UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CIÊNCIA DA COMPUTAÇÃO

LUÍS ARMANDO BIANCHIN

A Study of Similarities Between Brazilian Computer Science Conferences

Final Report presented in partial fulfillment of the requirements for the degree of Bachelor of Informatics

Prof. Dr. Lisandro Zambenedetti Granville
Advisor

Dr. Ricardo Neisse
Coadvisor

Porto Alegre, December 2012

CIP – CATALOGING-IN-PUBLICATION

Bianchin, Luís Armando

A Study of Similarities Between Brazilian Computer Science Conferences / Luís Armando Bianchin. – Porto Alegre: PPGC da UFRGS, 2012.

49 f.: il.

Final Report (Bachelor) – Universidade Federal do Rio Grande do Sul. Ciência da Computação, Porto Alegre, BR–RS, 2012. Advisor: Lisandro Zambenedetti Granville; Coadvisor: Ricardo Neisse.

I. Zambenedetti Granville, Lisandro. II. Neisse, Ricardo. III. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Diretor do Instituto de Informática: Prof. Luís C. Lamb
Coordenador da CIC: Prof. Raul Fernando Weber
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

"Know thy self, know thy enemy. A thousand battles, a thousand victories."
— Sun Tzu

ACKNOWLEDGMENTS

I would like to thank my advisor Lisandro Zambenedetti Granville and my co-advisor Ricardo Neisse for the knowledge, support and effort which were essential for the whole work.

I wish to thank my colleagues and professors for all the shared knowledge and experience.

I am deeply grateful for the support and confidence by my family, mainly during my studies.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

DBLP	Digital Bibliography & Library Project
JEMS	Journal and Event Management System
SBC	Brazilian Computer Society
TPC	Technical Program Committee
tf	Term frequency
idf	Inverse document frequency
MVC	Model-View-Controller
HTML	HyperText Markup Language
CSV	Comma-Separated Values
TSV	Tab-Separated Values
CSS	Cascading Style Sheets
SVG	Scalable Vector Graphics
RDBMS	Relational Database Management System
UML	Unified Modeling Language
JSON	JavaScript Object Notation
Ajax	Asynchronous JavaScript and XML
SaaS	Software as a Service

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Many research studies focus on the analysis of patterns in research communities. However, we are not aware of any study that performs similarity analysis of conferences considering their research topics. This work presents an innovative solution to measure similarities between conferences based on their research topics. The solution is implemented in a web-based tool that is capable of finding groups (clusters) of conferences classified considering their similarity degrees. Using our tool it is possible to generate visualizations of graphs and charts that show the relations between conferences in an user friendly way. In addition to similarity analysis for a specific conference edition in one year, our tool also supports the analysis of the evolution of the similarity degrees over many editions. To evaluate the proposed solution, a case study was conducted using a Brazilian computer science conference database managed by the JEMS system.

Keywords: Research Communities, Information Retrieval, Simmilarity, Clustering.

Um Estudo de Similaridades entre Confererências Brasileiras de Ciência da Computação

RESUMO

Muitos estudos de pesquisa focam na analise de padrões de comunidades de pesquisa. Porém, não temos conhecimento de nenhum estudo que realiza análise de similaridade de conferências considerando seus tópicos de pesquisa. Esse trabalho apresenta um solução inovadora para medir semelhanças entre conferências baseando-se nos seus tópicos de pesquisa. A solução é implementada numa ferramenta para web que é capaz de encontrar grupos (clusters) de conferências classificadas considerando seus graus de similaridade. Utilizando nossa ferramenta, é possível gerar visualizações de grafos e gráficos que mostram as relações entre conferências de maneira amigável. Além de análise de similaridade de edições de conferências em um ano específico, a ferramenta também suporta a análise da evolução dos graus de similaridade de várias edições. Para avaliar a solução proposta, um estudo de caso foi conduzido usando um banco de dados de conferências brasileiras de ciência da computação gerenciado pelo sistema JEMS.

Palavras-chave: Comunidades de Pesquisa, Recuperação de Informação, Similaridade, Clusterização.

1 INTRODUCTION

Scientific research communities are usually organized around a large number of areas of knowledge. Understanding the internal an external interactions among these communities is an interesting and relatively well exploited challenge. By observing the behavior of scientific communities it is possible, for example, to learn about trends in research topics. Some authors have studied the dynamics of some research communities by analyzing social networks of collaboration, evaluating interdisciplinary research, behavior of individuals and their relationship. For example, Elmacioglu and Lee [9] have analyzed these networks on the context of database community. Barbasi *et al*[2] have evaluated how these networks evolve in time. In particular, Bazzan and Argenta [3] elucidate the characteristics of the Brazilian computer science community in terms of co-authorship of Program Committee members.

In spite of many studies in social collaboration, a study that addresses the relationships between *conferences* considering their research topics is still missing. The perspective of observing research communities according to their research areas in conferences, instead of individual social relationships, would reveal differences among conferences, *e.g.*, in which topics they are related and in which they differ. In addition, one could understand how conferences approximate or dissociate of one another over the years. Given this information, researchers might consider related conferences for submission and cooperation.

The main goal of this work is study the relationships among conferences and their research topics. In order to achieve that, we seek answering the following research questions:

- Which conferences have more alike research topics?
- Which conferences have more alike research topics, considering paper submissions, acceptance, and technical program committee (TPC) members' interest on their research topics?
- How conferences are grouped into areas, and how these areas interact?
- Which conferences became more similar and which became different over the years, again considering submissions, acceptance, and TPC interest on the research topics.

To carry out our study, we use conference information retrieved from the Journal and Events Management System (JEMS)[1] maintained by the Brazilian Computer

[1] https://jems.sbc.org.br

Society (SBC). JEMS hosts several Brazilian and international conferences in many areas of Computer Science. We have retrieved the research topics of a set of relevant conferences and processed them using the information retrieval techniques, such as term weighting defined by Salton and Buckley [17]. We then compute a similarity factor among all observed conferences. The clustering algorithm K-means is applied to identify areas of research and their most common research topics. Finally, visualizations are generated to observe the conference's behavior in terms their similarities along several years, in a way that it is possible to verify the transformations over the years.

The remaining of this work is structured as follows. In Chapter 2, related work, concepts and definitions important to this research are described. In Chapter 3, we present our proposed solution. A prototype implementation is detailed in Chapter 4. A case study is presented in Chapter 5. Finally, in Chapter 6 we present conclusions of this work with final remarks and future directions.

2 RELATED WORK AND CONCEPTS

In this chapter, we start with a discussion about the previous investigations of existing research communities. Most of this work focus on the dynamics of the collaboration networks and the relations among research areas. We also introduce concepts of information retrieval that are used by these research communities and have been proved useful for clustering documents [13] and web-pages [18]. These concepts are the ground basis of our solution to measure the degree of the similarity among conferences.

2.1 Related Work

Several authors have studied the dynamics of collaboration networks. Among the earliest, Newman [14] has used an author network to verify some properties of collaborations between authors using databases from physics and medical areas. The author network was built by taking each researcher as a node and establishing a link when two of them have co-authored a paper together. The results of this work indicates the small world property, where two randomly selected authors are generally separated by a short path. This situation happens even considering such large communities. Moreover, the degree distribution follows a power law, in which many authors have few collaborators and just few authors have many collaborators.

Networks of co-authorship have also been applied to analyze collaborations in other specific communities. Elmacioglu and Lee[9] verified these collaboration networks on the database community and Cotta *et al*[8] investigated the discipline of evolutionary computation.

Barabasi *et al.* [2] analyzed data from a period of eight years of relevant journals of mathematics and neuroscience as a prototype of evolving networks. They learned that the network of co-authorship is scale-free, *i.e.* it also follows a power law. Also, they have evaluated the dynamics of the topology and the evolution of such networks. Among the results, they established that the average distance of authors decreases and the average degree of collaboration increases.

Backstrom *et al.* [1] investigated the community growth using the DBLP dataset. They developed a methodology for measuring the movement of individuals between communities, in which conferences are considered as communities. They found out that movements between communities are closed aligned with changes in the topics of interests. In their work they applied a decision-tree to identify the most important properties that influence these movements.

A comparison of several research areas of computer science has been done by Bird *et al.* [4]. In their work, they describe how interdisciplinary the fields are,

how well defined are the sub-areas inside each field and the collaborative patterns - dominance and assortativity - in each area. They also show the area overlap and migration patterns of authors.

Bazzan and Argenta [3] investigated the properties of the social network of Program Committee Members in conferences of the Brazilian computer science community. The relationship between authors was established also from co-authorship data from the DBLP dataset. Using some metrics as the number of connected components, the degree of the nodes, and the clustering coefficient, they showed that the given network differs from the pattern of the other scientific collaboration networks. One of the findings was that most connected nodes are non-Brazilian TPC members.

All the related work in this section focus on the relationship between individuals inside research communities [14, 9, 8, 2, 1, 3] or on properties of the research areas [4]. We are not aware of any work that focus on the analysis of the similarity and relationships of the *conferences*. Since researchers of specific research communities publish their contributions in a set of main target conferences, we consider it worth to analyze the similarity between these conferences to have a better understanding of the similarities among the research communities.

2.2 Document Representation

In information retrieval and text mining, there are different approaches to represent a document. One of these approaches is the *bag of words* model [19]. In this model, a bag contains words and their counter of occurrences in the document. Therefore, the order in which a word appears in the document and their grammar composition are irrelevant.

Following this model, a document is represented as a vector of non-negative values of the frequency of each of its words. The intuition of this vector is that terms that appear more are more important and descriptive for the document. In our solution described in Chapter 3 we consider this document representation to depict conference information.

Let $D = \{d_1, ..., d_n\}$ be a set of documents and $T = \{t_1, ..., t_m\}$ the set of distinct terms occurring in a particular document d. A document is represented as a m-dimensional vector $\vec{t_d}$. Let $tf(d, t)$ denote a non-negative value of the frequency of term $t \in T$ in document $d \in D$, that is the number of occurences of t divided by the total amount of words in the document. Then the vector representation of a document d_i is presented in Equation 2.1.

$$\vec{t_d} = (tf(d, t_1), ..., tf(d, t_m)) \tag{2.1}$$

The term frequency captures its relevance inside a document. Other aspect is the external relevance of a term. That is, terms that appear frequently in just a few documents tend to be more relevant and specific for that group of documents. In order to capture the importance of such terms the formula *inverse document frequency* (idf) can be used.

Let $|D|$ be the total number of documents and $|\{d \in D : t \in d\}|$ the number of documents where the term t appears, *idf* is presented in Equation 2.2.

$$idf(t, D) = log(\frac{|D|}{|\{d \in D : t \in d\}|}) \tag{2.2}$$

Finally, we assume the term weight as the product of *term frequency* and *inverse document frequency*, as defined in Equation 2.3.

$$w_{t,d} = tf - idf(t, d, D) = tf(t, d) \times idf(t, D) \tag{2.3}$$

The interpretation of this formula is that a high value of $tf - idf$ is achieved by a high term frequency - in the given document - and a low document frequency of term in the universe of documents. One must note that, since the ratio inside *idf log* function is always greater than or equal to 1, the value of *idf* will always be a non-negative value. When a term appears in many documents, the ration inside the logarithm tends to 1, bringing the *idf* and $tf - idf$ closer to 0.

