdf:type
     swrc:InProceedings
```

3.2 Conference Analysis

3.2.1. General Analysis

The general analysis of a conference consists of a sequence of SPARQL queries. SPARQL Query 1 returns the number of authors per edition of the conference. It first finds the editions of the conference, then the papers published in this edition and finally the authors of the papers. The *SELECT* clause counts the different authors per edition.

```
SELECT  ?year count(distinct ?author) as ?Number_of_Authors
WHERE {
?edition  swrc:series <%conf_param%>;
          rdf:type swrc:Proceedings;
          dcterms:issued ?year.
?paper    dcterms:partOf ?edition;
          dc:creator ?author.
}  ORDER BY ?year
```

SPARQL Query 1 - Number of authors per edition

SPARQL Query 2 computes the number of papers per edition of a conference.

```
SELECT   ?year count(?paper) as ?Number_of_Papers
WHERE {
?paper dcterms:partOf ?edition.
?edition  swrc:series <%conf_param%>;
          rdf:type swrc:Proceedings;
          dcterms:issued ?year.
}
ORDER BY ?year
```

SPARQL Query 2 - Number of papers per edition

SPARQL Query 3 computes the average of the number of authors per paper per edition of the conference. It computes the number of authors per paper in an edition (internal select) and then the average of these values.

```
SELECT  ?year avg(?nauthor) as ?avg_Author_per_Papers
WHERE {
?edition swrc:series <%conf_param%>;
     rdf:type swrc:Proceedings;
     dcterms:issued ?year.
     {
     SELECT ?paper count(?author) as ? nauthor
     WHERE {?paper dcterms:partOf ?edition;
     dc:creator ?author.}
     }
```

```
}  ORDER BY ?year
```

SPARQL Query 3 - Average of the author per papers per edition

SPARQL Query 3 is used to compute the standard deviation, the Lorenz curve, Gini coefficient and Robin Hood of the average of authors per edition of the conference.

SPARQL Query 4 computes the maximum and the minimum number of authors per paper in a conference.

```
SELECT   ?paper max(?author) as ?max_Authors min(?author) as
?min_Authors
WHERE { ?paper a swrc:InProceedings;
     swrc:series <%conf_param%>;
     dc:creator ?author.}
```

SPARQL Query 4 - Maximum and the minimum number of authors in a paper

Other relevant indices for conference analysis are the N-top authors for the conference (SPARQL Query 6) and for each edition (SPARQL Query 5). The N-top authors are the N authors with more publications.

```
SELECT   ?author count(?paper) as ?Number_Papers
WHERE {  ?paper dcterms:partOf <%conf_param%>;
           dc:creator ?a.
      ?a foaf:name ?author.}
ORDER BY DESC(?cant)
LIMIT %N%
```

SPARQL Query 5 - The N-top author for each edition

```
SELECT  ?author count(?paper) as ?cant
WHERE {  ?paper a swrc:InProceedings;
     swrc:series <%conf_param%>;
           dc:creator ?a.
     ?a foaf:name ?author.}
ORDER BY DESC(?cant)
LIMIT %N%
```

SPARQL Query 6 - The N-top author in the conference

3.2.2. SNA over the co-authorship network

For the social network analysis of a conference, the first step is to create the co-authorship network for the conference and its editions, which is carried out by SPARQL Query 7 and SPARQL Query 8.

```
SELECT   ?paper ?author
WHERE {  ?paper a swrc:InProceedings;
     swrc:series <%conf_param%>;
     dc:creator ?author.}
```

SPARQL Query 7 - Authors by paper in a conference

```
SELECT   ?paper ?author
WHERE {  ?paper dcterms:partOf <%edition_param%>;
     dc:creator ?author.}
```

SPARQL Query 8 - Authors by paper in an edition

With these results, we can build the co-authorship network. First, we create a node for each different author. We also need to group, for each paper, its authors, because the result is a list of pairs. Then, for each pair of authors that co-authored a paper, we create an edge between them. Algorithm 1 captures this method, in pseudo-code.

```
CreateCoAuthor (SPARQL_Result result)
    GroupList author_per_paper;
    Set   authors;
    Foreach <paper,author> in SPARQL_Result
           authors.put(author);
           if (!author_per_paper.exist_group(paper))
                 author_per_paper.add_group(paper)
           author_per_paper(paper).put(author)
    Co-authorship_Network cn.
    Foreach author in authors
          Cn.addNode(author)
     Foreach group in author_per_paper
           For i=0; i< count(group); i++
                 For j=i+1; j< count(group); j++
                       If(!Cn.existsEdge(group[i], group[j]))
                             Cn.addEdge(group[i], group[j])
```

Algorithm 1 Create the co-authorship network

With the co-authorship network constructed, we may conduct an SNA analysis.

It is very simple to calculate the number of nodes, edges, average degree and density and finding the number of connected components is a familiar graph problem (Algorithm 2). The classical method executes a DFS traversal from a given node *n* and marks all nodes visited as belonging to the same connected component (as *n*). If there exists some unmarked node *u*, the method starts a new DFS traversal from *u*, until no node remains unmarked. With the connected

components defined, it is trivial to find the giant component and calculate the giant coefficient.

```
Connected_Components (Graph G)
     Foreach node in G
           If !(node is mark)
                 Set nodes
                 DFS(node, nodes)
                 next_component++
                 Mark(c, next_component)
```

Algorithm 2 Find Connected Components

To calculate the diameter of the giant component, it requires computing the shortest paths between any pair of nodes in the giant component. The method consists in executing a BFS traversal from each node. The result of the BFS traversal in an undirected and unweighted graph is the shortest path from the source node to the other nodes. The diameter is the maximal value of all result of the BFS traversal. Algorithm 3 shows this method in pseudo-code.

```
Diameter (Graph G)
     Int diameter = -1
     Foreach node in G
           IntList ditances
           BFS(node, ditances)
           Foreach distance in ditances
                 If diameter < distance
                       diameter = distance
     return diameter
```

Algorithm 3 Diameter

We also compute the average clustering coefficient of the Giant Component. This method requires computing the clustering coefficient of each node, which in turn requires finding the set of the neighbors of the node, which we call *neighbors1*. For each node x_i in *neighbors1*, we need to find their neighbors, *neighbors*x_i, and find the number of nodes in the intersection of *neighbors1* and *neighbors*x_i. Finally, we sum the results of the all intersection and divided it by *2n(n-1)*. Algorithm 4 describes the details of the above method.

