SBBD 2015

XXX SIMPÓSIO BRASILEIRO DE BANCO DE DADOS

13 A 16 DE OUTUBRO DE 2015

PETRÓPOLIS

LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA - LNCC

PROCEEDINGS OF THE 30TH BRAZILIAN SYMPOSIUM ON DATABASES

VANESSA BRAGANHOLO, FABIO PORTO, EDUARDO OGASAWARA, JAVAM C. MACHADO (ORG.).

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

Realização

Laboratório Nacional de Computação Científica

CEFET/RJ

SBBD 2015

XXX SIMPÓSIO BRASILEIRO DE BANCO DE DADOS

30th BRAZILIAN SYMPOSIUM ON DATABASES

October 13th to 16th, 2015
Petrópolis – RJ – Brazil

PROCEEDINGS

Promotion
Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization
National Laboratory for Scientific Computing - LNCC
Federal Center of Technological Education of Rio de Janeiro - CEFET/RJ

Program Committee Chair
Vanessa Braganholo (UFF)

Steering Committee Chair
Mirella M. Moro (UFMG)

Local Organization Chairs
Fabio Porto (LNCC) and Eduardo Ogasawara (CEFET/RJ)

ISSN: 2316-5170

Brazilian Symposium on Databases (30. 2015, out. 13-16 : Petrópolis, RJ)
B823p Proceedings of the 30th Brazilian Symposium on Databases [recurso eletrônico] / Vanessa Braganholo [et al.] (org.). Petrópolis, RJ. : Laboratório Nacional de Computação Científica, 2015.
xvi, 164p. : il.

ISBN: 978-85-9996-120-9

1. Banco de dados - Encontros 2. Gerência de informações I. Brazilian Symposium on Databases II. Braganholo, Vanessa. (Org.) III. Título.

CDD – 005.74

Editorial

The Brazilian Symposium on Databases is the official database event of the Brazilian Computer Society (SBC) and the largest venue in Latin America for presentation and discussion of research results in the database domain. The 30th edition of the symposium (SBBD 2015) was held in Petrópolis city, also known as *The Imperial City of Brazil*, in the state of Rio de Janeiro, from the 13^{th} to 16^{th} of October 2015.

The SBBD 2015 program offers a variety of activities, suited for an audience ranging from undergraduate to Ph.D. students, database professionals, practitioners and researchers. The program includes: technical sessions, short courses, tutorials, invited talks, demo sessions, thesis and dissertation workshop, and PhD and Master contest. Papers presented at technical sessions have been submitted either to the Journal of Information and Data Management (JIDM) track or to the SBBD Proceedings track. JIDM articles present interesting results or novel thought-provoking ideas, and they are published in the October 2015 issue of the journal. SBBD Proceedings papers, on the other hand, present initial results or on-going work. After a thorough review process, a total of 2 high quality JIDM articles, and 11 SBBD Proceeding papers have been selected. Additionally, 18 posters papers have been selected.

Given the current buzz around big data and NOSQL, it is no surprise that the three short courses in SBBD 2015 were related to the management of large volumes of data: (i) "Processamento de Grafos em Big Data ", by Regis Pires Magalhães, Luís Gustavo C. do Rêgo, José Antônio F. de Macêdo, and Vânia Maria Ponte Vidal (UFC); (ii) "Publicação e Consumo de Dados na Web: Conceitos e Desafios ", by Bernadette Farias Lóscio and Ig Ibert Bittencourt (UFPE); and (iii) "Tecnologias para Gerenciamento de Dados na Era do Big Data", by Victor Teixeira de Almeida, Vitor A. Batista (Petrobras).

SBBD 2015 offers, in addition, two tutorials, both presented by experts on each topic. The first, with Brazilian authors, is "Reprodução de Experimentos Científicos: Teoria e Prática", by Ary H. M. Oliveira, Daniel de Oliveira, Marta Mattoso (COPPE-UFRJ). The international tutorial, entitled "Database Kernels Tailored for Data Exploration", is presented by Stratos Idreos (Harvard University).

We feel honored to have three distinguished invited speakers in the 2015 edition of SBBD: (i) Mike Carey, from University of California, Irvine, who is giving a talk entitled "AsterixDB: A Counter but Intuitive Approach to Big Data Management"; (ii) Pablo Barceló, from Universidade do Chile, talking about "Query Languages for Graph Databases: Bridging the Gap Between Theory and Practice"; and (iii) Ana Maria de Carvalho Moura, from National Laboratory of Scientific Computing (LNCC), with a talk entitled "Towards a Web of Semantically Integrated Data".

Finally, SBBD 2015 includes the presentation of 11 papers in the Demo session, 10 papers in the Thesis and Dissertation Workshop, and 7 papers in the PhD and

Master contest.

SBBD 2015 is the result of the collective effort of a large community, which we gratefully acknowledge. We thank the local organization committee and the symposium chairs who worked hard to guarantee an outstanding symposium. We are also grateful to the Steering Committee members for their help, advice and support. Further, we thank the program committee members and external reviewers who provided high quality reviews for the submitted papers, and the authors who submitted their papers to SBBD 2015. Finally, we are grateful to our sponsors. Without their support we would not be able to organize this annual event that brings together our community.

We hope you all enjoy SBBD 2015 in Petrópolis.