With the given $tf - idf$ values, each document will be described with a vector, where each element is the weight of a term t in document d. To generalize, the weight of a term t in document d is denoted as $w_{t,d}$.

2.3 Similarity Metrics

A similarity/distance metric is one way of estimating the degree of closeness or separation of target objects. This metric should reflect the characteristics that are believed to most distinguish the observed objects. Generally, these characteristics depend on the data or the problem context, therefore, no measure is universally the best.

In order to qualify a distance metric as a measure, some conditions must be satisfied. Let x and y be any two objects in a set and $d(x, y)$ be the distance function between x and y.

1. $d(x, y) = 0$ if and only if $x = y$, the distance is zero if and only if the objects are considered identical (axiom of identity).

2. $d(x, y) = d(y, x)$, the distance is symmetric, that is, the distance from x to y is the same as the distance from y to x. (axiom of symmetry)

3. $d(x, z) \leq d(x, y) + d(y, z)$, satisfy the triangle inequality.

4. $d(x, y) \geq 0$, the distance must be non-negative.

Similarity and distance are opposite concepts. While similarity measures how close objects are, distance measures how far objects are from each other. Several similarity metrics have been researched by the information retrieval community. In this work we consider three metrics: Euclidean Distance, Cosine Similarity and Jaccard Similarity. These are traditional metrics that have been proved to perform well for document and web-page clustering [13] [18]. Moreover, we are interested in having an intial similarity measure for conferences and not to compare the performance of these metrics.

2.3.1 Euclidean Distance

The Euclidean distance is a standard metric for geometrical problems. It represents the ordinary distance between two points in the space and is widely used in clustering problems.

Given two documents represented by vectors $\vec{t_a}$ and $\vec{t_b}$ of size m, the Euclidean distance is defined in Equation 2.4. As previously stated, terms weight are their $tf - idf$ values, $w_{t,a} = tf - idf(t, a, D)$

$$Dis_e(\vec{t_a}, \vec{t_b}) = \sqrt{\sum_{t=1}^{m} |w_{t,a} - w_{t,b}|^2} \tag{2.4}$$

2.3.2 Cosine Similarity

Since documents are represented as vectors, the similarity might be considered as the correlation among these vectors. This correlation may be defined by the cosine of the angle between vectors. Cosine similarity is a very popular measure for text documents [19]. The cosine similarity is defined in Equation 2.5, and when the vectors' coefficients are non-negative, the resulting similarity is bounded between $[0, 1]$.

$$Sim_c(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a} \cdot \vec{t_b}}{|\vec{t_a}| \times |\vec{t_b}|} \tag{2.5}$$

One property of this measure is that it does not depend on the length of the document. That is, documents composed by the same words, but with different quantity of words are treated identically. For example, if one takes two identical documents d and merge them, creating a new document d', then $Sim_c(d, d') = 1$. This means that d and d' are considered identical.

2.3.3 Jaccard Index

The Jaccard index measures the ratio of the number of shared attributes by the number of all attributes. In other words, it measures the cardinality of the intersection divided by the cardinality of the union of the attributes. The Jaccard index of two vectors $\vec{t_a}$ and $\vec{t_b}$ is described by Equation 2.6.

$$Sim_j(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a} \cdot \vec{t_b}}{|\vec{t_a}|^2 + |\vec{t_b}|^2 - \vec{t_a} \cdot \vec{t_b}} \tag{2.6}$$

The resulting value of the Jaccard index also ranges between 0 and 1. This means that, when the value is 0, the vectors are disjoint and so completely different. And when the value is 1 the vectors are identical.

2.4 Clustering

Clustering is defined as the task of assigning a set of objects to groups - called *clusters* - such that objects in each particular group are more similar to each other than between those of other groups [17]. There are two main kinds of clustering algorithms: hierarchical and partitional.

Hierarchical algorithms have their output as a nested sequence of partitions, with a single cluster at the top and clusters of individual objects at the bottom. Each intermediate level may be viewed as splitting a cluster from a higher level, or combining clusters of the next lower level. The result of hierarchical clustering are usually represented as a tree, denoted as dendogram. For document clustering, this dendogram acts as a taxonomy or a hierarchical index.

To generate hierarchical clusters, two strategies can be adopted:

- **Agglomerative**: the *bottom up* approach. Each object starts in its own cluster and at each step the closest cluster is merged. This requires the measurement of the cluster distance or similarity.

- **Divisive**: the *top down* approach. All objects start in one single cluster and at each step a cluster is partitioned until only singleton clusters of individual objects remain. In this case, one needs to decide which cluster to split and how to perform the split.

Partitional clustering creates a division (un-nested) of the data objects. With a predefined number of desired clusters, the partitional approaches typically find all these clusters at once. As opposed to the hierarchical ones, which splits a cluster to get two clusters or merge two clusters to get one.

To design our solution we choose only the partitional algorithm K-means. The reason for this choice is the simplicity and speed of this algorithm, and its large application in existing document clustering approaches [13]. Also, similar to our approach for the chosen similarity metrics, our objective is to have an initial method for grouping of conferences into areas, and not to evaluate the performance of existing clustering techniques.

In K-means, a cluster is represented as a centroid object, which is computed as the average of all objects in the cluster. Note that this centroid usually does not represent a real object. This algorithm is better suited for handling large datasets than hierarchical clustering algorithms, due to relatively low computation requirements.

The algorithm adjusts the clusters aiming to minimize the residual sum of squares (RSS). Let the objects be represented by $\vec{t_i}$ and its closest centroid as $\mu_{c^{(i)}}^{\rightarrow}$. The RSS is defined as the sum of the squares of the distances of the objects and its closest centroids, as stated in Equation 2.7.

$$RSS = \sum_{i=1}^{m} Dis(\vec{t_i}, \mu_{c^{(i)}}^{\rightarrow})^2 \tag{2.7}$$

As noticed, *K-means* uses distances measures to minimize the interior distance of the clusters. This makes the distance metric a fundamental component of the algorithm. However, some transformation must be applied to transform similarity measures to distance values. Euclidean distance is already a distance metric. For cosine similarity and Jaccard index, as they are both bounded to $[0, 1]$, we take $Dis = 1 - Sim$ as the corresponding distance value.

The K-means algorithm may be described in a simplified manner as follows:

1. *Select centroids*: randomly select k data objects as the *cluster centroids*

2. *Cluster assignment*: assign each object to the cluster with closest centroid.

3. *Move centroid*: compute new centroids as the mean of all objects inside each of the clusters.

4. Repeat (2) and (3) until the centroids have not moved

In the *move centroid* step, it is necessary to calculate the mean of all object inside the cluster. This is achieved as defined in Equation 2.8. That is, the sum of all vectors assigned to the cluster divided by the number of vectors.

$$\mu_k = \frac{1}{n}[\vec{t_{k_1}} + \vec{t_{k_1}} + ... + \vec{t_{k_n}}] \tag{2.8}$$

In each step of the algorithm the cost function decreases, and after a number of iterations it will *converge*, indicating that the clusters haven't changed. However, the generated clusters are only assured to be *locally optimal* for the given data set and the initial seeds. This implies that different choices of the initial seeds may result in different final partitions.

Several approaches have been investigated to overcome this problem. A comparison of some initialization methods was made by Pena *et al.* [15]. Their findings is that the random initialization is a good option due to its high performance. The idea is to run K-means several times initializing with random samples and pick the clustering that gave the lowest cost (RSS).

Other problem is that choosing the number K of clusters can be quite arbitrary and ambiguous. One way to overcome this is by using the Elbow method, that is plot the cost as function of clusters K. The cost function should decreases as the number of cluster increases and then flatten out. The idea is to choose K at the point which the cost function starts to flatten.

3 SOLUTION

The objective of our work is to provide a way to measure the similarity between conferences and to visually identify overlapping areas of conferences. In order to achieve that, our proposed solution design consists of activities that perform intermediate steps that are necessary to achieve our objective. The squared boxes represented in Figure 3.1 are the activities we proposed in our solution. The first activity is the retrieval of conference information from a database, followed by normalization, weighting, similarity calculation, clustering, and visualization of charts and graphs. The visualizations show the similarity between conferences in an user friendly way.

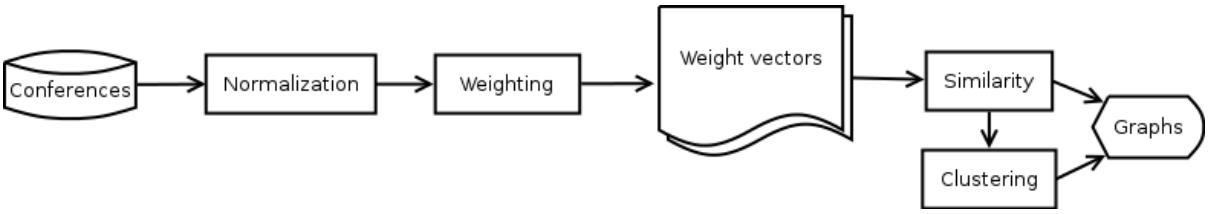

Figure 3.1: Components of the solution

3.1 Conferences

A conference held in a particular year has some information associated to it. For each ocurance of a particular conference we consider in our solution the following information:

- Research topics: list of topics where each topic consists of a couple of words describing a specific sub-area of interest of the conference;
- Technical Program Committee (TPC) members: each TPC member is responsible for selecting from the research topics list the set of topics of interest or expertise;
- Papers: each paper has a title, an abstract, and a list of topics selected from the research topics list by the authors when the paper is submitted. A paper has also an status that indicate the paper is current under review, has been accepted or reject for publication in the conference.

In our proposed solution we use the information from the *research topics* to characterize a conference. We apply the concepts of information retrieval presented

in the previous section making an analogy where conferences are the equivalent to documents, and words of the research topics the equivalent to terms.

The data from the *Research Topics* must first be processed in order to be available to similarity measurements. This is done by the the activities of *Normalization* and *Weighting* that generate the *Weight vectors* ready for similarity measurements.

3.2 Normalization

The first step in the normalization activity of the research topics consists of the *tokenization*. In this step, the research topics list of each conference is retrieved and are divided into terms (*i.e.* tokens or words). In this step we also remove the punctuation marks and all characters are converted to their lower case equivalent.

To illustrate this step we show an example input and output of the tokenization step for a conference of the area of computer architecture and high performance. Figure 3.2 shows the research topics of this conference that were used as input for the tokenization step and Figure 3.3 shows the output.

Benchmarking Performance Measurements and Analysis Cache and Memory Architectures Fault-Tolerant Architectures and Systems Grid Computing High Performance Applications Languages, Compilers and Tools for Parallel and Distributed Programming Large Scale Simulations Load Balancing and Scheduling Operating Systems Processor Microarchitectures Parallel and Distributed Algorithms, Architectures, Interconnection Networks, Routing and Communication Pervasive and Heterogeneous Computing Performance Measurements and Analysis Reconfigurable Systems

Figure 3.2: Research topics extracted from a conference

application-specific architectures benchmarking performance measurements analysis cache memory architectures fault-tolerant architectures systems grid computing high performance applications languages compilers tools for parallel distributed programming large scale simulations load balancing scheduling operating systems processor microarchitectures parallel distributed algorithms architectures interconnection networks routing communication pervasive heterogeneous computing performance measurements analysis reconfigurable systems

Figure 3.3: Tokenized research topics

After the tokenization step, we need to ensure all terms are in a common language. This is necessary because the JEMS system hosts conferences with data both

in the English and the Portuguese languages. To overcome this issue, our solution includes a translation step where all terms are translated into the English language.