```
Clustering Coefficient (Node i)
     Set neighbors1 = adjacents(i)
     Int interceptions = 0
     Foreach node in neighbors1
           Set neighborsx_i= adjacents(node)
           interceptions += count(neighbors1 ∩ neighborsx_i)
     return interceptions/(2*count(neighbors1)*
(count(neighbors1)-1))
```

Algorithm 4 Clustering Coefficient

The algorithm to compute the communities presented in Blondel et al. (2008) is divided into two phases that are repeated iteratively. It starts with a weighted network of *N* nodes. First, the algorithm assigns a different community to each node of the network. Then, for each node *I*, the algorithm considers the neighbors *j* of *i* and evaluates the gain in modularity that would take place by removing *i* from its community and by placing *i* in the community of *j*. The algorithm stops when no individual move can improve the modularity. The second phase of the algorithm consists in building a new network whose nodes are now the communities found during the first phase and the weights of the links between the new nodes are given by the sum of the weight of the links between the nodes in the corresponding two communities. Once this second phase is completed, it is then possible to reapply the first phase of the algorithm to the resulting weighted network.

4
Comparing and Recommending Conferences

This chapter describes our approach to develop conference recommendation systems. Section **Error! Reference source not found.** presents the construction of a recommendation system, based on collaborative filtering, that adapts the utility matrix and similarity measures to the author and conference scenario. Section 4.2 explains a new algorithm for conference recommendation based on the relatedness of authors on the co-authorship network.

4.1 Comparing Conferences

In what follows, we use the following notation:

- C is a set of conferences
- N is a set of authors
- P is a set of papers
- x and y are two conferences
- A_x and A_y are the set of authors that published in x and y
- $I_{x,y}$ is the set of authors that published in both conferences x and y
- $p: N \rightarrow P$ is a function that assigns to each author i in N a set of papers $p(i) \subseteq P$
- $pc: N \times C \rightarrow P$ is a function that assigns to each author i in N and each conference x in C the set $pc(i, x) \subseteq P$ of publications of author i in conference x
- G_x is the co-authorship network for conference x defined as the undirected and unweighted graph $G_x = (N_x, E_x)$ where $i \in N_x$ indicates that author i published in conference x and $e_{i,j} \in E_x$ represents that author i and author j co-authored one or more papers published in conference x

4.1.1. Utility Matrix

Authors have preferences for certain conferences, where publish his work constantly, and these preferences must be extracted out of the data. The data itself is represented as a utility matrix that gives, for each conference-author pair, a value that represents what is known about the degree of preference of that author for that conference to publish his work.

In this scenario, the utility matrix expresses the preferences of an author for a conference to publish his research. More formally, the utility matrix $[r_{x,i}]$ is such that the lines represent conferences and the columns represent authors and is defined as:

$$r_{x,i} = \frac{|pc(i,x)|}{|p(i)|} \quad (16)$$

4.1.2. Similarity measures

Recommender systems based on collaborative filtering depend on computing the similarity between two users or items. Therefore, to create a collaborative conference recommender system, we propose to adapt or create similarity measures between conferences or authors. We adapt the Jaccard, Pearson and Cosine similarity measures to conferences, using the previous definitions and the utility matrix. We also propose a new similarity measure called Communities Similarity.

Jaccard Similarity between Conferences

The *Jaccard similarity coefficient* for conferences x and y is directly defined as:

$$jaccard\_sim(x,y) = \left(A_x \cap A_y\right)/\left(A_x \cup A_y\right) \quad (17)$$

Note that $A_x = \{i \in N \,/\, |pc(i,x)| > 0\}$ and, likewise, $A_y = \{i \in N \,/\, |pc(i,y)| > 0\}$.

Pearson's Correlation Coefficient Similarity between Conferences

Based on the utility matrix $[r_{x,i}]$, we define the Pearson's correlation coefficient similarity between conferences x and y as follows:

$$pearson\_sim(x,y) = \frac{\sum_{i \in l_{xy}} (r_{x,i} - \overline{r_x})(r_{y,i} - \overline{r_y})}{\sqrt{\sum_{i \in l_{xy}} (r_{x,i} - \overline{r_x})^2 \sum_{i \in l_{xy}} (r_{y,i} - \overline{r_y})^2}} \quad (18)$$

Cosine Similarity between Conferences

Based on the utility matrix $[r_{x,i}]$, we define the cosine similarity between conferences x and y as follows:

$$cos\_sim(x,y) = \frac{\sum_{i \in l_{xy}} (r_{x,i} r_{y,i})}{\sqrt{\sum_{i \in l_{xy}} (r_{x,i})^2 \sum_{i \in l_{xy}} (r_{y,i})^2}} \quad (19)$$

Communities Similarity

We introduce a new similarity measure between conferences based on communities defined over the co-authorship network G_x of conference x.

We define an *author community* c_x of conference x as the net of nodes of a connected component of G_x.

Let c_x and c_y be author communities in the co-authorship networks of conferences x and y, respectively. We say that c_x and c_y are *equivalent* w.r.t. a similarity measure *sim* and a threshold level α iff $sim(c_x, c_y) \geq \alpha$. For example, *sim* may be defined using Jaccard similarity coefficient.

Let C_x and C_y be the sets of communities in the co-authorship networks of conferences x and y, respectively. Let $EQ(x,y)$ be the set of communities in the co-authorship network of conference x that have an equivalent community in the co-authorship network of conference y (and symmetrically $EQ(y,x)$).

The *co-authorship network communities similarity* (based on a similarity measure *sim* and a threshold level α) between conferences x and y is then defined as:

$$c\_sim(x,y) = \begin{cases} \frac{|EQ(x,y)|}{|C_x|}, & if\ |C_x| < |C_y| \\ \frac{|EQ(y,x)|}{|C_y|}, & otherwise \end{cases} \quad (20)$$

Note that $C_x > 0$ and $C_y > 0$ since G_x and G_y must have at least one node each and therefore at least one connected component each.

4.1.3. Example

In this example, we have two conferences, five authors and six publications distributed as follows:

The publications of the C1 (conference 1)

Publication	Authors
P1	*A1, A3*
P2	*A1, A4*
P3	*A2*

The publication of the C2 (conference 2)

Publication	Authors
P4	*A1, A3*
P5	*A2, A5*
P6	*A5*

Classical Similarity Measures

C1 and *C2* have three authors with publications on both conferences and the total of authors is five. Then, the Jaccard similarity is:

$$jaccard\_sim(C1, C2) = 3/5$$

We know that A1 has three publications, two in C1 and one in C2; A2 and A3 have two publications each, one for each conference; A4 has one publication in C1 and A5 has two publications in the C2. The utility matrix used to compute $pearson\_sim(C1, C2)$ and $cos\_sim(C1, C2)$ is:

	A1	A2	A3	A4	A5
C1	2/3	½	½	1	0
C2	1/3	½	½	0	1

To calculate $pearson\_sim$ we have that:

$\bar{r_x} = (\frac{2}{3} + \frac{1}{2} + \frac{1}{2} + 1)\ /4 = 8/3$

$\bar{r_y} = (\frac{1}{3} + \frac{1}{2} + \frac{1}{2} + 1)\ /4 = 7/3$

$$pearson\_sim(C1, C2)$$

$$= \frac{\left(\frac{2}{3} - \frac{8}{3}\right) * \left(\frac{1}{3} - \frac{7}{3}\right) + 2 * \left(\frac{1}{2} - \frac{8}{3}\right) * \left(\frac{1}{2} - \frac{7}{3}\right)}{\sqrt{\left(\frac{2}{3} - \frac{8}{3}\right)^2 + 2 * \left(\frac{1}{2} - \frac{8}{3}\right)^2} * \sqrt{\left(\frac{1}{3} - \frac{7}{3}\right)^2 + 2 * \left(\frac{1}{2} - \frac{7}{3}\right)^2}}$$

$$= \frac{-2 * -2 + 2 * \left(-\frac{13}{6}\right) * -\frac{11}{6}}{\sqrt{(-2)^2 + 2 * \left(-\frac{13}{6}\right)^2} * \sqrt{(-2)^2 + 2 * \left(-\frac{11}{6}\right)^2}}$$

$$= \frac{4 + 2 * \frac{143}{36}}{\sqrt{4 + 2 * \frac{169}{36}} * \sqrt{4 + 2 * \frac{121}{36}}} \approx \frac{11.94}{11{,}98} \approx 0.99$$

$$cos\_sim(C1, C2) = \frac{\left(\frac{2}{3}\right) * \left(\frac{1}{3}\right) + 2 * \left(\frac{1}{2}\right) * \left(\frac{1}{2}\right)}{\sqrt{\left(\frac{2}{3}\right)^2 + 2 * \left(\frac{1}{2}\right)^2} * \sqrt{\left(\frac{1}{3}\right)^2 + 2 * \left(\frac{1}{2}\right)^2}} \approx 0.95$$

Communities Similarity

The co-authorship network for each conference is illustrated in Figure 2. Assume a threshold level α = 0.6.

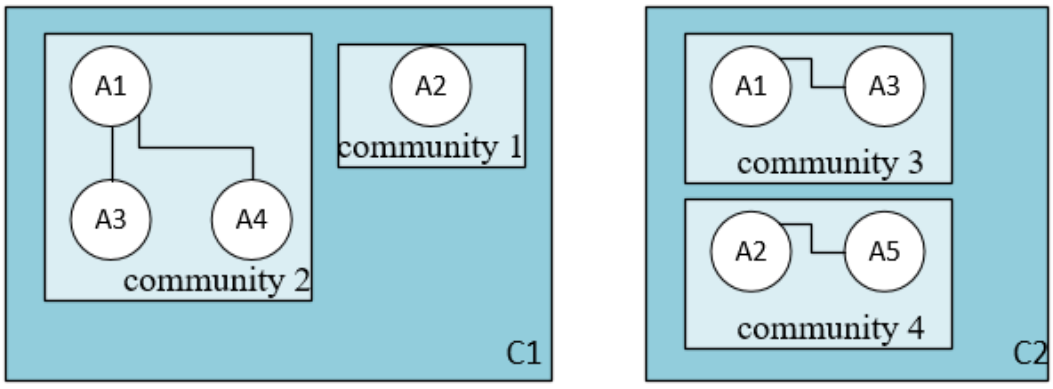

Figure 2 Co-Authorship Network for the example

The first step to compute $c\_sim(C1, C2)$ is to find the number of equivalent communities between C1, C2. Recall that two communities are equivalent if their Jaccard similarity is greater than 0.6.

The Jaccard similarity values between community 1 and communities 3 and 4 are 0 and 0.5, respectively, i.e. community 1 has no equivalent community. The Jaccard similarity values between community 2 and communities 3 and 4 are 0.67

and 0, respectively; in this case we find that communities 2 and 3 are equivalent and we then have:

$$c\_sim(C1, C2) = \frac{|EQ(C1, C2)|}{|\mathrm{C1}|} = 1/2$$

We note that the four methods result in different values, but the values of $pearson\_sim$ and cos_sim are close, as are the values of $jaccard\_sim$ and c_sim. Taking into account that $jaccard\_sim$ and c_sim use only information about what authors publish in the conference and $pearson\_sim$ and cos_sim also use information about how many publications the author has in a conference, these results are expected.

4.2 Recommending Conferences

4.2.1. Conference Recommendation Based on the Collaborative Filtering Algorithm

The first step of the collaborative filtering algorithm is to obtain the authors' history profiles, which are represented with the utility matrix. The second step is to calculate the similarity between conferences and to find their nearest neighbors (most similar) conferences. The last step is to calculate which conferences are recommended to the author using a conference rating measure.

Recall that recommendation techniques based on collaborative filtering (Leskovec et al., 2014) depend on computing the similarity. Therefore, we may immediately define a family of conference recommendation techniques using the algorithm presented in Section 2.2 based on the similarity measures introduced in previous section, that we call CF-Jaccard, CF-Pearson, CF-Cosine and CF-Communities, according to the similarity measure adopted.

To calculate the rating of a conference x for author i we use the formula:

$$CF(x, i) = \frac{\sum_{y \in S_x}(r_{y,i})sim(x, y)}{\sum_{y \in S_x} sim(x, y)} \quad (21)$$

where S_x are the most similar conferences to x.

4.2.2. Conference Recommendation Based on the Weighted Co-authorship Network

We propose a specific conference recommendation algorithm, using the notion of a weighted co-authorship network.

A *weighted* co-*authorship network* based on $p: A \rightarrow P$ is an edge-weighted undirected graph $G = (N, E, w)$, where

$i \in N$ represents an author

$e_{i,j} \in E$ indicates that i and j are co-authors, that is, $e_{i,j} \in E$ iff $p(i) \cap p(j) \neq \emptyset$

$w(e_{i,j})$ assigns a weight to the co-authorship relationship between i and j and can be defined with several metrics for example as:

$$w(e_{i,j}) = \begin{cases} 11 - \left(\frac{|p(i) \cap p(j)|}{|p(i) \cup p(j)|}\right) * 10 \text{ , if } |p(i) \cup p(j)| \neq \emptyset \\ \infty \text{ , otherwise} \end{cases} \quad (22)$$

Hence, the smaller $w(e_{i,j})$ is, the stronger the co-authorship relationship will be: if authors i and j co-authored all papers they published, then $w(e_{i,j}) = 1$; and if they have not co-authored any papers, then the $w(e_{i,j}) = \infty$, because the edge $e_{i,j}$ not exist. Note that the constants 10 and 11 in Eq. (22) assure that the weight of an edge is always greater than 1. Other metric would assign 1, if $|p(i) \cup p(j)| \neq \emptyset$, and ∞, otherwise.

The second family of recommendation techniques explores the weighted co-authorship network and adopts two scores: the *weighted semantic connectivity score – WSCS* and the *modified weighted semantic connectivity score – MWSCS*. Hence, these techniques are called *WSCS-based* and *MWSCS-based recommendation techniques*.

4.2.3. WSCS-based Conference Recommendation Technique

For the first method we propose, we also define a *weighted semantic connectivity score*, $WSCS_e$, by modifying the definition of the semantic connectivity score SCS_e to take into account the weight of the path, computed as the sum of the weights of the edges in the path.

$$WSCS_e(i,j) = \sum_{w=1}^{T} \beta^w \cdot \left|paths_{<i,j>}^{<w>}\right| \quad (23)$$

where $|paths_{<i,j>}^{<w>}|$ is the number of paths of weight equal to w between i and j and T is the maximum weight of paths and $0 < \beta \leq 1$ is a positive damping factor.

The conference recommendation technique based on $WSCS_e$ score works as follows. Given an author i, it starts by computing the $WSCS_e(i,j)$ score between i and any other author j in the weighted co-authorship network. Then, it orders authors by decreasing order of $WSCS_e$ since authors that are better related to author i will have a higher $WSCS_e(i,j)$ score. For better performance, the algorithm considers only the first n authors in the ordered list of $WSCS_e$. For each author j, the algorithm selects the conference with the highest rank (Eq. 22), denoted $MaxC_j$. The rank of conference x for author i is defined as follows:

$$rank(x,i) = \sum_{j \in A \text{ and } MaxC_j = x} WSCS_e(i,j) \quad (24)$$

Algorithm 5 presents this process in pseudo-code.