Fabio Porto, LNCC
SBBD 2015 Local Organization Committee Chair

Eduardo Ogasawara, CEFET/RJ
SBBD 2015 Local Organization Committee Chair

Table of Contents

30th Brazilian Symposium on Databases

full

October 13-16, 2015
Petrópolis – RJ – Brazil

FULL PAPERS

PROCEEDINGS

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

National Laboratory of Scientific Computing – LNCC
Federal Center of Technological Education of Rio de Janeiro – CEFET/RJ

SBBD Program Committee Chairs

Vanessa Braganholo, UFF

Editorial

It is a great pleasure to introduce the Proceedings of the Brazilian Symposium on Databases (SBBD) with the full papers accepted for presentation at the 30th edition of the symposium, held in Itaipava, Petrópolis, in the state of Rio de Janeiro, Brazil, from October 13th to 16th, 2015. This year, SBBD was organized by the National Laboratory of Scientific Computing (LNCC) and Federal Center of Technological Education of Rio de Janeiro (CEFET/RJ), both located in the state of Rio de Janeiro. SBBD is the official database event of the Brazilian Computer Society (SBC) and the largest venue in Latin America for presentation and discussion of research results in the databases domain. Along with technical sessions, SBBD includes invited talks, tutorials and short courses given by distinguished speakers from the national and international research communities. SBBD regularly promotes a Demos and Applications Session, and a Thesis and Dissertations Workshop as co-located events. This year, for the first time, SBBD is also promoting a Thesis and Dissertation Contest, where the best Brazilian thesis and dissertations in the database area, defended in 2013 and 2014, will be presented and run for the award. All papers submitted to SBBD present interesting results or novel thought-provoking ideas in several subjects on the databases and related areas. For the 2015 edition, SBBD has accepted three kinds of submissions: JIDM articles, full papers, and poster papers. Submissions to the JIDM Articles category work in a continuous flow throughout the year. The review is conducted by the editorial board of JIDM, leaded by the editor-in-chief Caetano Traina Jr. Articles accepted by August 15th were invited to present at SBBD 2015. This year, two JIDM articles will be presented in SBBD.

The review process for the SBBD full papers was performed in one round with a rebuttal phase. All authors were initially notified with the reviews and then had one week for answering the reviewers' comments during the rebuttal phase. After evaluating the rebuttal comments, a final decision was achieved. Out of 48 submitted papers, eleven papers were accepted as full papers (acceptance rate of 23%), and eight as poster papers. All accepted papers are published in these proceedings and were invited for oral presentation at SBBD 2015. The best full papers will be invited to submit an extended version of the paper to JIDM.

The topics with more full paper submissions (according to the author's selection from the Topics of Interest) were: Data Classification (thirteen submissions), Machine Learning Applications (eleven submissions), and Applications of Data Mining (ten submissions). Likewise, the topics with more accepted submissions were Applications of Data Mining, with four accepted papers.

The Proceedings of SBBD and its respective JIDM issue are the result of the collective effort of a large community, which we gratefully acknowledge. We thank SBBD 2015 local organization committee and its symposium chairs, who worked

hard to guarantee an outstanding Symposium. We are also grateful to the Steering Committee members for their help, advice and support. We also thank the Program Committee members and External Reviewers who provided high quality reviews for the submissions, working very hard in all rounds of review. Finally, we are grateful to the authors who submitted their work to JIDM and SBBD.

Vanessa Braganholo (UFF)
SBBD 2015 Program Committee Chair

30th Brazilian Symposium on Databases

October 13-16, 2015
Petrópolis – RJ – Brazil

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

National Laboratory of Scientific Computing – LNCC
Federal Center of Technological Education of Rio de Janeiro – CEFET/RJ

SBBD Steering Commitee

Altigran Soares da Silva (UFAM)
Caetano Traina Jr. (ICMC - USP)
Cristina Ciferri (ICMC-USP)
Javam Machado (UFC)
Marco A. Casanova (PUC-Rio)
Mirella M. Moro (UFMG, chair)
Vanessa Braganholo (UFF)

SBBD 2015 Commitee

Program Committee Chair
Vanessa Braganholo (UFF)

Local Organization Chairs
Fabio Porto (LNCC) and Eduardo Ogasawara (CEFET/RJ)

Steering Committee Chair
Mirella M. Moro (UFMG)

Demos and Applications Session Chair
Valéria C. Times (UFPE)

Workshop on Thesis and Dissertations in Databases Chair
Ricardo Torres (UNICAMP)

Lectures Chair
Carmem S. Hara (UFPR)

Tutorials Chair
Javam Machado (UFC)

Contest on Thesis and Dissertations Chairs
José Palazzo M. de Oliveira (UFRGS)

Local Organization Committee

Adolfo Simões (LNCC)
Daniel S. Kaster, UEL (consultant)
Eduardo Ogasawara (CEFET-RJ, chair)
Fernando Ferreira dos Santos (consultant)
Fábio Porto (LNCC, chair)
Paulo Pires (UFRJ, finances)

SBBD Program Committee

Vanessa Braganholo (UFF, Brazil, PC Chair)
Agma J. Traina (USP São Carlos, Brazil)
Alexandre Plastino (UFF, Brazil)
Altigran Silva (UFAM, Brazil)
Amr El Abbadi (UCSB, USA)
Ana Carolina Salgado (UFPE, Brazil)
André C. P. L. F. Carvalho (USP, Brazil)
André Santanchè (UNICAMP, Brazil)
Angelo Brayner (UNIFOR, Brazil)
Bernadette Lóscio (UFPE, Brazil)
Caetano Traina Jr (USP, Brazil)
Carina Dorneles (UFSC, Brazil)
Carmen Hara (UFPR, Brazil)
Celso Massaki Hirata (ITA, Brazil)
Cláudio de Souza Baptista (UFCG, Brazil)
Clodoveu Davis (UFMG, Brazil)
Cristina Ciferri (USP São Carlos, Brazil)
Daniel Kaster (UEL, Brazil)
Daniel Oliveira (UFF, Brazil)
Edleno de Moura (UFAM, Brazil)
Eduardo Ogasawara (CEFET-RJ, Brazil)
Elaine P. M. de Sousa (USP, Brazil)
Eli Cortez (Microsoft Research, USA)
Fabio Porto (LNCC, Brazil)
Fernanda Baião (UNIRIO, Brazil)