Considering the output of the translation steps, we observed that some terms are semantically not descriptive. Examples of these are words like *and*, *for* and *to*. Mainly these terms are conjunctions, articles and prepositions. These irrelevant words are called *stop words*. We do not consider these terms relevant for our similarity analysis and we propose in our solution a filtering step that removes all these terms.

For the example of conference given above, only the word *for* is removed.

After the filtering step we perform a stemming step where we apply the Porter's suffix-stripping algorithm [16]. Thereby, terms with different endings will be mapped into a single term. For example, the terms *networks*, *networking* and *network* will be mapped to the stem *network*. This allows that terms with the same steam/root - and thus semantically identical - be depicted as the same term.

Following the example, after stemmed, the data will look like Figure 3.4.

application-specif architectur benchmark perform measur analysi cach memori architectur fault-toler architectur system grid comput high perform applic languag compil tool parallel distribut program larg scale simul load balanc schedul oper system processor microarchitectur parallel distribut algorithm architectur interconnect network rout communic pervas heterogen comput perform measur analysi reconfigur system

Figure 3.4: Stemmed terms

Infrequent terms are also removed. The reason is that these terms are not very descriptive about the document (conference). Moreover, rare terms might introduce noise in the clustering process and also increase the similarity computation time. Therefore, we removed all the terms that occurred only once in the dataset.

3.3 Weighting

After all terms are normalized, we need to assign weights that represent the relevance of each term to describe the topics of the conference. In this step, we consider the following four aspects (heuristics) to measure the relevance of a term:

- Number of occurrences in the topics list
- Amount of persons from the TPC that are interested in that term
- Amount of submitted papers in a topic having that term
- Amount of accepted papers in a topic having that term

In this way, each term will have its frequency calculated by only one of the given aspects. For the given example, if one considers the number of occurrences in the topics list, the amount of repeated terms are presented in the first column of the Table 3.1.

With the given counting, this activity calculates the $tf - idf$ weight for each term. The tf values of the given example are presented in the third column of

Term	Count	Frequency
architectur	4	0.082
perform	3	0.061
system	3	0.061
comput	2	0.041
parallel	2	0.041
distribut	2	0.041
analysi	2	0.041
measur	2	0.041
processor	1	0.020
microarchitectur	1	0.020
algorithm	1	0.020
oper	1	0.020
balanc	1	0.020
schedul	1	0.020
interconnect	1	0.020
rout	1	0.020
heterogen	1	0.020
reconfigur	1	0.020
pervas	1	0.020
communic	1	0.020
load	1	0.020
network	1	0.020
scale	1	0.020
applic	1	0.020
languag	1	0.020
high	1	0.020
grid	1	0.020
memori	1	0.020
fault-toler	1	0.020
compil	1	0.020
tool	1	0.020
larg	1	0.020
cach	1	0.020
program	1	0.020
application-specif	1	0.020
benchmark	1	0.020
simul	1	0.020

Table 3.1: Terms with count and frequency

Table 3.1. *idf* values were not calculated because they depend on other conferences making it hard to describe.

3.4 Similarity Measuring

After the *normalization* and *weighting* activities, the output will be a *weight vector* for each conference. As previoulsy mentioned, this vector contains the $tf-idf$ values for each term.

With the given *weight vectors*, it is possible to measure the degree of relationship among pairs of conferences. For this solution it was implemented the metrics of Euclidean Distance, Cosine Similarity and Jaccard index, presented in the previous chapter.

For the Euclidean Distance, one must note that it is a distance metric. To convert it to a similarity metric we take add one and take the inverse, that is $Sim_e = \frac{1}{1+Dis_e}$. Adding one is imporant because when two objects are the same, the Euclidian Distance evaluates to zero and that would raise a divison by zero. Note that distance zero is the same as the similarity been one.

3.5 Clustering

One activity of this solution is to find grouping of conferences. That is, clusters of conferences that look more similar. This is done by using the K-means algorithm described in Section 2.4 and a similarity metric. The input will be the weighting vectors presented in Section 3.3 and a number K of required clusters. The output will be a K clusters, each with several conferences and a weighting vector describing the average of the conferences inside the cluster.

3.6 Visualizations

The final component of the solution is the visualizations. This is an important part because it will allow to perform analysis of the relationships of conferences.

One very popular way to represent data is by using charts. Charts provide a way to graphically represent symbols, generally in a two-dimensional space. However, one must notice that a conference does not directly map to a position in the two-dimensional space. That is, a conference does not have a explicit pair of coordinates (x, y).

Two kinds of visualization were implemented: *Force Graphs* to compare all conferences, and charts to verify the evolution of similarity among years.

At the force graph, each conference is represented as a node and the weights of the links between nodes are the similarity between conferences. After applying the force layout, similar conferences will be arranged closer. Also, the conferences' nodes are colored according to their cluster. In this way, conferences that belong to the same group, will have the same color. One may note that in this approach the nodes are automatically positioned in the screen.

In line charts, the similarities are plotted, showing which pairs of conferences neared and which distanced. The X-axis is represented by years and the Y-axis is the similarity, ranging from 0 to 1. Each line correspond to the similarity measure of a pair of conferences on several years. For generating the chart, the user might select

to pin a conference and visualize the evolution of the similarity of all conferences compared to the selected conference.

4 IMPLEMENTATION

The solution we propose was implemented as a tool for retrieving conference information of the JEMS system database and visualizing similarity between these conferences. This tool is directed to users who have some knowledge of information retrieval techniques. In this chapter we start with a general overview of the design and a high-level description of the related technologies we have used in the development of this tool. This general overview is followed by a description of the system design model and implemented architecture. Finally, we show an user guide that supports possible users of the tool.

4.1 System Architecture and Related Technologies

The tool implemented was conceived as a web application using the model of Software as a service (SaaS) described in Figure 4.1. In this model, the software and its data might be hosted somewhere in the cloud. Users access the system by using a simple web browser client that request the site through the internet.

The application is divided in three tiers. Presentation tier comprises a web server responsible for handling requests, calling the logic and delivering the web page. The logic tier is where the core of the application is executed. And, finally persistence tier is responsible to store data important for the application.

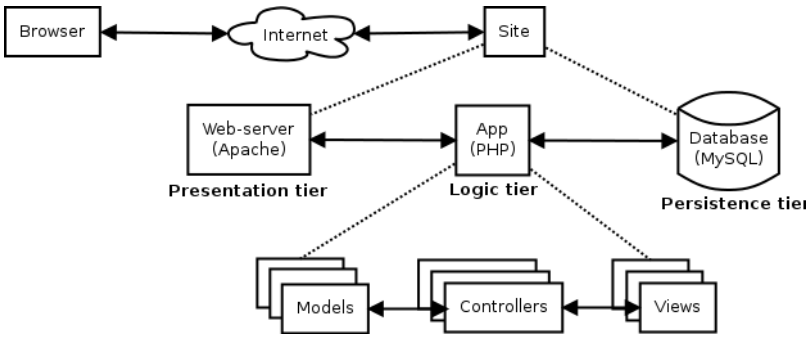

Figure 4.1: Diagram of a classical SaaS architecture

4.1.1 Programming Language

The programming language PHP is a general-purpose server-side scripting language first released in 1995. Its main purpose is the generation of dynamically web pages. PHP scripts run in the server-side and generate HTML web pages that are rendered in client web-browsers.

PHP was chosen due to the existence of an initial draft implementation of the solution we propose that could be re-used by us in the implementation of our tool. Furthermore, the author has some experience with this language and it was more convenient since no time had to be spent in learning a new technology. Nonetheless, PHP is a flexible language featuring simplicity, portability, object orientation and dynamic typing that was adequate considering our implementation requirements.

4.1.2 Application Framework

A framework is a set of libraries and components providing a generic functionality. It is able to use an architectural style to inherit it advantages: modularity, low coupling and high cohesion. This allows the rapidly development of an application. Also, one can achieve faster development by using proven patterns and reusing functionality provided by components provided with the framework.

For the implementation, we chose the framework CakePHP (v2.1) [6]. The reason was also that the author has some experience with this framework. CakePHP provides an easy programming interface to retrieve the required information for the tool we have implemented from the JEMS database.

CakePHP is an open-source framework licensed under MIT License with the goal to provide an environment for fast development of web applications. It has a highly active community and was conceived based on the design patterns of MVC (Model View Controller) and Active Record.

4.1.2.1 Model-View-Controller Design Pattern

The MVC [11] architecture aims to separate the application logic from the data representation and user interaction. This allows to produce more cohesive software and to easy manage changes in the data access, application logic, and presentation layers.

The model represents the data management layer. It manages the storing, retrieving and handling of data. A model is mapped to an entity or table in the database.

The view represents the visual presentation of the model for human users. One specific view could, for example, highlight some data attributes and ignore attributes that are not relevant for a specific user interaction. The view layer render a presentation using as input the data provided from the model. Therefore, the view implementation must know the semantics of the attributes of the model it represents. Multiple views can be defined for different user interactions using the same model as input.

A controller is what connects the user to the system and implements the logic of the application. It is responsible to handle user input, contacting the model and passing its information for presentation (view).

4.1.2.2 Active Record

Active record [10] is an architectural pattern used by systems that rely on relational databases for data storage. With this pattern, each table on the database is mapped into one class of the system and the fields of the table are mapped to attributes of the class. The CRUD operations (reate, Read, Update and Delete) on the table are mapped to methods of the class. In CakePHP, the Active Record

pattern is implemented by the model class, which is the usual way also found in other MVC frameworks.

4.1.3 Database

The conference information that is used as input for our tool is retrieved from the JEMS database. This database is implemented in the production JEMS server using the MySQL server version 5.5.19. MySQL [7] is an open source relational database management system (RDBMS) that has been first released in 1995.

Since the focus of our work was initially to perform similarity analysis of conferences that use the JEMS system the use of a MySQL interface was mandatory to access JEMS data. The MySQL syntax is very simple, and complies with the SQL (Structured Query Language) standards. MySQL is also one of the standard databases that is supported by CakePHP, making it simple and easy to retrieve the conferences information in our tool.

4.2 Database modeling

To retrieve the data necessary for our tool, we only need to access a sub-set of the JEMS database. The JEMS tables required by our tool and their relationships are presented in the UML (Unified Modeling Language) diagram in Figure 4.2.