```
RecomendConference(Author i, Co-authornetwork c, Set maxC,
number n, Set confs)
  OrderList wscs
  foreach author in c
    decimal wscse = WSCSe(i, author)
    wscs.add(author, wscse)
  truncate wscs for n
  OrderList ranks
  foreach conference in confs
    decimal rank =0
    foreach <author, wscse> in wscs
      if maxC[author]==conference then rank += wscse
    ranks.add(conference, wscse)
  return ranks
```

Algorithm 5 Use Katz index to recommend conference

Eq. 24 can be generalized to sum not only the conference with the highest rank ($MaxC_j$) but the Top-N conferences for author j, denoted $MaxC_j(N)$. In this case, the rank of a conference is defined as:

$$rank(x,i) = \sum_{j \in A \text{ and } x \in MaxC_j(N)} \frac{WSCS_e(i,j)}{pos(x,j)} \quad (25)$$

where $pos(x, j)$ is the position of the conference for the author in the order given by Eq. 22.

4.2.4. MWSCS-based Conference Recommendation Technique

Since computing the $WSCS_e$ score can be very slow for large graphs, we propose to modify Algorithm 5 or, more precisely, only the formula used to calculate $WSCS_e$, by computing the shortest paths from author i to other authors using Dijkstra's algorithm. For this alternative, we modify the definition of the $WSCS_e$ score as follows:

$$WSCS_e(i, j) = \beta^w \quad (26)$$

where w is a length of the shortest path from author i to author j.

The algorithm for this alternative is similar to Algorithm 5, but the results can be very different. Is easy to perceive that, with this alternative in the calculation of the semantic connectivity score, we lose the information about all paths between the authors, except the shortest. For example, in the co-author network of Figure 3 Co-author networkthe pairs of authors (A1, A3) and (A1, A2), using Eq. 26, have the same $WSCS_e$ whereas the pair (A1,A3) should have a larger value; indeed, the path A1, A4, A3 is ignore in the calculation of the $WSCS_e$ using Eq. 26.

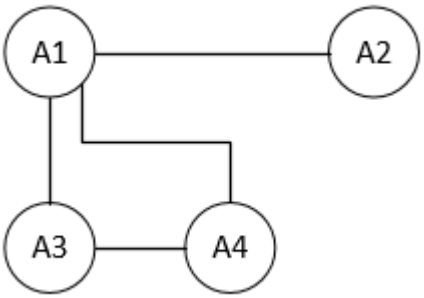

Figure 3 Co-author network

5
Implementation

This chapter summarizes some of the details of the implementation of the Web-based application constructed to enable users to analyze, compare and recommend conferences. Section 5.1 describes the architecture of the Web-based application. Section 5.2 and Section 5.3 present some of the details about the use of the *Neo4j* graph database and the SPARQL endpoint, respectively. Section 5.4 discusses the interface and illustrates how to analyze a conference.

5.1 Architecture

Our application fundamentally uses three external resources. It only requires the *URL* of the *SPARQL* endpoint of the *RDF* store, which must follow the schema describe in Section 3.1. Since, for many conferences, the co-authorship network may not fit in main memory, we store the graph on disk. However, since access to disk can be very slow, we use the *Neo4j* graph database management system, which offers facilities to create and index graphs and implements many graph functions. Finally, we add a new external resource to save all analysis done for a conference thereby avoiding to repeatedly recompute the same analysis.

Figure 4 summarizes the architecture of the application. The *Conferences Data Service* handles the queries to the triple store with information about conferences. The *Co-authorship Network Service* receives the data from the *Conferences Data Service* and handles the queries to the *Neo4j* database. When an analysis is executed, the system saves the results for future analysis; the *Previous Calculation Service* manages these functions. The API of the services is detailed in the Annex.

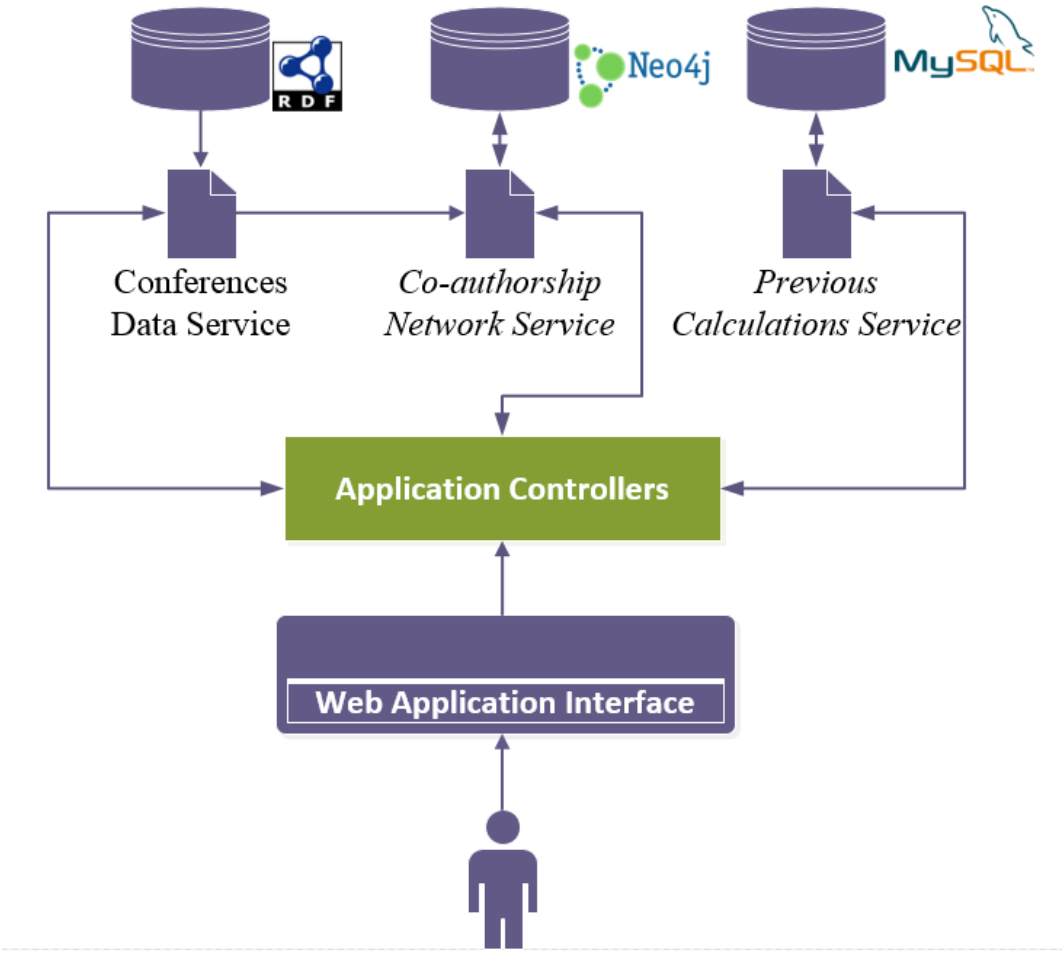

Figure 4 Web Application Architecture

5.2 Co-authorship Network Service

As already mentioned, the co-authorship network service uses *Neo4j* to apply social network analysis to a conference, to compute communities similarity and to compute the $WSCS_e$ score.

Neo4j is an open-source, transactional database with native graph storage and processing, implemented in Java. It is one of the most popular graph database.

Neo4j is accessible from software written in other languages using the Cypher Query Language through a transactional HTTP endpoint.

Cypher is a declarative graph query language that allows to efficiently query and update the graph store. The main clauses used to read from the graph are:

- MATCH: The graph pattern to match. This is the most common way to get data from the graph.

- WHERE: Not a clause in its own right, but rather part of MATCH, OPTIONAL MATCH and WITH. Adds constraints to a pattern, or filters the intermediate result passing through WITH.
- RETURN: What to return.

To add a node (author) to the co-authorship graph of a conference, the Cypher query used is:

```
'CREATE (:`conf_iri` {uri: `author_iri`, name: `author_label` })
```

To add an edge (co-author relation) to the co-author graph of a conference, the Cypher query used is:

```
MATCH (a1:`conf_iri` { uri: `author1_iri`}),
 (a2:`conf_iri` { uri: `author2_iri`})
CREATE (a1)-[:`Co-Author`]-(a2)
```

To mark a node of a connected component in the co-authorship graph of a conference, the Cypher query used is:

```
MATCH node WHERE id(node)= node_id SET node:`conf_iri`+`cc1`
```

To get the nodes, the number of nodes and the edges in the co-author graph of a conference, the Cypher queries used are:

```
1- MATCH (n:`conf_iri`) return n
2- MATCH (n:`conf_iri`) return count(n)
3- MATCH (n:`conf_iri`)-[r:`Co-Author`]->() return count(r)
```

With only one query, we can calculate the clustering coefficient for all nodes of the giant component. The query used is:

```
MATCH (a:`component_mark`)--(b)
WITH a, count(DISTINCT b) AS n
MATCH (a)--()-[r]->()--(a)
RETURN a.uri, count(DISTINCT r)/(n*n-1) AS cc
```

With the first *MATCH*, we find all authors *a* in a connected component, as well as the neighbors *b* of a. With the *WITH* clause, we calculate, for each *a*, the number of its neighbors. With a second match, we find the neighbors of *a* that have a relation *r* with other neighbors of *a*. Finally, we return, for each node, the URI and its clustering coefficient.

To traversal a graph, *Neo4j* uses a *REST API*, whose main parameters are:

- returnType: the kind of objects in the response can be *node*, *relationship* or *path*.
- order: decides in which order to visit nodes; the possible values are *breadth_first* and *depth_first*.

- max_depth: the max depth of the traversal.

For compute the communities similarity we need to find the number of authors in common between communities. The Cypher query used is:

```
MATCH (a1:`%com1%`),(a2:`%com2%`)
WHERE a1.uri=a2.uri
return count(a1) as I
```

Another problem that we can solve using Cypher is to calculate the Katz index. With next Cypher query, we obtain, for each node, all paths with weight less than a given threshold.

```
START n=node(id)
MATCH p=n-[r*]->m
WITH reduce(res=0, x in extract(y in r| y.w)| res + x) as
weight, last(nodes(p)) as dest
WHERE weight < num
RETURN dest, COLLECT(score) AS scores
```

Then, from the result of the query, we count the number of paths with the same weight.

5.3 Conference Data Service

The *Conferences Data Service* uses a SPARQL endpoint to obtain the data. Most of the SPARQL queries used to analyze conferences are showeds and explained in Sections 3.2.1 and 0.

To compare and recommend conferences, we compute the utility matrix using SPARQL Query 9 - Utility Matrix for a conference..