Javam Machado (UFC, Brazil)
João Eduardo Ferreira (USP, Brazil)
Jorge Perez (University of Chile, Chile)
José Palazzo Moreira de Oliveira (UFRGS, Brazil)
José Viterbo Filho (UFF, Brazil)
Leo Bertossi (Carleton University, Canada)
Lipyeow Lim (University of Hawaii, USA)
Luciano Barbosa (IBM Research, Brazil)
Marcelo Arenas (PUC, Chile)
Marco Antonio Casanova (PUC-Rio, Brazil)
Marcos André Gonçalves (UFMG, Brazil)
Marta Mattoso (COPPE-UFRJ, Brazil)
Mirella Moro (UFMG, Brazil)
Pablo Barceló (University of Chile, Chile)
Renata Galante (UFRGS, Brazil)
Ricardo Torres (UNICAMP, Brazil)
Ronaldo dos Santos Mello (UFSC, Brazil)
Sandra de Amo (UFU, Brazil)
Theo Haerder (Un. of Kaiserslautern, Germany)
Valéria Times (UFPE, Brazil)
Victor Almeida (Petrobrás, Brazil)
Viviane P. Moreira (UFRGS, Brazil)
Wagner Meira (UFMG, Brazil)

External Reviewers

André Luis Schwerz (UTFPR)
Crícia Z. Felício (IFTM)
Damires Souza (IFPB)
Eduardo Corrêa Gonçalves (UFF)
Fabiola Pereira (UFU)
Humberto Luiz Razente (UFU)
João Carlos de A. R. Gonçalves (UNIRIO)
José De Aguiar Moraes Filho (UNIFOR)
José Antonio Fernandes de Macedo (UFC)
José Maria da Silva Monteiro Filho (UFC)
Luciano V. Araújo (EACH-USP)
Lucinéia Heloisa Thom (UFRGS)
Márcio K. Oikawa (UFABC)
Michele A. Brandão (UFMG)
Rafael Gomes Mantovani (ICMC-USP)
Rafael Liberato (UTFPR)
Rafael Pereira (UFF)
Raqueline Ritter de Moura Penteado (UEM)
Renan Sousa (COPPE-UFRJ)

Theodore Georgiou (UC Santa Barbara)
Vania Vidal (UFC)
Vaibhav Arora (UC Santa Barbara)
Vitor Silva (COPPE-UFRJ)

Table of Contents (JIDM Articles)

jidm

Table of Contents (SBBD Proceedings)

sbbd

An Architecture for Monitoring and Recommending Active Database DDLs

Paulo H. dos Santos[1], **Nádia P. Kozievitch**[1], **Roberto C. Betini**[1]

[1]Departamento de Informática, Universidade Federal Tecnológica do Paraná, Curitiba, PR, Brazil

phscuritiba@gmail.com, nadiap@utfpr.edu.br, betini@utfpr.edu.br

Abstract. *The integrated monitoring within heterogeneous database environments can become complex, due to the particularities in the language syntax and available tools. In particular, active databases allow developing mechanisms and automation of processes involving data or objects. This work proposes the development of an architecture to identify and monitor DDL, exploring the active databases and recommendations approach. This architecture could be explored in several ways: simple monitoring of DDL events in one or more databases, or recommendation of future DDLs or settings within the monitored database. In this context, this paper proposes: (i) an integrated architecture of active databases and recommendation; (ii) the adaptation of a recommendation algorithm, and (iii) the validation of concepts through a prototype.*

1. Introduction

Many researches and practical work related to Database Management Systems (DBMSs) have emerged, in particular the monitoring of changes within databases. The development of mechanisms for process the automation and handling of this information is also extensively explored. However, monitoring the changes within the object structures (tables, indexes, etc.) is somewhat overlooked, due to the complexity within the integration and treatment of the available DBMSs [Jun 2010].

In order to develop mechanisms that allow monitoring of such changes dynamically, Active Databases (AD) have been explored. The basic idea relies on the combination of DBMS technology programming techniques with rule-based and event-driven [Firoozy-Najafabadi and Navin 2012]. Examples include executive information systems (maintaining derived data), cancer clustering (knowledge discovery in databases) and software process management (process management and control) [Appelrath et al. 1996].

This type of system supports the definition of rules, along with system monitoring and execution of actions in response to defined rules. Basically, rules are used to monitor and respond to events that happen inside the database. Rules can reduce the amount of code that is required in applications, and can also prevent redundant code that would otherwise be required when multiple programs perform the same operations on the database. A trigger is one type of rules in an active database system which is used to detect and then react to some conditions.

For monitoring events related to object changes from the database (Data Definition Language - DDL events), the level of complexity increases, since different DBMSs

have specific characteristics within syntax. Another important and under explored context relies on the integration of AD with Recommendation Systems (RS). Here the objective is suggest changes or settings to the user, based on the history of the available system.

This paper proposes a conceptual architecture which includes: i) the development of mechanisms which enable the creation, execution, and storage of active rules; ii) the monitoring of changes made by DDL events; iii) the recommendation of possible configurations or changes in the structure (objects) of the monitored database; and iv) a preliminary evaluation. This document has the following organization: section 2 presents the related work; section 3 shows the conceptual architecture; section 4 presents the experimental evaluation; and finally section 5 presents conclusions and a discussion of future work.

2. Related Work

2.1. Active Databases

Active databases support the preceding application by moving the reactive behavior from the application (or polling mechanism) into the DBMS. Active databases are thus able to monitor and react to specific circumstances of relevance to an application. The reactive semantics is both centralized and handled in a timely manner [Paton and Díaz 1999]. An active database system must provide a knowledge model (i.e., a description mechanism) and an execution model (i.e., a runtime strategy) for supporting this reactive behavior. Nevertheless, ADs still provide a paradigm for new research approaches from combining Artificial Intelligence and DBMS technologies [Morgenstern 1983].

The reactive behavior resides on rules which integrates cause with an expected effect [Montesi and Torlone 1995]. These rules (mainly **active rules**), allow the monitoring and reaction to specific events. Such rules allow to express in a natural way, important database concepts, such as scenarios, constraints and methods. An active rule is a generalization of this schema language [Cardoso 2004]. Active rules applications support conventional DBMSs within different contexts, such as relational integrity, data maintenance, replication, monitoring data updates, etc. [Zaniolo et al. 1997].