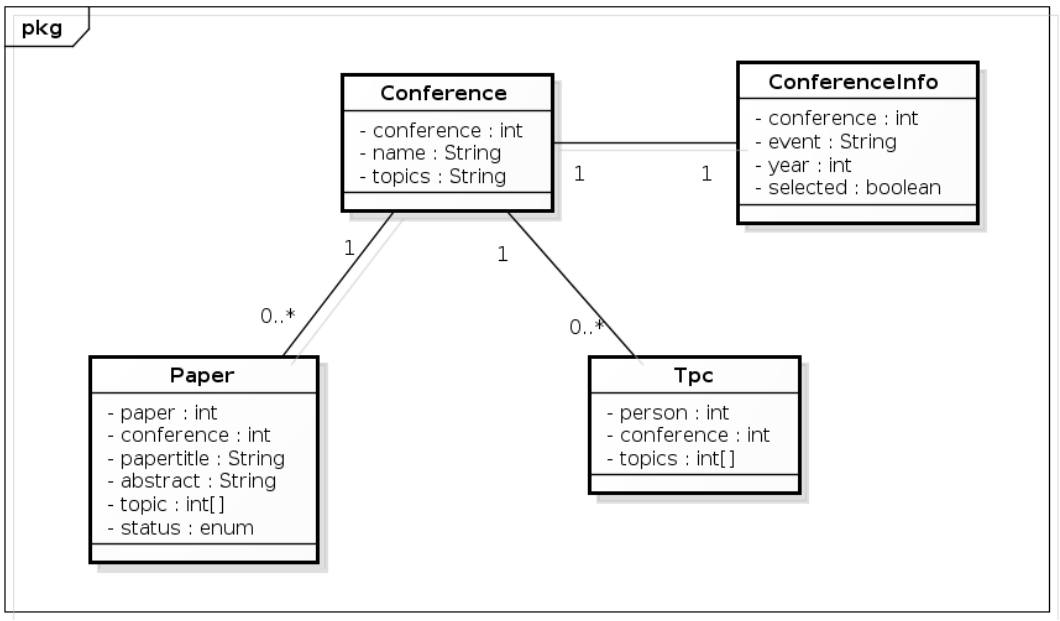

Figure 4.2: Class diagram of JEMS database

The *Conference* table describes the data of a conference held in a particular year. In spite of this entity has 115 fields in the JEMS database, for this solution it is important to consider only 3 fields. The primary key is the field $conference$ and is a unique integer identifier. The *name* field contains the conference's name with the year of occurrence. The *topics* field is a string with topics separated by new line ($\backslash n$) characters.

The table *Tpc* is used to record the association of a person to a conference with the technical program committee role and includes also the topics of interest of this

person for the conference. These topics are used by conference chair to assign papers to this person for revision. They are stored in the field *topics* represented by a string containing the index of the topics separated by commas. This index indicates the position of the topic in the topics string (from *Conference* table).

The table *Paper* stores information of a paper submitted to a conference. The fields are a primary key (*paper*), the conference to which the paper has been submitted, the title (*papertitle*), an abstract summarizing the paper content, a comma-separated list of topics that the paper covers (index of the topic), and the review status.

In the *ConferenceInfo* table, the data from the conference is normalized. That is important since the *Conference* table has only a field for the conference name including the year without distinguishing the regular occurrence of a conference in different years (e.g. name = "SBRC 2012"). The *ConferenceInfo* table contains the conference name in the field *event* and the year that it occurs. This allows us to easily query all occurrences of a specific event (e.g. event="SBRC" year="2012"). The normalization of the conference name was automatically done considering common patterns used in the JEMS system by chairs to specify the conference name. The field *selected* holds a boolean value describing if the conference should be considered for analysis.

4.3 Class architecture

In this section we describe the class architecture and the source files of our tool implementation. First we describe internal components of the CakePHP Framework followed by a description of the classes implemented by us (model and controller). Finally, we describe the implemeantion of the similarity visualizations (views).

4.3.1 Details of CakePHP Framework

As previously stated, CakePHP follows the MVC pattern. Hence, models - that represent tables from the database - inherit from the class *AppModel*, which implements the ActiveRecord pattern. The application controllers inherit from the class *AppController*. These both classes provide useful methods and attributes that are used in the inherited classes.

In the models classes there are some attributes that describe the relationship between models. There is the attribute *hasOne* which defines a relationship of one-to-one and the foreign key is at the associated model. The attribute *belongsTo* also defines a one-to-one relationship, but in this case the foreign key in the current model. There is also the attribute *hasMany* and *hasAndBelongsToMany*, used when there is an associative table. The model also has the methods *save* and *saveAll*, these methods take an array of data and save the records to the database. There are also methods *read()* (read only one record), *find()* (read several records), *update()* and *delete()*.

The controller class import models by specifying their names in the attribute *uses*. There is also the attribute *helpers* which import helper objects that assist in the view presentation, as for formatting (HTML, Number), forms creation (Forms), session handling (Session), etc. The controller usually reads data from the models and put in a variable; this variable is sent to the view by using the method *set()*. Also, each method defined in the controller is denoted as an action and generally

has a correspondent view.

Views are only template files, that is, they basically contain HTML markup embedding some PHP to make it dynamic. For each action in the controller (method), there view file withe .ctp extension (CakePHP View Template).

CakePHP implements a routing scheme that translates URLs to calling actions from controllers. The default routing is */<controller>/<action>/<param>*. That is, if one access *posts/view/1*, it will map to the *view()* action of the *PostsController* passing 1 as an argument.

CakePHP enforces some conventions in the file structure. The application code must be put inside the folder *app*. Inside this folder, there are several folders, among them: *Controller*, *Model*, *View*, *Lib*, etc. The architecture of system's classes is presented in Figure 4.4. Note that each subsystem correspond to a folder inside the *app* directory and each class has a file with its name and the *.php* extension.

4.3.2 System Classes

The tool was projected following CakePHP file structure. The project diagram is described in Figure 4.3. As previously mentioned controller classes inherit from *AppController* and Model classes inherit from *AppModel*. The model classes are the tables from JEMS database, which were discussed in Section 4.2.

The implementation has only two controllers. *ConferencesController* allows to list, visualize, select and normalize conferences. Whereas, *SimilaritiesController* retrieves conference data, calculates similarities, generate the clusters and prepares for visualization.

The core logic of the solution was implemented inside the Lib folder. The implemented classes are described in Figure 4.4. There are classes for normalization, weighting, similarity measurement and clustering.

Inside the model *Conference* there is the method *getVectors()*. This method fetches conference data from the database and, according to the selected scope to be considered for the relevance of the term, it will build the weight vectors. If the scope is *submitted or accepted papers*, it fetches data from the model *Paper*. If the scope is *TPC interest*, it fetches data from *Tpc* model. With the given data, for each conference, it will be applied the methods from the *Normalizer* and *TfIdf* classes.

The normalization step of the solution consists of three static classes. First the *Normalizer* receives the topics list by the method *tokenize()*, which converts characters to their lower case equivalent and returns an array of words. Then *normalizeKeywords()* method is called, which normalizes each word. The single word normalization is performed by the *normalizeKeyword()* method, which verify if the word is a stop word by calling *isStopWord* method from the *StopWords* class. If not, it tries to translate using the *Translator* class, and invoke the method *stemWord* from the *Normalizer* class. At the end of this process, the final output is an array of stemmed English terms.

With the given terms and weights of topics - that is the number of TPC members, submitted papers or accepted papers of each topic - it is calculated how relevant (descriptive) each term is, as described in Section 2.2. That is achieved by computing the term frequency in the static method *calculateTf()* of the class *TfIdf*.

After having the data ready for each conference, one may merge conferences by a window of years. For example, selecting a step of two, would merge each two conferences into a single conference. That is performed by the method *mergeByStep()* of