```
SELECT  ?author xsd:float(count(DISTINCT
?paper))/xsd:float(count(DISTINCT ?tpaper)) as ?cant
WHERE {  ?paper a swrc:InProceedings;
             swrc:series <%conf_param%>;
             dc:creator ?author.
          ?tpaper a swrc:InProceedings;
             dc:creator ?author.
}
```

SPARQL Query 9 - Utility Matrix for a conference.

We also need to compute the intersection of the set of authors of two conferences, from which we compute the Jaccard similarity between the conferences. SPARQL Query 10 shows how obtain the number of authors in common between two conferences.

```
SELECT  count(distinct ?author) as ?cant
WHERE {  ?papersC1 a swrc:InProceedings;
            swrc:series <%conf_param1%>;
            dc:creator ?author.
          ?papersC2 a swrc:InProceedings;
            swrc:series <%conf_param2%>;
            dc:creator ?author.
```

SPARQL Query 10 - Intersection of the set of authors of two conferences

```
We also need to compute the weight of the edges in the co-
authorship network (Eq. 22), which is used in the recommendation
algorithms proposed in Section 4.2.
```

SPARQL Query 11 is used for this purpose.

```
SELECT 11 – 10*(xsd:float(?interception)/xsd:float(?c1+?c2-
?interception))
WHERE
{
    {   SELECT count(distinct ?paper) as ?interception
        WHERE {
            ?paper a swrc:InProceedings;
            swrc:series ?c;
            dc:creator <%author1%>;
            dc:creator <%author2%>.
            VALUES ?c { %conf_list% } }
    }
    {   SELECT count(distinct ?paper1) as ?c1
        WHERE {
            ?paper1 a swrc:InProceedings;
            swrc:series ?c;
            dc:creator <%author1%>.
            VALUES ?c { %conf_list% } }
    }
    {   SELECT count(distinct ?paper2) as ?c2
        WHERE {
            ?paper2 a swrc:InProceedings;
            swrc:series ?c;
            dc:creator <%author2%>.
            VALUES ?c { %conf_list% }}
    }
}
```

SPARQL Query 11 - Edges Weight

5.4 Analysis Interface

In a Web application, it is important to present the results to the final user with the correct Web interface to make it easy for him to obtain the desired information about a conference analysis.

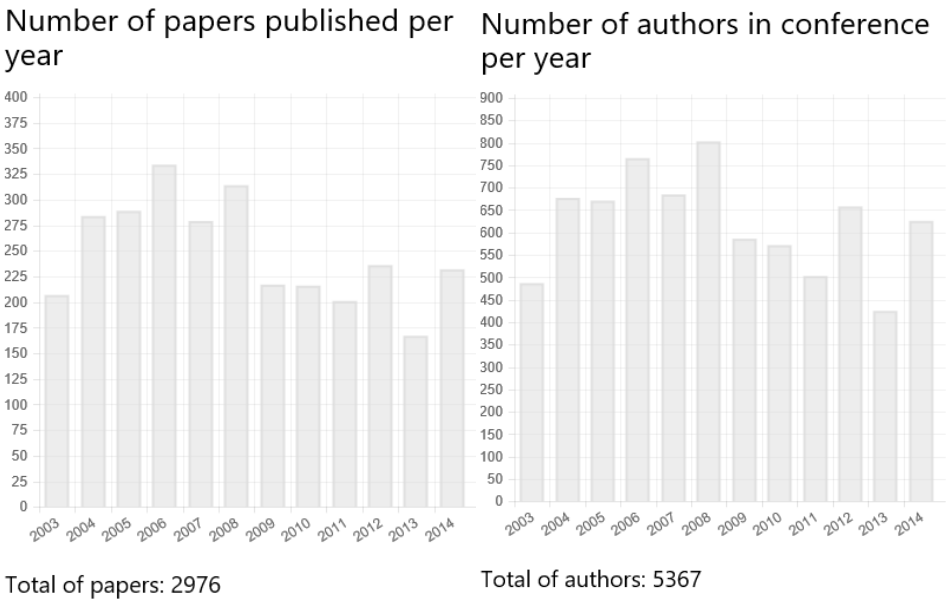

Figure 5 App Interface for the analysis of number authors and paper

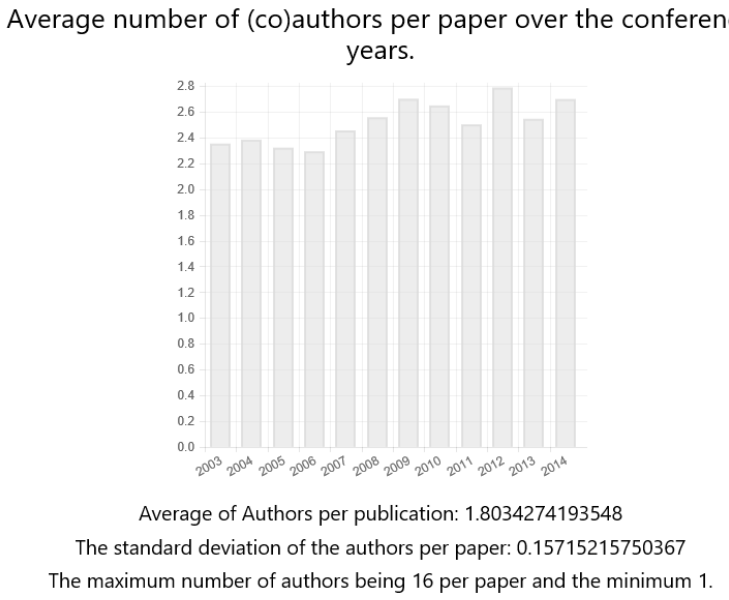

Figure 6 App Interface for the analysis of number co-authors

Following the previous idea, the application shows the number of papers, authors and the average of authors by paper as a bar chart per conference edition. Figure 5 and Figure 6 capture the results for the *ICALT* conference.

For the analysis of the top authors, the interface shows an author cloud where the names of the authors with more papers appear in larger fonts, as in Figure 7.

Figure 7 App Interface for the Top Authors

SNA

Edition	Nodes	Edges	Avg Degree	Density	No of CC	Giant Coeff	Avg Clust Coeff	Dmtr
"http://dblp.l3s.de/d2r/resource/conferences/icalt"	5367	10214	1.9031	0.0003	976	0.3432	0.0463	12
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2003"	487	590	1.2114	0.0024	161	0.0184	0.0656	2
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2004"	677	937	1.384	0.002	215	0.0221	0.1658	4
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2005"	671	909	1.3546	0.002	210	0.0342	0.1498	3
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2006"	766	1309	1.7088	0.0022	210	0.0731	0.687	7
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2007"	685	1072	1.5649	0.0022	170	0.0642	0.2006	7
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2008"	803	1357	1.6899	0.0021	200	0.0373	0.268	6
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2009"	586	937	1.5989	0.0027	146	0.0375	0.1996	5
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2010"	572	851	1.4877	0.0026	153	0.0279	0.1227	4
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2011"	503	791	1.5725	0.0031	126	0.0715	0.158	6
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2012"	658	1077	1.6367	0.0024	168	0.0364	0.1776	4
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2013"	425	685	1.6117	0.0038	101	0.1364	0.1078	6
"http://dblp.l3s.de/d2r/resource/publications/conf/icalt/2014"	626	1055	1.6853	0.0026	160	0.0734	0.1349	7

Avg Degree: Average Degree

No of CC: Number of Connected Components

Giant Coeff: Giant Coefficient

Avg Clust Coeff: Average Clustering Coefficient in Giant Component

Dmtr: Diameter in Giant Component

Figure 8 App-Interface for SNA Analysis

Finally, the results of the SNA analysis are presented as a table for conference editions to easily compare the evolution of the conference. Figure 8 shows the results of the SNA analysis for the ICALT conference.

6
Evaluation and Results

This chapter presents the experiments performed with the similarity measures and the recommendation system solutions. Section **Error! Reference source not found.** describes experiments to compare the similarity measure for communities with other similarity measures presented in Section 4.1.2. Section 6.2 describes experiments to compare the recommendation techniques.

6.1 Experiments with Conferences Similarity

6.1.1. Experimental Setup and Additional Definitions

We evaluated the conference similarity techniques using a dataset with 248 academic computer science conferences, divided into 13 categories (clusters) defined in the List of Computer Science Conferences in Wikipedia [3]. This categorization was adopted as the benchmark. The experiments applied a clustering algorithm to the set of conferences, using each of the conference similarity measures, and compared the clusters thus obtained with the benchmark.
We adopted the hierarchical agglomerative clustering algorithm, that treats each conference as a singleton cluster at the outset and then successively merges (or agglomerates) pairs of clusters, using the similarity measures between conferences presented in Section 4.1.2, until achieving the desired number of clusters, in this case, 13 clusters. To determine how similar clusters are, and agglomerate them, a linkage criterion is used. The shortest value of these links that remains at any step causes the fusion of the two clusters whose conferences are involved. Familiar linkage criteria between two sets of conferences A and B are:

[3] https://en.wikipedia.org/wiki/List_of_computer_science_conferences

- Complete-linkage: the distance between clusters equals the distance between the two conferences (one in each cluster) that are farthest away from each other:

$$\max\{1 - sim(a, b): a \in A, b \in B\}$$

- Single-linkage clustering: the distance between clusters equals the distance between the two conferences (one in each cluster) that are closest away from each other:

$$\min\{1 - sim(a, b): a \in A, b \in B\}$$

- Average linkage clustering: The distance between any two clusters is taken to be the average of the distance between all pairs of conferences:

$$\frac{\sum_{a \in A} \sum_{b \in B} d(a, b)}{|A||B|}$$

Before explaining the measures used to compare how well different data clustering algorithms perform on a set of data, we need the following definitions.

Given a set S of n conferences and two partitions of *S*, X and Y, where X is the correct partition and Y is the partition that results from running an algorithm:

- TP (True Positive) is the number of pairs of conferences in *S* that are in the same set in *X* and in the same set in *Y*
- TN (True Negative) is the number of pairs of conferences in *S* that are in different sets in *X* and in different sets in *Y*
- FN (False Negative) is the number of pairs of conferences in *S* that are in the same set in *X* and in different sets in *Y*
- FP (False Positive) is the number of pairs of conferences in *S* that are in different sets in *X* and in the same set in *Y*

The measures to evaluate the performance of the clustering algorithms using the proposed similarity functions proposed are:

- Rand Index: measures the percentage of correct decisions made by the algorithm.

$$RI = \frac{TP + TN}{TP + TN + FP + FN}$$

- *F-measure*: balances the contribution of false negatives by weighting the recall through a parameter $\beta > 0$.

$$F = \frac{(\beta^2 + 1)PR}{(\beta^2 P) + R}$$

where $P = \frac{TP}{TP+FP}$ and $R = \frac{TP}{TP+FN}$

6.1.2. Results

Figure 9 shows the Rand index obtained by executing the hierarchical agglomerative clustering algorithm with different linkages criteria using the Jaccard, Pearson, Cosine and Communities similarity.

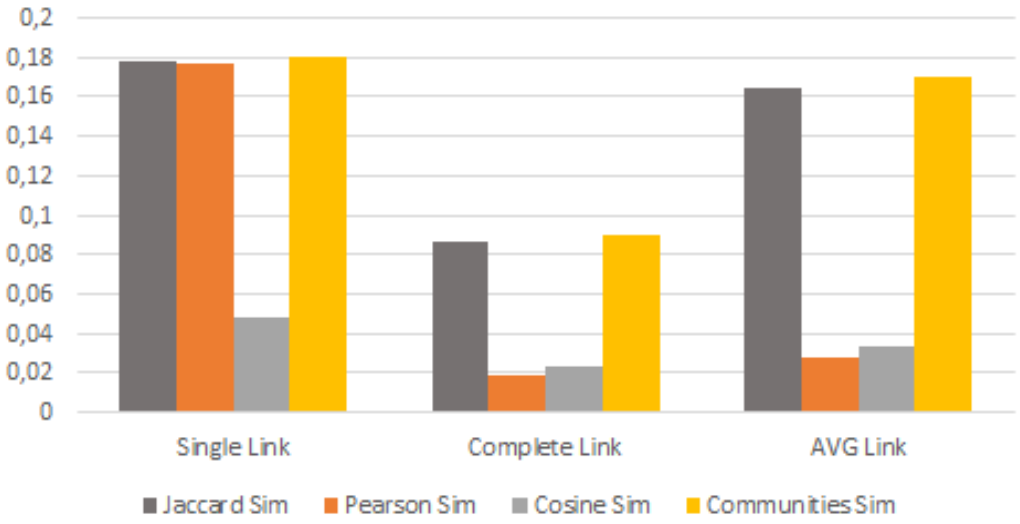

Figure 9 Rand Index of the Clustering Algorithms

Note that, in general, the algorithm based on communities similarity had the best performance, followed by the Jaccard similarity. In this case, the cosine similarity had the worst behavior.

Figure 10 shows the F-measure obtained by executing the same algorithms.

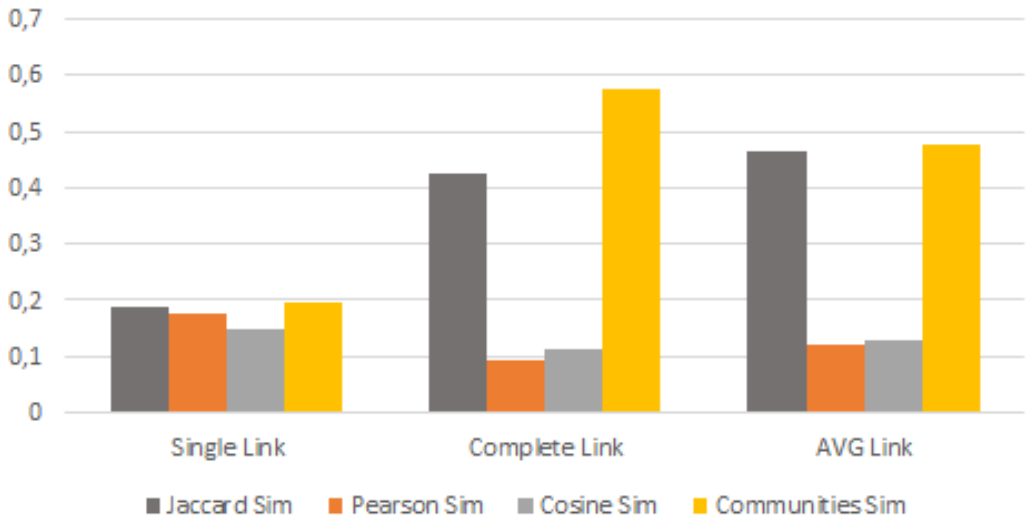

Figure 10 F measure with β=1 of Clustering Algorithms

Analyzing the results presented in Figure 10, we observe that the best performances were also obtained using the communities similarity and the Jaccard similarity measures. The worst performance was obtained using the Pearson similarity measure. The algorithm using the cosine similarity measure achieved the worst performance only with the single link linkage criterion. Note that the objective of these experiments is not to show which clustering method is better, but to indicate which similarity measure is better for the task of clustering conferences.

6.2 Experiments with Conference Recommendation

6.2.1. Experimental Setup and Additional Definitions

Recall that we proposed two families of recommendation techniques. One family is based on classical collaborative filtering and uses adaptations of the familiar similarity measures – Jaccard, Pearson, and cosine similarity – and a new similarity measure, the communities similarity. These techniques are respectively called CF-Jaccard, CF-Pearson, CF-Cosine and CF-Communities. The second family includes two techniques based on the weighted and the modified weighted semantic connectivity, called MWSCS-based and MWSCS-based recommendation techniques.

We evaluated the conference recommendation techniques using the same dataset as in Section 6.1.1, with 248 academic computer science conferences. The golden standard was created by selecting 243 random authors to predict the list of the conferences that an author prefers, constructed using the list of the conferences where the author has more publications. Then, for each author, we delete the information about his publications in the list of the preferred conference and we try to predict this list with our recommendation algorithms. We adopted Luong's most frequent conference technique as the benchmark. We repeat this test technique three times selecting three groups of 243 random authors each one.

The *mean average precision* measures how good a recommendation ranking function is. Intuitively, let a be an author and C_a be a ranked list of conferences recommended for a. Let S_a be a *gold standard* for a, that is, the set of conferences considered to be the best ones to recommend for a. Then, we have:

Prec@k(Ca), the precision at position k of Ca, is the number of conferences in Sa that occur in Ca until position k divided by k

AveP(Ca), the average precision of Ca, is defined as the sum of Prec@k(Ca) for each position k in the ranking Ca in which a relevant conference for a occurs divided by the cardinality of Sa:

$$AveP(\boldsymbol{C}_a) = \frac{\sum_k Prec@k(\boldsymbol{C}_a)}{|\boldsymbol{S}_a|}$$

MAP, the *Mean Average Precision* of a rank score function over all the authors used in these experiments (represented by set *A*) is then defined as follows:

$$MAP = average\{AveP(\boldsymbol{C}_a) \ / \ a \in \boldsymbol{A}\}$$

6.2.2. Results

Consider first the two conference recommendation techniques based on the co-authorship network, the WSCS-based and MWSCS-based recommendation techniques. To compare them, we performed experiments that measured their runtime, accuracy and average precision of the Top-10 conference of an author (thus, in this situation the maximum |Sa| value used in AveP calculation is 10). Figure 4 shows the runtime results of the algorithms that implement these recommendation techniques. Note that the MWSCS-based algorithm is far more efficient than the WSCS-based algorithm.

Table 1 shows the accuracy and MAP of the seven conference recommendation techniques. The two proposed techniques (first two rows of Table 1) have very similar accuracy. In fact, of the 243 authors that we tested for each group, the balance of the correct predictions was 201 against 197, 209 against 206 and 203 against 197 for group 1, 2 and 3 respectively. Based on these results, we may conclude that the MWSCS-based technique is much more efficient and maintains acceptable accuracy level and MAP.

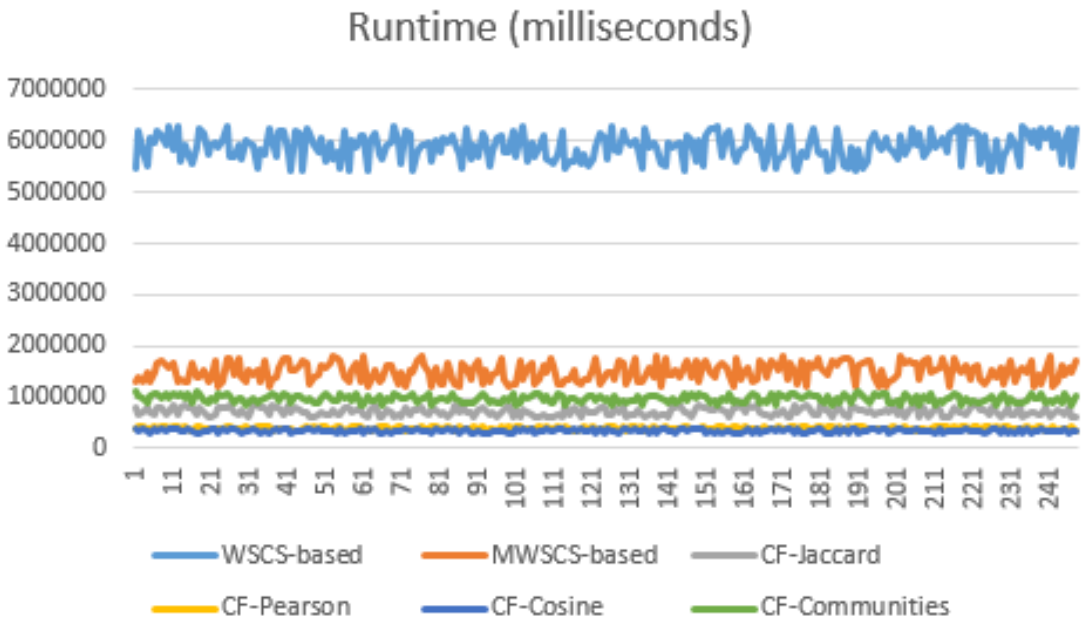

Figure 11 Runtime results of the algorithms.