Active rules are composed of three main components: an Event, a Condition and an Action (known as ECA rule). The basic semantics of an ECA rule can be summarized in the following sentence: when an event of type E occurs, if the condition C is true, then the action A is executed (as shown in Figure 1). The knowledge model for active rule sets defines the behavior of a set of rules at runtime [Paton and Díaz 1999], so the system monitors and reacts to certain circumstances. Within Figure 1, the symbol $\subset$ is used to indicate that the particular dimension can take on more than one of the values given, whereas $\in$ indicates a list of alternatives. The ECA knowledge model represented by [Paton and Díaz 1999] is based around a number of dimensions of active behavior, making explicit the decision space within which the designers of active rule system works.

When an DDL event (drop, alter, create) occur, and any changes are made to the database, the execution condition of the rule is verified, and if it matches the conditions set, the action is performed by recording the occurrence in a given location. Figure 2 illustrates the major steps used in processing an active set of rules: 1) the detection of the event; 2) the ”Triggered Rules” box rule is triggered according to the detected event;

Event	**Source** ⊂ {Structure Operation, Behavior Invocation, Transaction, Abstract, Exception, Clock, External} **Granularity** ⊂ {Member, Subset, Set} **Type** ⊂ {Primitive, Composite } **Operators** ⊂ {or, and, seq, closure, times, not } **Consumption mode** ⊂ {Recent, Chronicle, Cumulative, Continuous } **Role** ∈ {Mandatory, Optional, None}
Condition	**Role** ∈ {Mandatory, Optional, None} **Context** ⊂ {DB_T, $Bind_E$, DB_E, DB_C}
Action	**Options** ⊂ {Structure Operation, Behavior Invocation, Update-Rules, Abort Inform, External, Do Instead } **Context** ⊂ {DB_T, $Bind_E$, $Bind_C$, DB_E, DB_C, DB_A}

Figure 1. Active Rule components.

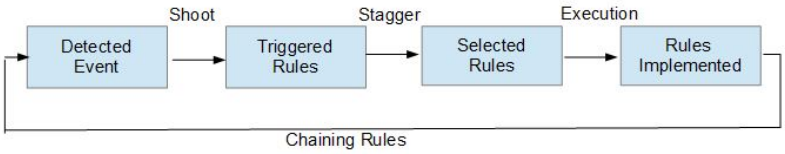


Figure 2. Steps of Event handling [Paton and Díaz 1999].

Table 1. Comparison among the available related work and this proposal: 1) [Badia 2003]; 2) [Qiao et al. 2007]; 3) [Lu and Miklau 2008]; 4) [Simon et al. 2008]; 5) [Huang et al. 2009]; 6) [Jin 2009]; 7) [Medina-Marin et al. 2009]; 8) [Jun 2010].

Papers / Characteristics	1	2	3	4	5	6	7	8	Proposal
Active rules implemented.	+	+	+	+	+	+	+	+	+
Storage of records changes.	-	-	+	+	-	-	-	-	+
Platform independent.	+	-	+	-	+	+	+	+	+
Based on events.	+	+	+	+	+	+	+	+	+
Monitoring changing data.	+	+	+	+	-	+	+	+	-
Monitoring change in the structure, events DDL (database objects).	-	-	-	-	-	-	-	-	+

3) in the ”Selected Rules” box the condition is evaluated; and finally, 4) if the condition evaluated is true, the rule is executed. Note that in Figure 2 there is a loop for selection rules - this occurs when there are more rules to be executed depending on the type of event detected.

Considering several proposals (such as [Badia 2003, Qiao et al. 2007, Lu and Miklau 2008, Simon et al. 2008, Huang et al. 2009, Jin 2009, Medina-Marin et al. 2009, Jun 2010]), Table 1 presents a comparative analysis. The first column lists the characteristics, the identifier ”+” indicates that the work covers such a feature, and ”-” in case it does not cover.

2.2. Recommendation Systems

Modern systems, especially those available at Internet, use mechanisms to explore the preferences of its users. Others are based largely on the recommendations of these sys-

tems when making decisions. These systems are examples of Recommendation Systems (RS) [Cazella et al. 2010].

RSs often can be considered as a particular type of personalization, learning about the needs of an individual or of a community and then proactively identifies and recommends information that meets these needs [Smeaton and Callan 2005]. These systems are based on personalization of information. This customization is related to the way in which information and services can be tailored to the specific needs of a user or a community. So, one can say that RSs are used to help users identify products or services of interest to them, within a lot of options.

A recommendation is a mapping function of interest of the customer to obtain one or more products [Pimentel and Fuks 2012]. A recommendation R for the choice "p" can be defined as:

$$R(c, p) = \max F(c, pi)$$

where:

- c represents the customer which uses the RS;
- p a set of products available to evaluation;
- $pi \in p$;
- F is a function which verifies the relevance of pi considering c.

According to [Pimentel and Fuks 2012], the RS process represented as the definition above evaluates the usefulness of a product for a particular customer. Generally, function F takes into account the similarity between the customer profiles.

3. The Conceptual Architecture

The objective of this paper is to explore the applicability of the automatic monitoring DDL events for active databases and, therefore, develop an architecture by integrating the necessary mechanisms to provide such automation. Figure 3 presents the proposed architecture. The database layer can comprise different technologies, and their DDL can be monitored through the applied rules (active rules), transformed into triggers. The triggers will represent the main mechanisms for identifying and registering how, when and who made the change or creation of an object in the database structure monitored.

The centralized layer includes AD repositories and mechanisms for recommendation generation and application of rules. The change repository stores information related to changes within the monitored database. The recommen-dation parameters repository stores information related to the parameters requested by the user when the recommen-dation is requested. The rules repository stores the active rules registered to apply to the monitored database. The syntax and semantics repository is used by the rules module, storing appropriate syntax and semantics for rules in different DBMS. The rule enforcement mechanism is responsible for applying the rules in the monitored databases. And finally, the mechanism of recomme-ndation generation is responsible for generating recommendations for a database and a chosen server.

The interface modules are set to interact with features such as: registration and application of rules; querying and recommendations; change records of view; and syntax and semantics of inclusion.