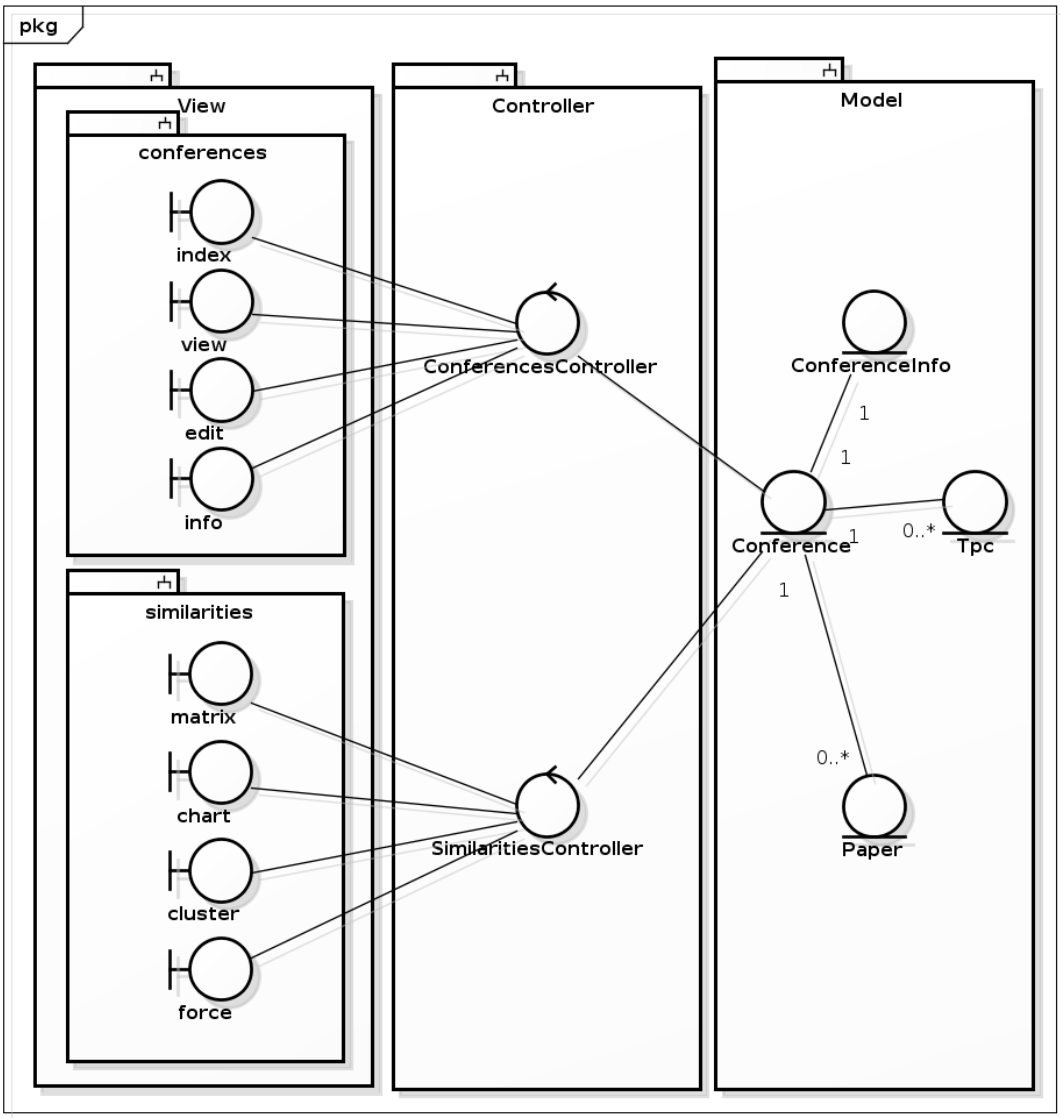

Figure 4.3: Project Diagram

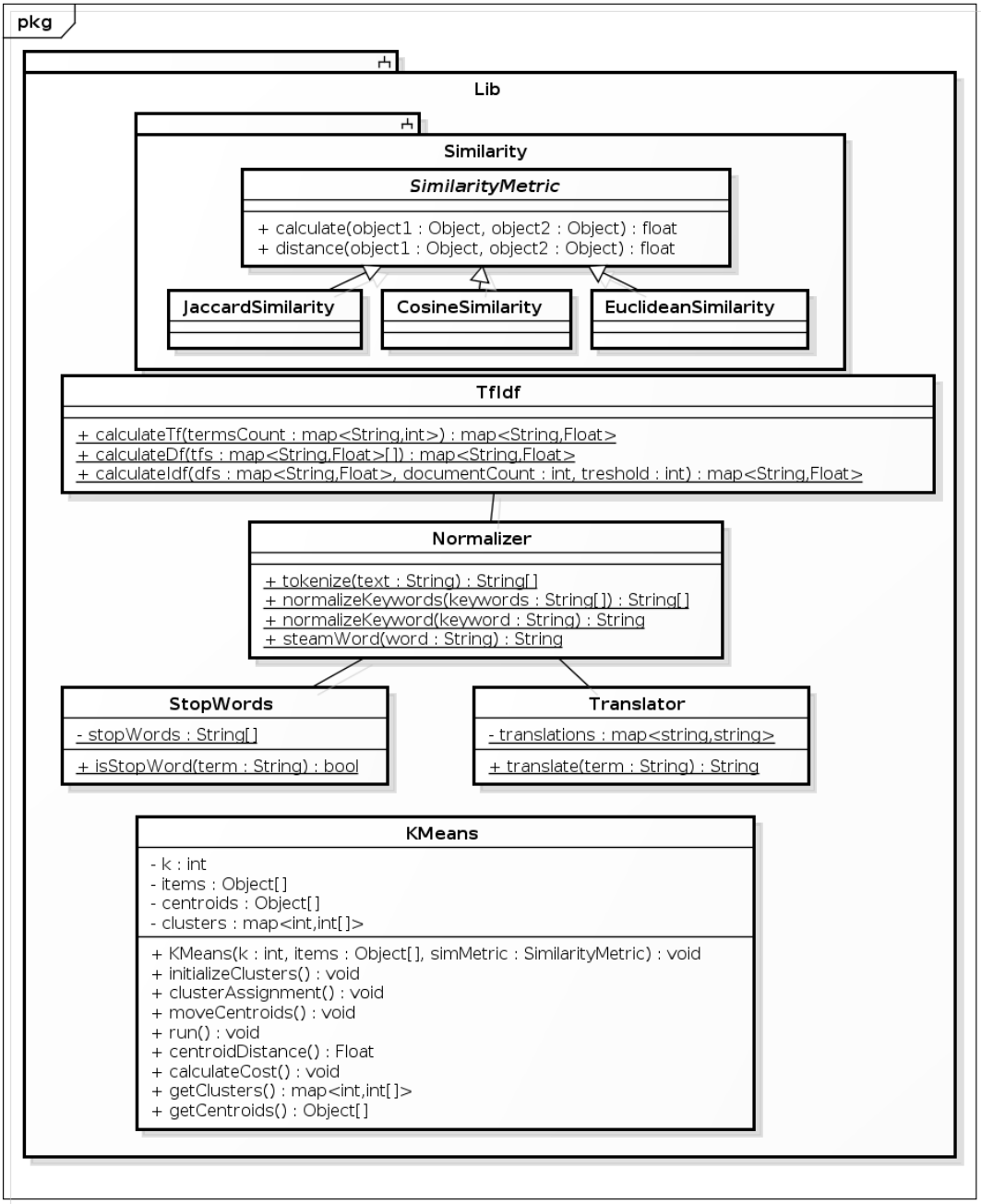

Figure 4.4: Diagram of the implementation

the model *Conference*. Inside of it, it is also calculated the *idf* value of each term by using the method *calculateIdf()* of the class *TfIdf*. Then, the $tf - idf$ is calculated by multiplying the *idf* value with the previous calculated tf value.

For similarity calculation, there is a main abstract class *SimilarityMetric*. This class has only two methods: *calculate* (to calculate the similarity measure) and distance (to calculate the distance measure). These methods take two arrays describing the comparison objects. Each metric was implemented as a class that inherits from *SimilarityMetric* and implements the measurement methods.

K-means algorithm was implemented in the class *KMeans*. An object of this class is constructed using as arguments: the number K of clusters, the weighting vectors (*items*) and a *SimilarityMetric* instance. First, it calls the method *initializeClusters()*, that randomly select the first K centroids. After that, it iteratively calls *clusterAssignment()* and *moveCentroids()* until the cost - calculated by *calculateCost* - has not changed. This class is used in the action *cluster* of the *SimilaritiesController*.

4.3.3 Visualization

The visualization was implemented in the view files, importing some JavaScript libraries and CSS files. When the action *force* from the *SimilarityController* is called, it fetches the conference data by calling *getVectors()* of *Conference* model, measures similarity (at the controller) and delivers for visualization. The view is rendered in the user's browser displaying the force graphs.

To display graphs, we use the library *D3.js* (Data-Driven Documents) [5]. *D3.js* is a JavaScript library created with the purpose of displaying digital data into graphical dynamical forms. It relies on SVG (Scalable Vector Graphics), JavaScript and CSS (Cascading Style Sheets) languages.

After the conference data is fetched, it is adapted to describe a graph. As previously stated, the nodes in this graph represent conferences and the weight of the links represents the similarity degree among two conferences. This graph is then exported into a JSON (JavaScript Object Notation) following the *D3.js* convention. This data is captured by an *Ajax* (Asynchronous JavaScript and XML) call from *D3.js* and rendered as a graph.

A dynamic force layout is applied to this graph. The goal of this layout is to iteratively assign new positions to the nodes such that those nodes that are connected by stronger weights are placed closer to each other.

Moreover, our tool also supports line charts, to allow quick visualization of the similarity evolution. They are at the action *charts* of the *SimilaritiesController*. The chart was implemented using Google Chart Tools [12]. Each line in the chart is the similarity of a pair of conference on several years. Besides being quick visualized in a line chart, the data might be exported to CSV (comma-separated values) or TSV (tab-separated values) that is suitable for input to external analysis tools (e.g. MS-Excel).

4.4 User interface

This section aims to present the main pages and functionality of the implemented system. The user interface is composed by screens for Conference Selection, Similarity Matrix Visualization, Similarity Chart Visualization, and *Clustering and Graph*

Visualization.

4.4.1 Conference Selection

In this interface the user may select the conferences he wants to analyze. The conference list page, shown in Figure 4.5, displays the selected conferences. The attributes highlighted are its name, the first part of the topics list, normalized name and year.

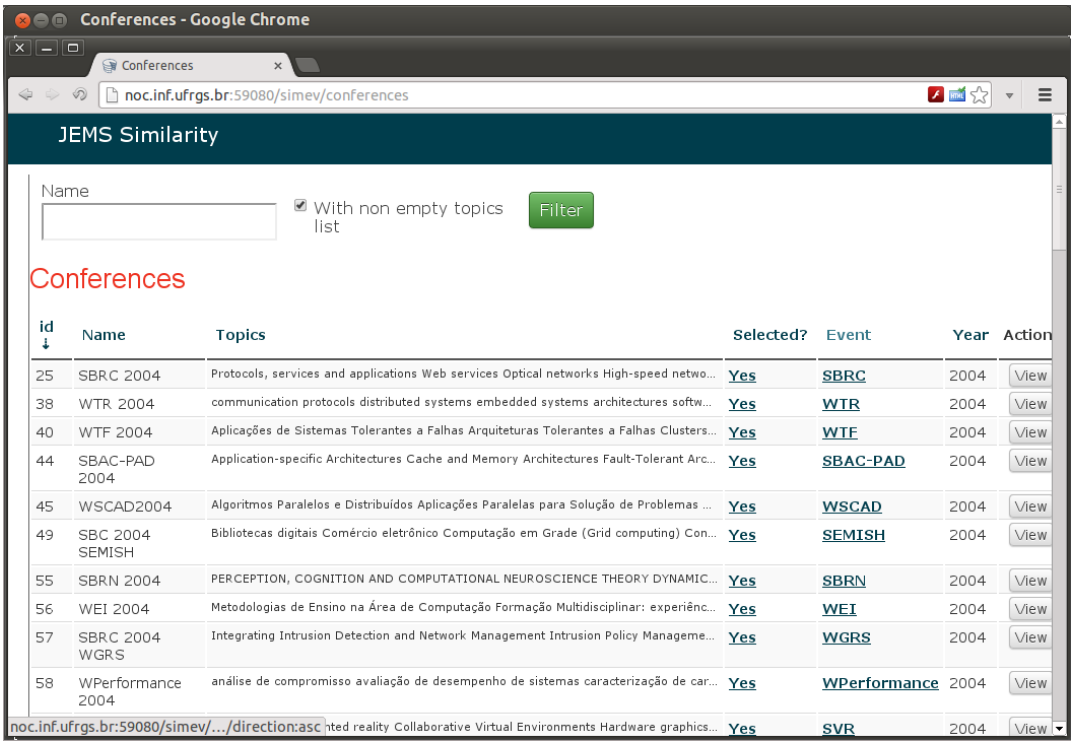

Figure 4.5: Conference list page

To select a new conference the user may enter the name and click the *Filter* button. After that, a list of conferences matching that name is listed. To change the selected status, the user must click in the column selected of the item. The user might also specify the conference name (for the *event* field) and the year.

4.4.2 Similarity Matrix

In this part of the system, the user can generate the similarity of the combination of all conferences with each other. The available options for generating this matrix a user might configure are:

- The similarity metric;
- Which field to use for term weighting;
- The output format, that might be a HTML table, CSV file, graphml file or JSON;
- Which conferences to consider - when none is selected, all conferences will be considered;

- Which years to consider - also when none is selected, conferences of all years will be considered;
- A threshold value, *i.e.* the similarity value is not displayed when it is bellow *threshold*/100;
- The step size, to merge conferences by year, *e.g.* if this value is 2 it will merge conferences each 2 years;
- The step, that is which year to show, if it is 0 it will show for all the period.

Figure 4.6 shows the Similarity Matrix screen with filters and options for the similarity output.

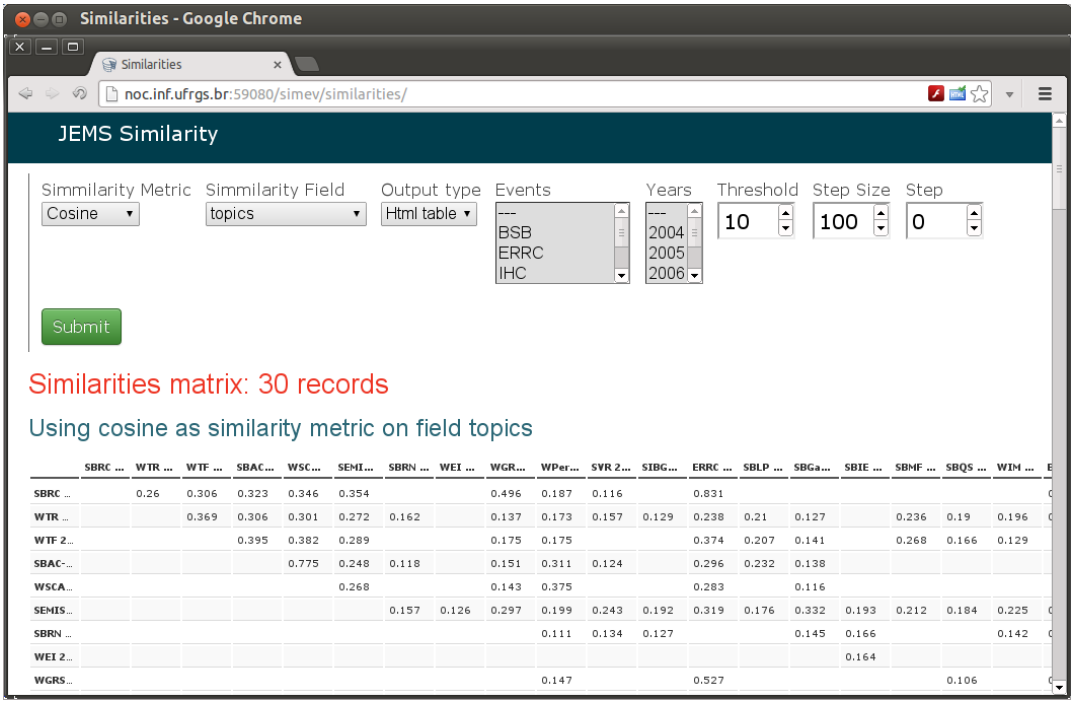

Figure 4.6: Filter and options to configure similarity output

4.4.3 Charts of comparison

In this screen, the user may select a conference and view the similarity evolution compared to all other conferences. The Figure 5.3 shows an example of a generated chart.