Table 1 also indicates that the WSCS-based and the MWSCS-based techniques have better accuracy and MAP than the benchmark. The CF-Jaccard and the CF-Communities techniques have very acceptable results and very close to the benchmark. The CF-Pearson and CF-Cosine techniques have poor accuracy.

Table 1 Comparison of the Accuracy and MAP of the recommendation techniques.

Method	Group 1		Group2		Group3	
	Accuracy	MAP	Accuracy	MAP	Accuracy	MAP
WSCS-based	**82.72%**	**80.93%**	**86,01%**	**82.61%**	**83.54%**	**80.33%**
MWSCS-based	**81.07%**	**80.01%**	**84.77%**	**82.10%**	**81.07%**	**78.98%**
CF-Jaccard	78.19%	77.73%	77.37%	75.74%	76.95%	75.10%
CF-Pearson	55.56%	50.21%	55.97%	51.03%	54.73%	50.13%
CF-Cosine	56.79%	51.89%	57.61%	55.51%	56.38%	52.97%
CF-Communities	79.02%	77.93%	79.02%	78.43%	80.25%	78.53%
Benchmark	79.84%	77.88%	82.72%	80.05%	81.07%	78.81%

7
Conclusions

In this work, we presented several methods and a tool to analyze, compare and recommend conferences. To analyze conferences, we stored the conference bibliographic data using a triple-store database that follows a specific RDF schema. Based on this schema, we implemented a system that automatically performs several analyses for a given conference. To compare conferences, we adapted some familiar similarity measures and proposed a new similarity measure between conferences based on the similarity between the co-authorship network communities of two conferences. To recommend conferences, we introduced two families of conference recommendation techniques. The first family is based on collaborative filtering, using the metrics proposed to compare conferences. The second family is based on the relatedness of two authors in the co-authorship network, using the weighted and the modified weighted semantic connectivity score.

We also performed experiments to test the performance of the metrics to compare conferences. The metrics were tests using a clustering algorithm and collaborative filtering algorithm. The lessons learned were:

- For clustering conferences, the algorithms using the Jaccard and the communities similarity measures achieved the best results.
- For recommending conferences, the algorithms using the Jaccard and the communities similarity measures achieved the best results. They do not improve the benchmark, but are very close to it.
- The results of the algorithms using as similarity measure Pearson, and Cosine Similarity have poor results.

To compare the recommendation techniques, we conducted an experiment to measure the runtime, accuracy and precision of the ranking of both strategies. The experiments suggest that the techniques of the second family perform better

than the benchmark and that the technique based on the new modified weighted semantic connectivity score is much faster.

As future work, we plan:

- To developed additional statistical tests to investigate the superiority of the WSCS-based and MWSCS-based techniques.
- To improve the performance of the WSCS-based and the MWSCS-based techniques, which use a co-authorship network built with all authors in the database. For the case of the test with 248 conferences, the co-authorship network had more than 300,000 nodes. An improvement would be to test the conference recommendation algorithms for just one area (or other criterion), to reduce the dimension of the co-authorship network.
- To propose new metrics to populate the utility matrix in order to improve the results of the Pearson, and Cosine Similarity.
- To improve the interface and make the application publicly available.
- To study how to construct RDF datasets in other academic areas.

8
Bibliography

BATISTA, M. G. R.; LÓSCIO, B. F. **OpenSBBD: Usando Linked Data para Publicação de Dados Abertos sobre o SBBD**. Brazilian Symposium on Databases-SBBD. **Anais**...2013

BERNERS-LEE, T. Linked data, in design issues: architectural and philosophical points. **W3C website**, 2006.

BLANCHARD, E. G. **On the WEIRD Nature of ITS/AIED Conferences**. Intelligent Tutoring Systems. **Anais**...2012

BLONDEL, V. D. et al. Fast unfolding of communities in large networks. **Journal of Statistical Mechanics: Theory and Experiment**, v. 2008, n. 10, p. P10008, 2008.

BRICKLEY, D.; GUHA, R. V. RDF Schema 1.1. **W3C Recommendation**, v. 25, p. 2004–2014, 2014.

CHEN, C.; SONG, I.-Y.; ZHU, W. **Trends in conceptual modeling: Citation analysis of the ER conference papers (1979-2005)**. Proceedings of the 11th International Conference on the International Society for Scientometrics and Informatrics. **Anais**...2007

CHEN, C.; ZHANG, J.; VOGELEY, M. S. **Visual analysis of scientific discoveries and knowledge diffusion**. Proceedings of the 12th International Conference on Scientometrics and Informetrics (ISSI 2009). **Anais**...2009

CHEONG, F.; CORBITT, B. **A social network analysis of the co-authorship network of the Australasian Conference of Information Systems from 1990 to 2006**. 17th European Conference on Information Systems (ECIS 2009). **Anais**...2009a

CHEONG, F.; CORBITT, B. J. A social network analysis of the co-authorship network of the Pacific Asia Conference on Information Systems from 1993 to 2008. **PACIS 2009 Proceedings**, p. 23, 2009b.

GASPARINI, I.; KIMURA, M. H.; PIMENTA, M. S. **Visualizando 15 anos de IHC**. Proceedings of the 12th Brazilian Symposium on Human Factors in Computing Systems. **Anais**...2013

GINI, C. W. Variability and Mutability, Contribution to The Study of Statistical Distribution and Relaitons. **Studi Economico-Giuricici della R**, 1912.

HARRIS, S.; SEABORNE, A. SPARQL 1.1 query language. W3C working draft. **World Wide Web Consortium (January 05, 2012), http://www. w3. org/TR/sparql11-query**, 2012.

HENRY, N. et al. 20 Years of four HCI conferences: A Visual Exploration. **International Journal of Human-Computer Interaction**, v. 23, n. 3, p. 239–285, 2007.

HOOVER, E. M. Interstate redistribution of population, 1850--1940. **The Journal of Economic History**, v. 1, n. 02, p. 199–205, 1941.

JACCARD, P. **Distribution de la Flore Alpine: dans le Bassin des dranses et dans quelques r{é}gions voisines**. [s.l.] Rouge, 1901.

KATZ, L. A new status index derived from sociometric analysis. **Psychometrika**, v. 18, n. 1, p. 39–43, 1953.

LEE RODGERS, J.; NICEWANDER, W. A. Thirteen ways to look at the correlation coefficient. **The American Statistician**, v. 42, n. 1, p. 59–66, 1988.

LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. D. **Mining of massive datasets**. [s.l.] Cambridge University Press, 2014.