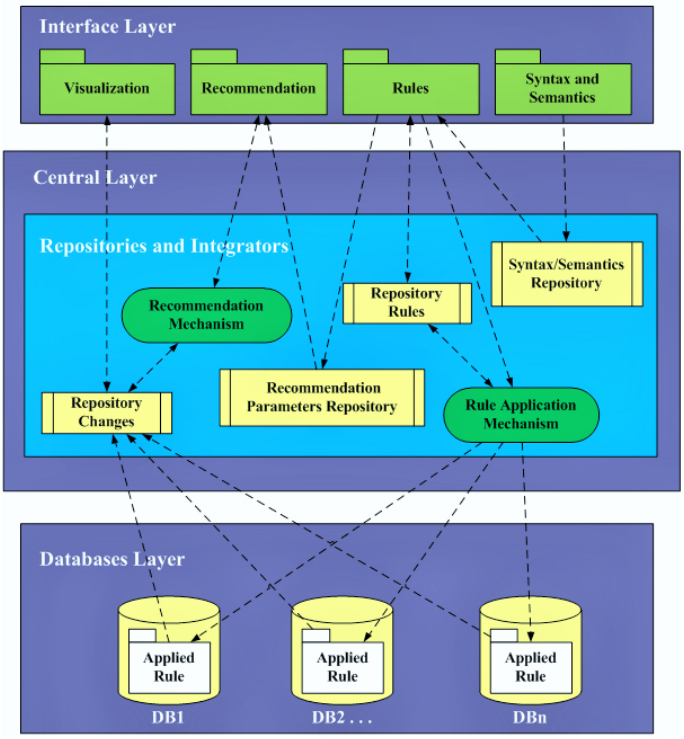


Figure 3. The conceptual architecture.

In this work we use the tuple-based recommendation approach proposed by [Eirinaki et al. 2013]. The original algorithm is part of a framework which generates recommendation queries from the current session of a user, combining the current session query with a predictive model generated from queries made by other users in past sessions. The proposal has been updated within two directions: (i) instead of data, the algorithm was adapted to work with DDL, such as tables creation; and (ii) the session summaries have been updated to each user, server and instances.

The framework consists of three components: (a) a model of session summaries, (b) a method to compute the session summaries $S_0, ..., S_n$, the construction of a session summary for each user *i* based on the **DDL commands** in Qi, (c) the computation of S^{pred}, and (d) a method to recommend queries based on S^{pred}, resulting as top ranked queries returned as the recommendation.

The first and second phase of the framework comprises the model for session summary along with the computation of the session summaries $S_0, ..., S_n$. The session summary S_i is represented as a weighted vector that corresponds to a DDL monitored. We assume that the total number of tables in the database, and as a consequence, the length of the vector is T. The weight $S_i\,[\tau]$ represents the importance of a given DDL $\tau \in T$ in session S_i, and is non zero if τ is a witness for at least one DDL in the session. The intuition is the same of the binary weighting scheme (from Eirinaki), capturing the tables in the base that are touched by the DDL in the user´s session. Hence, sessions that contain equivalent DDLs will map to the same summary. We assume that the vector S_Q represents a single DDL Q.

$$S_Q\lceil\tau\rceil = \begin{cases} 1 & if\ \tau \text{ is a witness;} \\ 0 & if \tau \text{ is not a witness.} \end{cases}$$

Given the vectors S_Q for each DDL Q posed by user i, the session summary S_i is defined as:

$$S_i = \sum_{Q \in Q_i} S_Q$$

The session summaries of past users are used to construct the $(n \times T)$ session-DDL matrix which will serve as input to the recommendation algorithm.

The third phase comprises the vector of DDL weights S^{pred}, where each weight signifies the predicted importance of the corresponding tuple for the active user. We assume the existence of a function $sim(S_i, S_j)$ that measures the similarity between two session summaries and takes value in [0,1]. The similarity function employed was the cosine similarity. In resume, two users are similar if their queries imply similar weights for the database DDLs. The S_0^{pred} summary is defined as a function of the current user´s summary S_0 and the normalized weighted sum of existing summaries, for each user, server and instance:

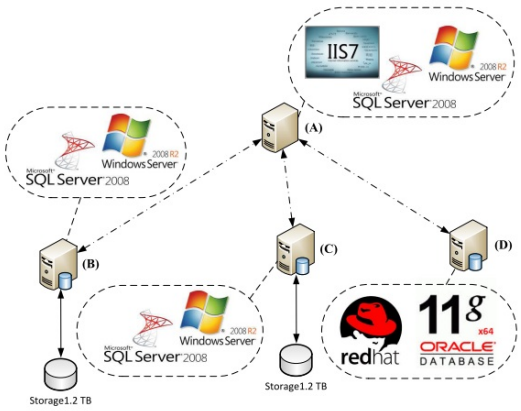
$$\mathrm{S}_0^{pred}(vuser, vserv, vinst) = \alpha * S_0 + (1-\alpha) * \frac{\sum_{1 \le i \le h} sim(S_0, S_i) \cdot S_i}{\sum_{1 \le i \le h} sim(S_0, S_i)}$$

The value of $\alpha \in [0, 1]$ determines which user´s traces will taken into consideration. If $\alpha = 0$, then it takes into account only the user´s traces. When $\alpha = 1$, only the active user´s session summary is taken into account when generation recommendations. Any value in between allow us to assign more "'importance" to either side. Within this paper, we assume $\alpha = 0,5$ to assign importance to both user´s traces. Overall, S^{pred} yields a weight per that corresponds to the importance of the DDL within the user´s exploration. Having computed S^{pred}, the algorithm indicates the top 5 ranked queries as the recommendation.

4. Experimental Evaluation

We concentrated our efforts within a tuple-based approach (as indicated by [Eirinaki et al. 2013]). The DDL commands are relaxed towards the operated table in order to increase the cardinality. The intuition is that if two users create the same table, using slightly different conditions (such as the column type as varchar for user A and integer for user B), the algorithm will consider them similar. Basically, the user creates a rule which is later derived to a specific semantic (to an oracle database, for example). This rule is translated to a trigger which is applied to a database, starting the monitoring phase. Later, the logs of monitored events are then used to create recommendations.

The rest of this section discusses the preliminary experimental results with a real database, along with examples of DDLs and generated reco-mmendations.