Moreover, the user might choose to view the similarity evolution of pairs of conference which have the greatest similarity value. And also, to view those pairs that have the greatest standard deviation, that is, those that have a greater variation of similarity values.

4.4.4 Clustering

To perform the clustering, the user must select how many clusters he wants. The number K of clusters is a required input for the K-Mean algorithm. Also, it is possible to configure in this interface all the options already described in the Similarity

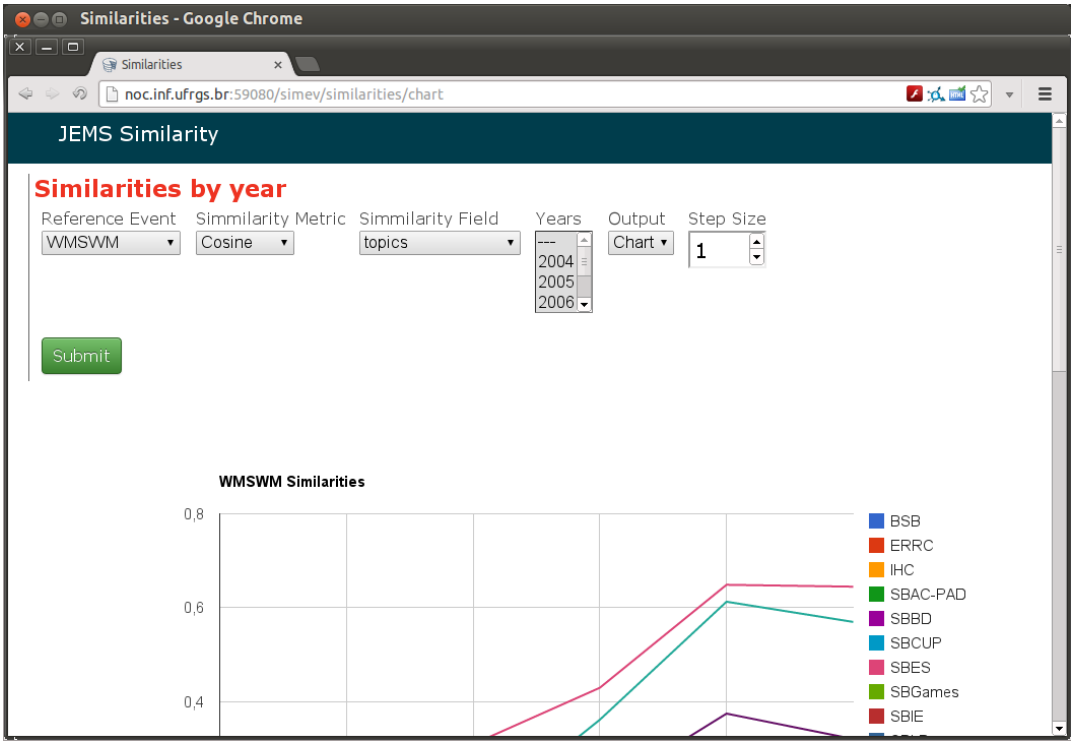

Figure 4.7: Chart of comparison of similarities with a given event

Matrix screen, introduced in Section 4.4.2. Figure 4.8 shows the Clustering screen with the given options.

After the clustering has been performed, it will display the number of iterations required for convergence, the cost (Residual Sum of Squares) of each iteration and amount of time spent on clustering. It also displays the contents of each cluster. That is, which conference is inside of them and the most descriptive terms of the cluster. This terms are those with the higher weights from the clusters' centroids.

4.4.5 Graph Visualization

After the clustering activity is completed, the user can visualize the force graph. In the Figure 4.9 an example of a force graph is shown. As we can see in this figure, the colors represent the cluster in which the conference is located, and the conferences represented with the same color are located in the same cluster. Using the force graph in combination with the coloring allows us to analyze the soundess of the clustering technique. If the clustering technique is sound, conferences in the same cluster will be placed closer.

It is possible to specify parameters for the graph layout. The charge value is a negative value that results in node repulsion. The strength specifies the strength (rigidity) of the links, this value is multiplied by the similarity represented by the link weight.

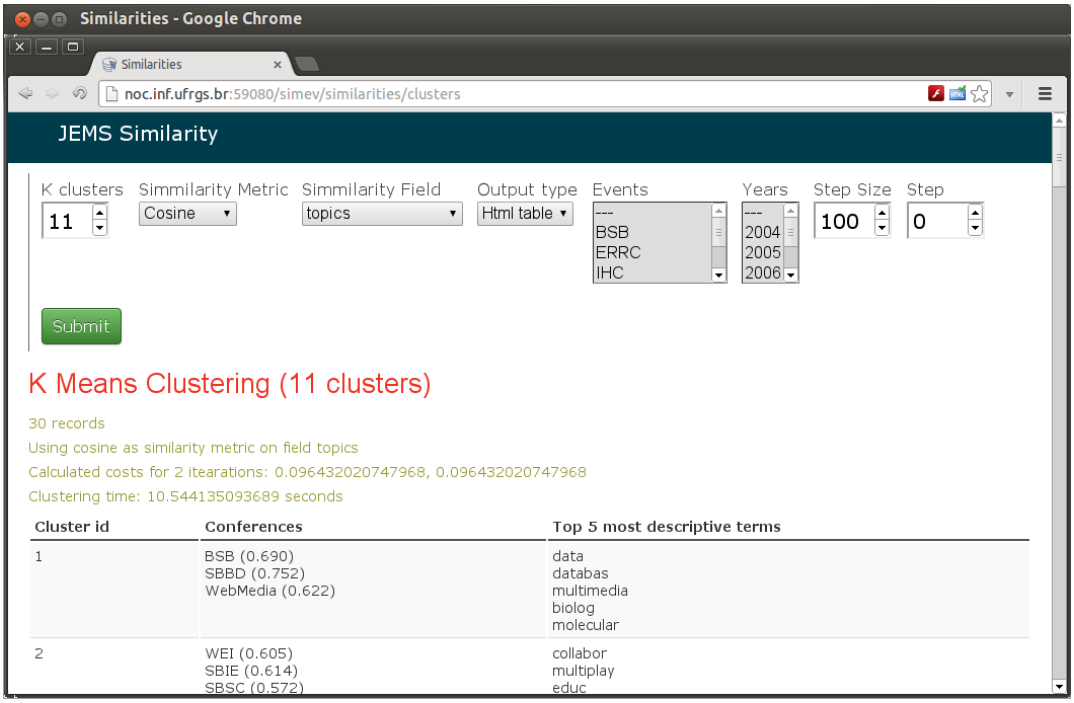

Figure 4.8: Clustering

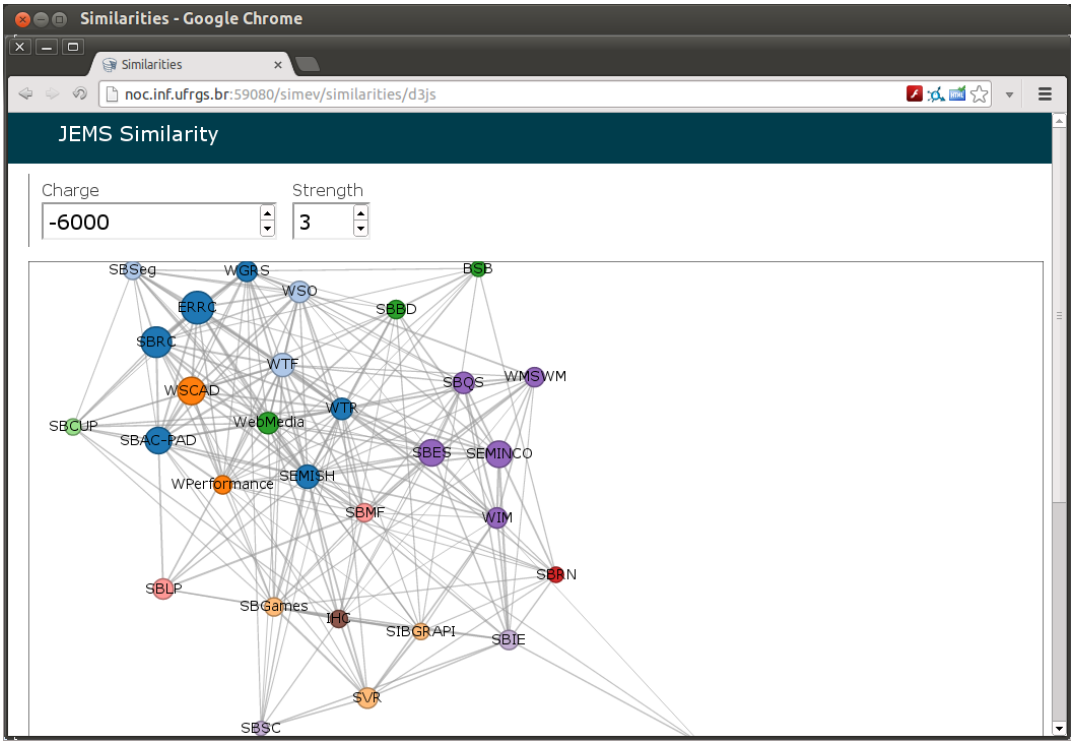

Figure 4.9: Graph Visualization

5 CASE STUDY

In order to evaluate our solution, we performed similarity analysis of a set of conferences using our tool in a case study. In this case study, we were able to find some characteristics of the relationship between conferences. These findings are not exhaustive, our aim was to demonstrate that our solution performs adequately and is useful.

5.1 Conferences

The conferences we selected for this study are in the computer science discipline and are promoted by the Brazilian Computer Society (SBC) [1]. We selected those that are registered in JEMS database with more than one occurence. That was 30 conferences that occurred from 2004 to 2011, in a total of 169 conference ocurrences. Table 5.1 presents the conference acronyms and the years in which they were selected. The complete description of the conference name is described in Appendix A.