LOPES, G. R. et al. **Knowing the past to Plan for the Future - An In-depth Analysis of the First 10 Editions of the WEBIST Conference**. (V. Monfort et al., Eds.)WEBIST 2015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies, Lisbon, Portugal, 20-22 May, 2015. **Anais**...SciTePress, 2015Disponível em: <http://dx.doi.org/10.5220/0005447704310442>

LUONG, H. et al. Publication venue recommendation using author network's publication history. In: **Intelligent Information and Database Systems**. [s.l.] Springer, 2012. p. 426–435.

NEWMAN, M. E. J.; GIRVAN, M. Finding and evaluating community structure in networks. **Physical review E**, v. 69, n. 2, p. 26113, 2004.

NUNES, B. P. et al. Interlinking documents based on semantic graphs. **Procedia Computer Science**, v. 22, p. 231–240, 2013.

PROCÓPIO, P. S.; LAENDER, A. H. F.; MORO, M. M. Análise da rede de coautoria do simpósio brasileiro de bancos de dados. **SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, Florianópolis, 2011. Proceedings... Florianópolis**, 2011.

WASSERMAN, S.; FAUST, K. **Social network analysis: Methods and applications**. [s.l.] Cambridge university press, 1994. v. 8

ZERVAS, P. et al. Studying Research Collaboration Patterns via Co-authorship Analysis in the Field of TeL: The Case of Educational Technology & Society Journal. **Journal of Educational Technology & Society**, v. 17, n. 4, p. 1–16, 2014.

9 Annex

In this annex, we describe the API of the *Conferences Data Service*, the *Co-authorship Network Service* and the *Previous Calculation Service*.

9.1 Conferences Data Service API

Get: ConferencesList
Parameters: integer offset integer limit
Result SPARQL query result: the list with the URI and NAME of the conferences

Get: EditionsPerConference
Parameters: string conferences URI
Result SPARQL query result: the list with the URI of the conference editions

Get: TotalConferences
Parameters:
Result integer: number of conference with bibliographic data in the endpoint

Get: Authors
Parameters: string conferences URI
Result SPARQL query result: the list with the URI of the authors with publications

in the conference

Get: TotalAuthors
Parameters: string conferences URI
Result interger the number of author with publications in the conference

Get: AuthorsPerPublication
Parameters: string conferences URI
Result SPARQL query result: the list of pairs with the author URI and publication URI of the authors with publications in the conference

Get: MaxAuthors
Parameters: string conferences URI
Result string the publications URI of the paper with the highest number of authors

Get: MinAuthors
Parameters: string conferences URI
Result string the publications URI of the paper with the smaller number of authors

Get: AuthorsPerPublicationInEdition
Parameters: string conference edition URI
Result SPARQL query result: the list of pairs with the author URI and publication

URI of the authors with publications in the conference edition

Get: TopAuthors
Parameters: string conference URI or conference edition URI boolean edition, true if the first parameter is a conference edition URI interger top
Result SPARQL query result: the list of triple with the author URI, author name and number of publications in the conference or conference edition

Get: NumberPublicationsPerEdition
Parameters: string conference edition URI
Result SPARQL query result: the list of pairs with the edition year and number of publications in this year

Get: NumberAuthorsPerEdition
Parameters: string conference edition URI
Result SPARQL query result: the list of pairs with the edition year and number of authors in this year

Get: UtilityMatrixLine
Parameters: string conference URI
Result SPARQL query result: the list of pairs with the author URI and author preference value of the conference

Get: AuthorsInterception
Parameters: string conference 1 URI string conference 2 URI
Result integer number of authors that publish in both conferences

Get: publicationsPerAuthor
Parameters: string conference URI string author URI
Result integer number of publications of a authors in a conferences

Get: Keywords
Parameters: string conference URI or conference edition URI boolean edition, true if the first parameter is a conference edition URI
Result SPARQL query result: the list of keywords in the publication in the conference or conference edition

9.2 Co-authorship Network Service API

Get: NumberEdges
Parameters: string graphId string edgesLabel
Result interger number of edges of the graph with a specified label

Get: NumberAuthors

Parameters: string graphId
Result interger number of authors in graph

Get: NumberOfConnectedComponents
Parameters: string graphId
Result interger number of connected components in the co-authors network

Get: GiantConnectedComponents
Parameters: string graphId
Result string connected component ID

Get: NumberCommunities
Parameters: string graphId
Result integer number of communities in the co-authors network

Get: CommunitiesInterception
Parameters: string community 1 ID string community 2 ID
Result interger number of authors in common in the communities

Get: AverageDegreeAndDensity
Parameters:

string graphID
Result interger average degree of the co-authors network interger density of the co-authors network

Get: diameter
Parameters: string connected component ID
Result interger diameter of the specified connected component in co-authors network

Get: averageClusteringCoefficient
Parameters: string connected component ID
Result interger average clustering coefficientof the specified connected component in co-authors network

Get: WSCS1
Parameters: string author 1 string author 2
Result double katz index between authors in co-authors network

Get: ShortestPath
Parameters: string author 1 string author 2
Result double shortest path between authors in co-authors network

9.3 Previous Calculation Service API

Get: SNA
Parameters: String graphID
Result List with the co-authors social network analysis Date of calculation

Get: Paths
Parameters: string author 1 string author 2
Result List with the weighs paths between the authors in co-author network

Get: UtilityMatrixLine
Parameters: string conference ID
Result List with the authors preferences for the conference

Get: UtilityMatrix
Parameters: string conference ID string author
Result double author preference for the conference

Get: ShortestPath
Parameters: string author 1 string author 2

Result
double weight of the shortest path between authors