(A)

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
alter trigger [trg_audit_ddl] on create table
as  begin set nocount on
        declare @erro nvarchar(4000)
        declare @evento xml
        declare @instancia nvarchar(250)

        set @erro = (select error_message());
        set @evento = eventdata()
        set @instancia = convert(nvarchar(250), (select serverproperty('InstanceName')))

        if (@instancia is null) set @instancia = 'MSSQLSERVER'

        if ((is_srvrolemember('sysadmin',system_user) = 1) or
                        (system_user = 'lgFPw') or
                        (system_user = 'jira6dbuser') or
                        (system_user = 'GSFinancingUser'))
                        begin set ansi_padding on
                        set ansi_padding off end else begin
                        rollback
                        begin transaction BicBic
                        set ansi_padding on
                        insert  master.dbo.Alteracoes
                                        (dataalteracao,bdusuario,login,nomehost,nomeinstancia,
                                         bancodados,identddl,nomeEsquema,tipoobjeto,nomeobjeto,
                                        scriptalteracao,msgerro)
                        values  (getdate()
                                        ,convert(nvarchar(250), current_user)
                                        ,convert(nvarchar(250), system_user)
```

(B)

Figure 4. (A) The environment for the experimental evaluation and (B) the trigger for monitoring the create tables.

Table 2. Data Set Statistics.

Description	Server B	Server C	Server D
Database Size	5GB	5GB	2.9 GB
# Sessions	49	81	35
# DDLs	459	435	209
Avg # distinct DDLs	51	48	23
Min # distinct DDLs per session	9	5	6

4.1. Monitored Dataset and the Database for the Central Layer

Three different servers (B,C,D) (illustrated in Figure 4-A), with 3 months of historical data (Jun-Aug 2014) were monitored during the experiments by another server (A) (details are listed in Table 2). Server A presented 400GB of financial data, stored under Microsoft SQL Server 2008 R2. Server B had 1.2 TB storage, with 10 databases, comprising ERP, CRM and financial data, stored under Microsoft SQL Server 2008 R2. Server C had 1.2 TB, with 8 databases, comprising data from HR, Clients and company tickets, stored under Microsoft SQL Server 2008 R2. Server D had 1 database, comprising financial data, stored under Oracle 11G.

To store the complete information regarding the architecture and the central layer (located at server A), 6 groups of information are stored in the schema presented in Figure 5: the users which access the system, the rules, the rules applied to the SGBD instances, the semantics of the applied rule, the databases which are monitored, and the DDL changes (along with information such as user which executed the DDL, the schema used, the script for the update, etc.).

4.2. Hardware

The conceptual architecture was implemented as shown by Figure 4-A. Server A was used to implement the interface and the centralized layer. Servers B,C,D represent the monitored databases.

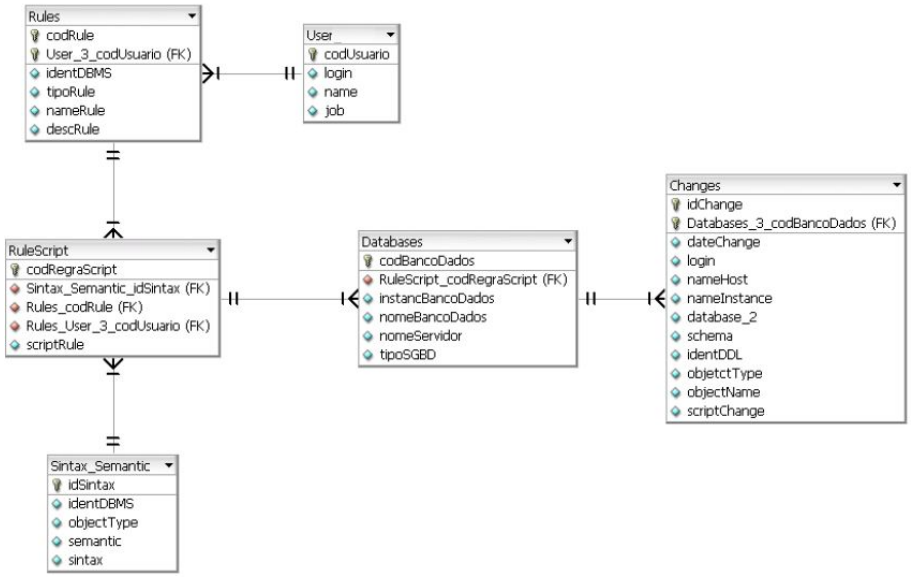


Figure 5. The database schema for storing the architecture information.

Server A (with the interface and centralized layer) was a HP ProLiant BL460c G7, with 2 processors Six-Core Intel Xeon with 2667 MHz, 16 GB of RAM, 600 GB of disk SAS, with 15 rpm in RAID 5, along with 4 network interfaces of 1 Gbps each one.

Server B was a HP ProLiant BL460c G6, with 2 processors Six-Core Intel Xeon with 2667 MHz, 72 GB of RAM, 1,2 TB of disk SAS, with 10 available at a IBM DS3512 storage, with 2 network interfaces of 1Gbps each one, along with 1 Fibre Channel interface of 8 Gbps connecting this server to the storage.

Server C was a HP ProLiant BL460c G7, with 2 processors Six-Core Intel Xeon of 2667 MHz, 96 GB of RAM, 1,2 TB of disks SAS with RAID 10 using storage IBM DS3512, with 2 network interfaces of 1 Gbps each one, and 1 interface Fibre Channel of 8 Gbps.

Server D was a HP ProLiant BL460c G6, with 1 processor Six-Core Intel Xeon of 2667 MHz, 8 GB of RAM, 600 GB of storage SAS with 15 rpm in RAID 5, and 2 network of 1 Gbps each one.

The interface used Microsoft .Net (Visual Studio 2012), located at Server A. The recommendation algorithm was implemented through a SQL procedure, and the cosine similarity was implemented by a SQL function (more details are available at [dos Santos 2014]).

4.3. Prototype Limitations

Toward a prototype implementation, we explored a simplified version of the architecture proposed within the previous section. Basically these limitations include:

- the available tests explored only the create table DDL: all the create DDL commands (from all users) from three servers (B,C,D) were monitored and logged within server A. The trigger representing the rule for monitoring the table creation is presented in Figure 4-B;