Table 5.2 describes how many conferences were selected by each year. As one may note from this data, there are conferences that only have data in the recent years (*e.g.* SBCUP,)and others that only have old data (*e.g.* WSCAD, SBRN, SBLP, WSO and WIM). This is the case because some conferences are new and their first edition happened just recently, and other conferences do not exist anymore or were merged with other events. Also, some conferences do not have registered topics of interest.

5.2 Similarity Measurements

The worst performance of the similarity metrics we observed was the Euclidean distance. This metric took to many resources to be computed and generate similarity values that did not differentiate conferences adequately. This was also observed in other researches of document clustering [18] [13]. For this study, we choose to use only the metric cosine similarity. The reason of this is to limit the length of this analysis and also because cosine similarity performed very well.

For the first measurement, all the years of each conference have been merged into one single object. That is, the topics lists of a conference that happened several years were merged to a unique topics list.

To perform the clustering analysis using K-means, we first need to find out which is a good number of K clusters. To achieve this number, we used the Elbow

[1] https://www.sbc.org.br

Conference	**Years**
BSB	2005, 2007, 2008, 2010, 2011
ERRC	2004, 2005, 2006, 2007
IHC	2006, 2008, 2010
SBAC-PAD	2004, 2005, 2006, 2007, 2008, 2009, 2010
SBBD	2005, 2006, 2007, 2010
SEMISH	2004, 2005, 2006, 2007, 2008, 2009, 2011
WEI	2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011
WPerformance	2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011
WIM	2005, 2006, 2007, 2008
WSO	2006, 2007, 2008
SBES	2005, 2006, 2007, 2008, 2009, 2010, 2011
SBGames	2004, 2006, 2007, 2008, 2009, 2010, 2011
SBIE	2004, 2005, 2006, 2007, 2008, 2009, 2010
SBLP	2005, 2006, 2007, 2008
SBMF	2004, 2006, 2007, 2008, 2009
SBQS	2005, 2006, 2006, 2007, 2008, 2009, 2010, 2011
WMSWM	2006, 2007, 2008, 2009, 2010, 2011
SBRC	2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011
WGRS	2004, 2006, 2007, 2008, 2009, 2010, 2011
WTF	2004, 2005, 2006, 2007, 2008
WTR	2004, 2005, 2006, 2007, 2008
SBRN	2004, 2006, 2008
SBSC	2006, 2007, 2008, 2008, 2009, 2010, 2011
SBSeg	2005, 2006, 2007, 2008, 2009, 2010, 2011
SEMINCO	2006, 2007, 2008, 2009, 2010, 2011
SIBGRAPI	2004, 2005, 2006, 2007, 2008, 2009, 2010
SVR	2004, 2006, 2007, 2008, 2010
WebMedia	2006, 2007, 2008, 2009, 2010
WSCAD	2004, 2005, 2006, 2007
SBCUP	2009, 2010, 2011

Table 5.1: Selected Conferences

Year	**Amount**
2004	16
2005	18
2006	29
2007	27
2008	27
2009	18
2010	20
2011	14

Table 5.2: Amount of conferences selected by year

method. That is, running the K-means algorithm for several K values, measuring the residual sum of squares (RSS) value and plotted these costs on a line chart. The RSS measures how far items are from their cluster centroid. In this way, by analyzing the chart, one may notice how the clustering improve by having a large number of K clusters. This chart is presented in Figure 5.1. The x-axis represents the number of K clusters and the y-cluster represents the correspondent RSS value.

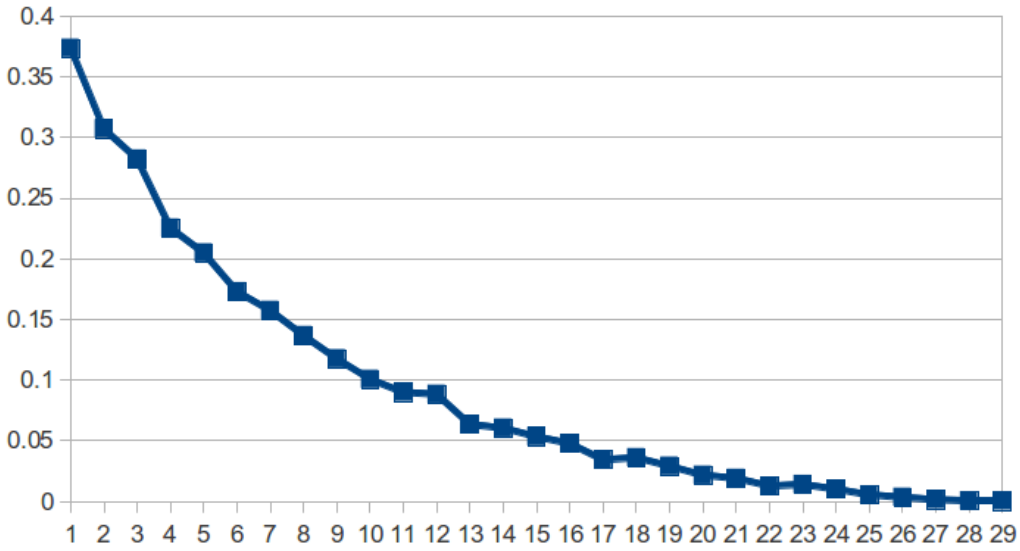

Figure 5.1: RSS when using several K clusters

From this chart it is possible to observe that the numbers of K clusters in the chart flatten are at 11, 13 and 17. These are points where incrementing the number of clusters does not improve much the clustering result. We chose 11 since it is the smaller of then.

Using $K = 11$ clusters, K-Means was applied. The output clusters are described in Table 5.3. The first column contains the cluster identifier, the second there are the conferences that are inside this cluster (and the similarity with the cluster centroid) and the third contains the five most descriptive terms of the cluster.

With the clusters generated, a visualization of the force graph was rendered and is shown in Figure 5.2. The links represent the similarity between pairs of nodes (conferences), and when they are smaller than 0.1 they are not represented. The color code represents the membership of the node to one particular cluster. That is, nodes with the same color belong to the same cluster.

First, one should notice how clusters are disposed in the force graph. Most of the clusters have their conferences placed closer, since they have the same color code. This indicate that the force graph and the clustering agreed that those conferences are very similar.

Cluster 1 is composed mostly by conferences of the area of networks (ERRC, SBRC, WGRS) and security (SBSeg). It is mainly represented by SBRC (Brazilian Symposium of Computer Networks) and ERRC (Regional School of Computer Networks). WTF is a workshop of Fault Tolerance that happens inside of SBRC (Brazilian Symposium of Computer Networks). WTR is a workshop of Real-Time systems that also occurs inside of SBRC.

Cluster	Conferences	Top 5 most descriptive terms
1	ERRC (0.755) SBRC (0.789) SBSeg (0.540) WGRS (0.610) WTF (0.532) WTR (0.452)	manag servic network secur protocol
2	SBES (0.673) SBQS (0.749) WMSWM (0.714)	mainten qualiti softwar product reengin
3	SBAC-PAD (0.762) SBCUP (0.396) WPerformance (0.710) WSCAD (0.687)	perform pervas parallel capac load
4	SBGames (0.694) SIBGRAPI (0.786) SVR (0.667)	imag game render realiti 3d
5	SBBD (0.802) WebMedia (0.790)	databas data multimedia document tempor
6	SBSC (1.000)	collabor group support work workflow
7	SBLP (0.529) SBMF (0.628) WEI (0.739)	experi formal teach program languag
8	SBIE (0.753) SEMINCO (0.786) WSO (0.404)	educ oper graphic learn artifici
9	IHC (0.742) SEMISH (0.825)	interact human challeng interfac impact
10	WIM (1.000)	medicin health telemedicin sign intelig
11	BSB (0.827) SBRN (0.653)	biolog structur sequenti dynam percept

Table 5.3: Clusters output of K-Means algorithm

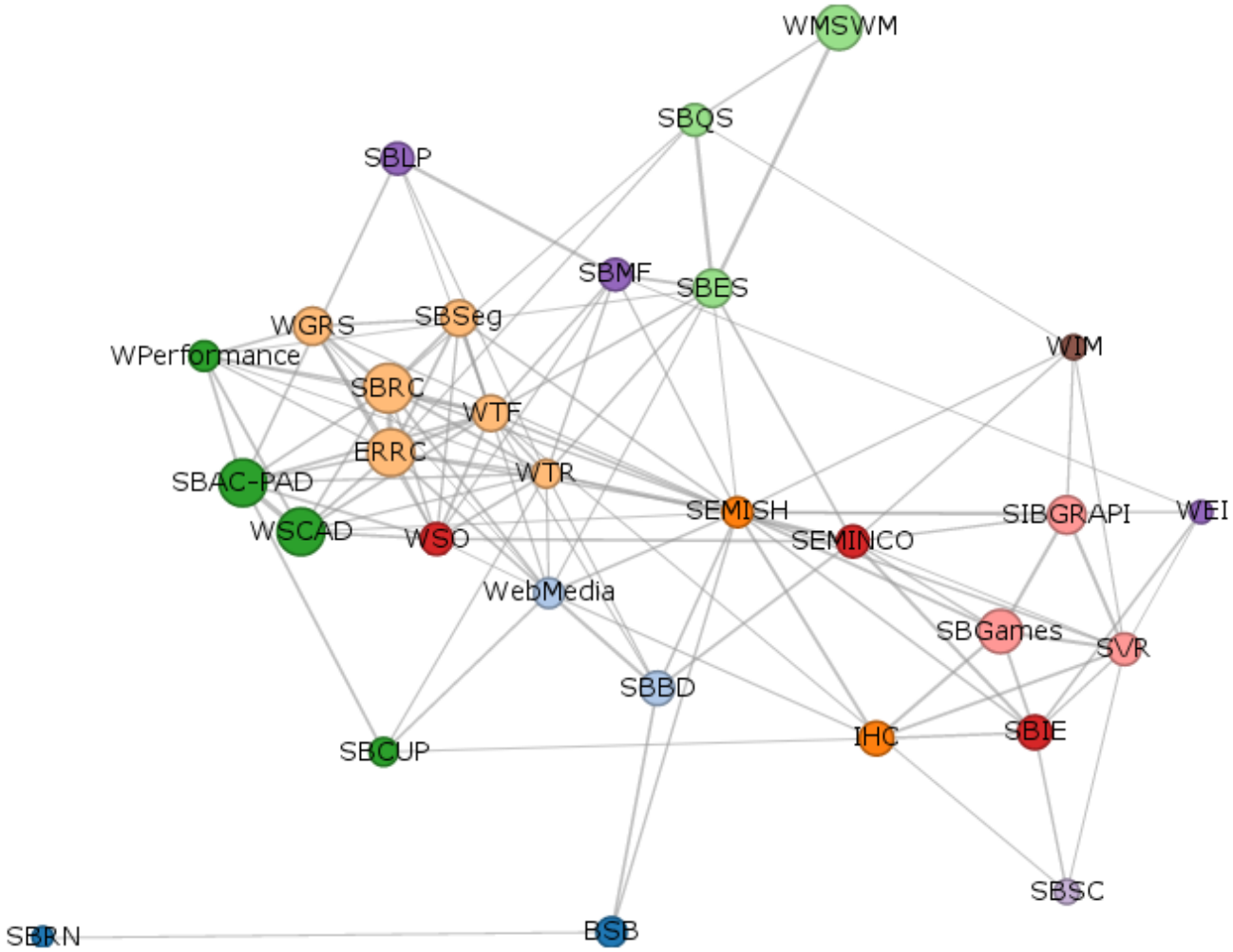

Figure 5.2: Force Graph using cosine similarity and topics

Cluster 2 is represented by the software engineer conferences. SBQS is the Brazilian Symposium of Software Quality. SBES is the Brazilian Symposium of Software. WMSWM is the Workshop on Modern Software Maintenance and occurs inside of SBQS.

Cluster 3 contains conferences of Computer Architecture and High Performance Computing. One may notice that the SBCUP (Brazilian Symposium on Pervasive and Ubiquitous Computing) might be considered an outlier in this cluster, since it has the small similarity to the centroid and is farthest positioned in the force graph.

Cluster 4 has conferences of games (SBGames), graphics (SIBGRAPI) and virtual reality (SVR). All of these conferences have very related subjects. Also, the conferences are placed closer in the force graph.

Cluster 5 contains SBBD (Brazilian Symposium on Database) and WebMedia (Brazilian Symposium on Multimedia Systems). They are very related conferences and actually they are jointly organized every year.

There are two clusters with only one conference. Cluster 6 has SBSC (Brazilian Symposium on Collaborative Systems). Cluster 10 has WIM (Workshop on Medical Informatics). By looking at the force graph, one could notice that these conferences are positioned almost outside of the graph. This indicate that there are no conferences in their related areas. Therefore, it makes sense that they are in their own clusters alone.

Cluster 7 contains conferences of programming languages (SBLP), formal methods (SBMF) and education (WEI). By looking at the force graph, it is possible to notice that SBLP and SBMF might be related conferences. WEI, by being of the education area, might be a conference with topics in several areas.

Cluster 8 has operating systems (WSO), SEMINCO (Computing Seminar) and education (SBIE) conferences. This cluster is very fuzzy if one observe the position of the conferences in the graph. This happens because they are very generic conferences, with topics that spread over several areas.

Cluster 9 consists of the IHC (Symposium of Human Factors in Computer Systems) and the Semish (Integrated Seminar of Software and Hardware) conferences. This cluster does not make sense since it combines conferences fo two complete different areas. But by looking at the force graph it is difficult to decide in which cluster these conferences should be.

Cluster 11 identified conferences of bio-informatics (BSB) and neural networks (SBRN). By analyzing the force graph, it is possible to note that these conferences are weakly connected to others. With a greater number of K clusters, they would probably have their own clusters.

5.3 Evolution

In this section, we have taken pairs of conferences on several years and evaluated the evolution of similarity. In this case, we used cosine similarity considering the TPC interest in topics as weighting. The Figure 5.3 shows the evolution of 5 pairs of conferences. These are the pairs that presented the biggest variance. The x-axis contains years in which the editions of the conferences occur and the y-axis contains the correspondent similarity value of the pair of conferences.

By analyzing this chart, one could notice that SBES became more similar with SBQS and with WMSWM in recent years. That is due to their call for paper that

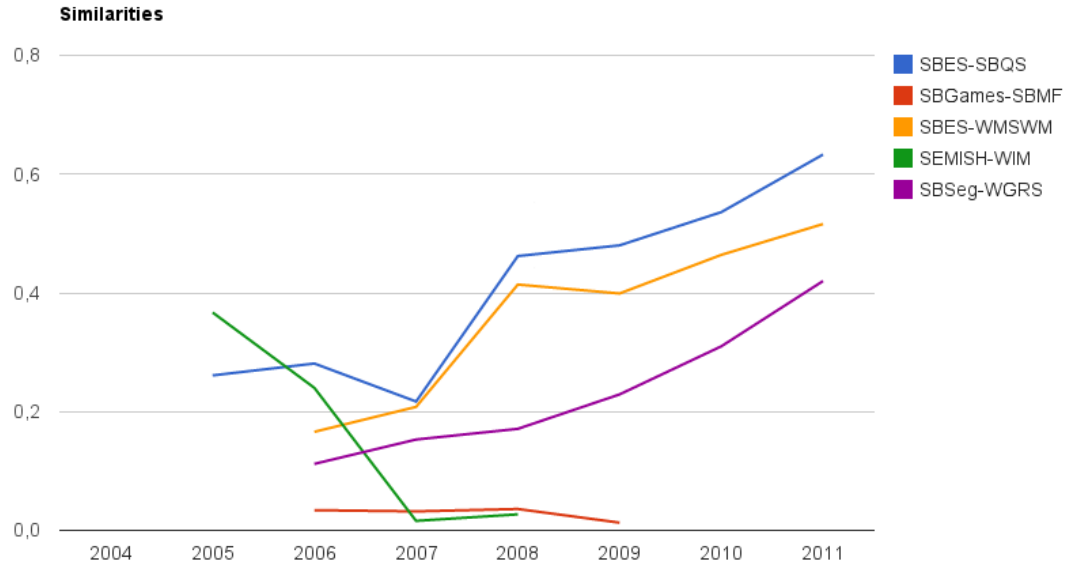

Figure 5.3: Similarities chart using cosine similarity and TPC interest

became more mature, less broader and more focused in its area.

SEMISH have distanced from WEI. SEMISH is a conference with broad topics with a call for papers that changed every year.

SBSeg and WGRS (Workshop on Management and Operation of Networks and Services) also became closer. One could infer that the WGRS have included topics in the security area.

In this chart it is also shown SBgames and SBMF. But by looking closely, they have a similarity degree that only variate between 0.034 and 0.013. This means that they are very different and it is not appropriate to do an analysis in their similarity evolution.

6 CONCLUSION

In this work we present a solution to measure and to support analysis of the similarity between conferences of research communities. Our solution is able to:

- Represent conferences in a format that allows the application of information retrieval techniques. This representation emphasizes the topics list of the conference, and considers relevant the: occurrences of term in the topics lists, TPC interest in topics, papers submitted and accepted in specific topics;
- Measure the degree of similarity between conferences. We used three metrics: Euclidean distance, cosine similarity and Jaccard index;
- Find groups (clusters) of conferences based on their similarity. This was achieved using the K-means algorithm;
- Visualize and export the similarity between pairs of conferences.

This solution was implemented as a web-based tool. This tool was designed using the MVC pattern and the CakePHP framework. The tool retrieves data from the JEMS conference database and performs the measurements proposed in the solution. Also, since it is implemented for the web, it allows great portability and ease of use, as anyone with a browser may access it.

To evaluate the proposed solution, we measured similarities of some Brazilian conferences on computer science in a case study. It was possible to verify that most clusters identified conferences of the same area. It was also shown that the similarity between conferences vary over years, we observed that some conferences became more similar and others less similar for a given time window.

For future work it would be interesting to compare and analyze the performance of other similarity metrics and clustering techniques. Examples of other similarity metrics that could be considered are the Pearson Correlation Coefficient and Averaged Kullback-Leibler Divergence [13]. Also, it would be interesting to use other initialization method for K-Means or using other clustering techniques. To evaluate these techniques, one could label all conferences into categories, run the clustering algorithm, and verify how the generated clusters correspond to the labeled categories using measures such as precision and recall[17].

Our current PHP implementation allows for great portability of the tool. However, we believe that a scripting language (mainly like PHP) would not be the better option for managing larges amounts of data to perform information retrieval (similarities measurements) and machine learning (clustering) techniques. We noticed

mainly when running K-means algorithm that the time needed is high for a web request (around 5 seconds).

Finally, we believe other visualizations could be researched. One problem we observed with force graphs is that it does not allow to visualize evolution of similarities. It is possible to dynamically change the links weights. But that would drastically change the nodes position in the graph, making it difficult to visualize evolutions over a time window. One possibility could be the animation of force graphs using a movie that shows the force changes over the year.

APPENDIX A CONFERENCES NAMES

Acronym	Conference Name
BSB	Brazilian Symposium on Bioinformatics
ERRC	Regional School of Computer Networks
IHC	Symposium of Human Factors in Computer Systems
SBAC-PAD	International Symposium on Computer Architecture and High Performance Computing
SBBD	Brazilian Symposium on Database
SBCUP	Brazilian Symposium on Pervasive and Ubiquitous Computing
SBES	Brazilian Symposium of Software
SBGames	Brazilian Symposium on Games and Digital Entertainment
SBIE	Brazilian Symposium on Computer in Education
SBLP	Brazilian Symposium on Programming Languages
SBMF	Brazilian Symposium on Formal Methods
SBQS	Brazilian Symposium of Software Quality
SBRC	Brazilian Symposium on Computer Networks and Distributed Systems
SBRN	Brazilian Symposium on Artificial Neural Networks
SBSC	Brazilian Symposium of Collaborative Systems
SBSeg	Brazilian Symposium on Information Security and Computer Systems
SEMINCO	Computing Seminar
SEMISH	Seminar on Hardware and Software
SIBGRAPI	Conference on Graphics, Patterns and Images
SVR	Symposium on Virtual and Augmented Reality
WebMedia	Brazilian Symposium on Multimedia Systems
WEI	Workshop on Computer Education
WGRS	Workshop on Management and Operation of Networks and Services
WIM	Workshop on Medical Informatics
WMSWM	Workshop on Modern Software Maintenance
WPerformance	Workshop on Performance of Computer and Communication
WSCAD	Symposium on Computing Systems
WSO	Workshop on Operating Systems
WTF	Workshop on Testing and Fault Tolerance
WTR	Workshop on Real-Time Systems

Table A.1: Conferences acronyms and names

REFERENCES

[1] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54. ACM, 2006.

[2] A.L. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3):590–614, 2002.

[3] A.L.C. Bazzan and VF Argenta. Network of collaboration among pc members of brazilian computer science conferences. *Journal of the Brazilian Computer Society*, 17(2):133–139, 2011.

[4] C. Bird, E. Barr, A. Nash, P. Devanbu, V. Filkov, and Z. Su. Structure and dynamics of research collaboration in computer science. In *SDM*, pages 826–827. Citeseer, 2009.

[5] Michael Bostock. Data-driven documents, November 2012. `http://d3js.org/`.

[6] Inc. Cake Software Foundation. Cakephp framework, November 2012. `http://www.cakephp.org/`.

[7] Oracle Corporation. Mysql database, November 2012. `https://www.mysql.com/`.

[8] C. Cotta and J.J. Merelo. The complex network of ec authors. *ACM SIGEVOlution*, 1(2):2–9, 2006.

[9] E. Elmacioglu and D. Lee. On six degrees of separation in dblp-db and more. *ACM SIGMOD Record*, 34(2):33–40, 2005.

[10] M. Fowler. *Patterns of enterprise application architecture*. Addison-Wesley Professional, 2003.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design patterns: Elements of reusable object-oriented design, 1995.

[12] Google. Gogle chart tools, November 2012. `https://developers.google.com/chart/`.

[13] A. Huang. Similarity measures for text document clustering. In *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

[14] M.E.J. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[15] J.M. Pena, J.A. Lozano, and P. Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, 20(10):1027–1040, 1999.

[16] M.F. Porter et al. An algorithm for suffix stripping, 1980.

[17] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[18] A. Strehl, J. Ghosh, R. Mooney, et al. Impact of similarity measures on webpage clustering. In *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64, 2000.

[19] R.A.B. Yates and B.R. Neto. Modern information retrieval. 1999.