- the prototype was evaluated through two different SGBDs (SQL Server and Oracle);
- the prototype was tested only within preliminary phases;
- the available tests explored only one recommendation algorithm (DDL-Based Query Recommendation), and only one similarity function (cosine). Note that the architecture and modules could be easily extended to other recommendation algorithms and similarity functions;
- three months of historical data from a real company was monitored.

4.4. The Prototype

From the interface perspective, once the user accesses the system, four modules are available: the visualization of he monitored dataset logs, the search for recommendations, the creation of rules, and the verification of syntax and semantics.

To startup the monitored system, the user should register a rule through the interface, as shown in Figure 6-A, using information such as DBMS, event name, the semantic of the ECA events, and the body trigger for the rule (storing the information within the rules repository.

The second step includes the application of the rule to a specific server instance (Figure 6-B). This step includes the description of the server name, the database, the DBMS type and the trigger itself adapted to the database (and the respective information is stored at the syntax/semantics repository). Each instance will create a trigger which will monitor the respective database (through the rule application mechanism module).

The application of rule in Figure 6-A starts logging all the create table DDL (started by users or applications), along with the information such as the login of the user and the host, as shown in Figure 7-A. Note that within the same session, an user can create two or more tables at the same host, within the same date.

For each session, user, server and instance, a summary of accessed tables is created (phase 1), serving as input to the recommendation algorithm. Phase 2 comprises the same information is summarized by user, server and instance, as showed in Figure 7-B. Finally, phase 3 calculate S^{pred}. The 5 top ranked queries are returned as the recommendation. The interface for searching recommendations is presented within Figure 6-C.

Table 3. The top-5 ranked queries returned as recommendation.

Table	Weight
Tbl_C	1.03
Tbl_B	1.03
Tbl_D	1.00
Tbl_A	0.23

Two preliminary experiments were executed within the prototype. At the first experiment, a rule for create table was created. The objective was to monitor the production environment of a real enterprise environment, where several tables were created without DBA's permissions. Figure 8 shows the logs created by the rule. Note that two log entries (in red) were created by an application in another server (login and host names were

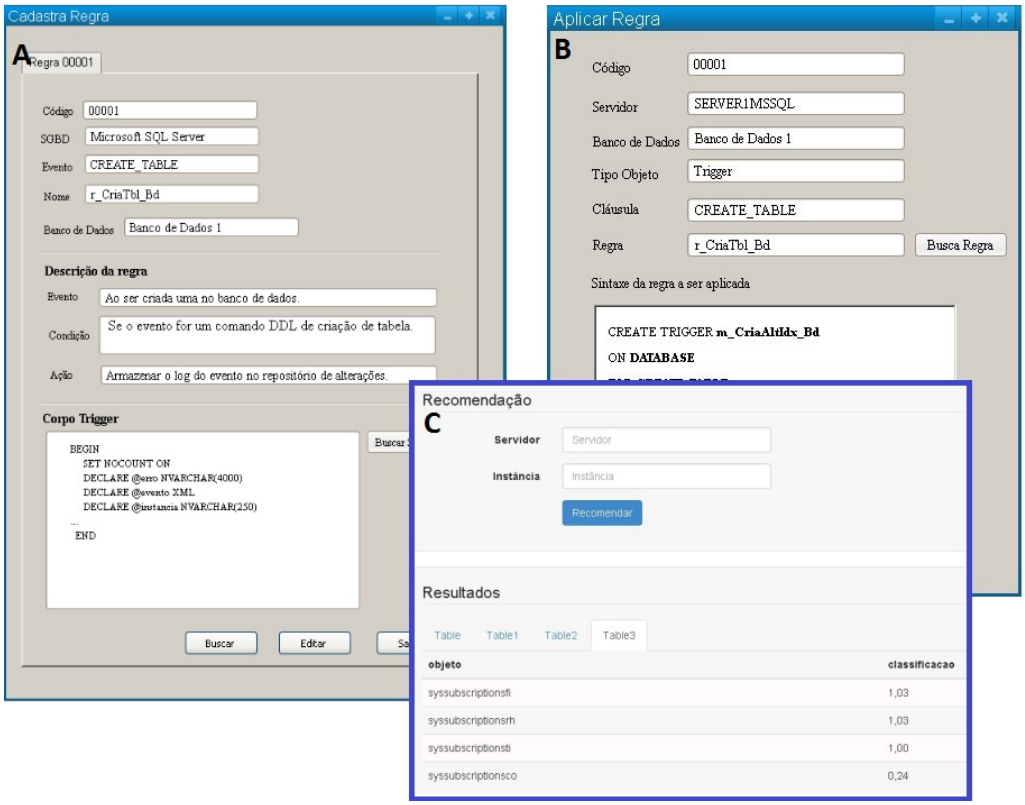


Figure 6. The interface for (A) creating the rule, (B) applying the rule to a server and (C) recommending a table creation.

Session	Date Change	User	Server	Instance	Database	Object Name
19	22/03/2014	DOMINIO\usuario2	SRVBDDV1	SRVBDDV1	BdSis1	Tbl_D
19	22/03/2014	DOMINIO\usuario2	SRVBDDV1	SRVBDDV1	BdSis1	Tbl_C
21	23/03/2014	DOMINIO\usuario3	SRVBDDV1	SRVBDDV1	BdSis1_T	Tbl_C
23	23/03/2014	DOMINIO\usuario1	SRVBDDV1	SRVBDDV1	BdSis1	Tbl_B
17	24/03/2014	DOMINIO\usuario1	SRVBDDV1	SRVBDDV1	BdSis1_H	Tbl_C
22	23/03/2014	DOMINIO\usuario3	SRVBDDV1	SRVBDDV1	BdSis1_H	Tbl_B
22	23/03/2014	DOMINIO\usuario3	SRVBDDV1	SRVBDDV1	BdSis1_H	Tbl_A
20	22/03/2014	DOMINIO\usuario2	SRVBDDV1	SRVBDDV1	BdSis1_T	Tbl_D
20	22/03/2014	DOMINIO\usuario2	SRVBDDV1	SRVBDDV1	BdSis1_T	Tbl_B

(A)

			Tables				
User	Server	Instance	Tbl_A	Tbl_B	Tbl_C	Tbl_D	···
DOMINIO\usuario3	SRVBDDV1	INSTSQL	1	1	1	0	···
DOMINIO\usuario2	SRVBDDV1	INSTSQL	0	1	1	2	···
DOMINIO\usuario1	SRVBDDV1	INSTSQL	0	1	1	0	···

(B)

Figure 7. (A) The log information and (B) the log summarization.

date	login	nameHost	identDDL	...
2014-02-20 17:15:00.823	Domínio\dba1	Desktop1	CREATE_TABLE	...
2014-02-13 04:00:01.677	Domínio\serviço	ServidorExt1	CREATE_TABLE	...
2014-02-13 04:00:01.570	Domínio\serviço	ServidorExt1	CREATE_ TABLE	...
2014-02-20 01:51:26.557	Domínio\ dba2	Desktop2	CREATE_ TABLE	...
...	...	...	...	...

Figure 8. Logs from the monitored environment.

replaced due to security restrictions). After monitoring the environment for one week, it was verified that the create table DDL triggered by an application in another server was actually created by developers, which discovered a security breach. After identified the event, the developer accesses were removed.

At the second preliminary experiment, three months of data (1103 records) were stored for analyzing the recommendation algorithm. The data between march/2014 and may/2014 resulted in 93 create table events. Exploring a specific user, instance and server, the sessions were summarized, as shown in Figure 7-B. the recommendation algorithm was processed, resulting in the recommendation (previously shown in Figure 3) of Table B and Table C as the most created at the server. Although being preliminary, the two tests states that the implementation of the architecture is viable, using a DDL-tuple based approach along with different databases (SQL server and Oracle).

5. Conclusions and Future Work

Monitoring the changes within the object structures (tables, indexes, etc.), is still overlooked, due to the complexity within the integration and treatment of the available DBMSs. This paper proposed a conceptual architecture which includes: i) the development of mechanisms which enable the creation, execution, and storage of active rules; ii) the monitoring of changes made by DDL events; and iii) the recommendation of possible configurations or changes in the structure (objects) of the monitored database. Finally an experimental prototype was presented, monitoring different databases for a three month period, using the development, testing and production environments of a real business. Preliminary tests concludes that the implementation of the architecture is viable, even using different databases along with distinct DBMS. Future work includes the precision/recall, performance and weighting scheme evaluation, the inclusion of other DDL events (such as alter table, drop table, etc.) along with the fragmented analysis of DDL commands (such as the same create table command which differs by the number of created columns). **Acknowledgments** We thank CNPq, CAPES, and Fundação Araucária for financial support.

References

Appelrath, H.-J., Behrends, H., Jasper, H., and Zukunft, O. (1996). Case studies on active database applications. In Wagner, R. and Thoma, H., editors, *Database and Expert Systems Applications*, volume 1134 of *Lecture Notes in Computer Science*, pages 69–78. Springer Berlin Heidelberg.

Badia, A. (2003). Active database systems for monitoring and surveillance. In Chen, H., Miranda, R., Zeng, D., Demchak, C., Schroeder, J., and Madhusudan, T., editors, *Intelligence and Security Informatics*, volume 2665 of *Lecture Notes in Computer Science*, pages 296–307. Springer Berlin Heidelberg.

Cardoso, V. M. (2004). Uma ferramenta para teste estrutural de regras ativas. Master's thesis, Mestrado em Engenharia Elétrica - Unicamp, São Paulo.

Cazella, S. C., Nunes, M., and Reategui, E. B. (2010). A ciência da opinião: Estado da arte em sistemas de recomendação. *XXX Congresso da SBC Jornada de Atualização da Informática, Porto Alegre, Brasil, 52p*.

dos Santos, P. H. (2014). Uma arquitetura para monitoramento de Banco de Dados e Recomendações utilizando Sistema de Banco de Dados Ativos. Master's thesis, Universidade Tecnológica Federal do Paraná, Curitiba - PR - Brasil.

Eirinaki, M., Abraham, S., Polyzotis, N., and Shaikh, N. (2013). Querie: Collaborative database exploration. *Knowledge and Data Engineering, IEEE Transactions on*, PP(99):1–14.

Firoozy-Najafabadi, H.-R. and Navin, A. (2012). Rule scheduling methods in active database systems: A brief survey. In *Application of Information and Communication Technologies (AICT), 2012 6th International Conference on*, pages 1–5.

Huang, K., Chen, P., Chen, Y., Song, W., and Chen, X. (2009). The research for embedded active database based on eca rule and implementation in sqlite database. In *Database Technology and Applications, 2009 First International Workshop on*, pages 476–479.

Jin, Y. (2009). Management of composite event for active database rule scheduling. In *IEEE International Conference on Information Reuse Integration (IRI'09), 2009*, pages 300–304.

Jun, C. (2010). Active web database technology and its applications in crm. In *International Conference on Environmental Science and Information Application Technology (ESIAT), 2010*, volume 3, pages 676–678.

Lu, W. and Miklau, G. (2008). Auditguard: A system for database auditing under retention restrictions. *Proc. VLDB Endow.*, 1(2):1484–1487.

Medina-Marin, J., Perez-Lechuga, G., and Li, X. (2009). Eca rule analysis in a distributed active database. In *Computer Technology and Development, 2009. ICCTD '09. International Conference on*, volume 2, pages 113–116.

Montesi, D. and Torlone, R. (1995). A transaction transformation approach to active rule processing. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 109–116.

Morgenstern, M. (1983). Active databases as a paradigm for enhanced computing environments. In *Proceedings of the 9th International Conference on Very Large Data Bases*, VLDB '83, pages 34–42, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Paton, N. W. and Díaz, O. (1999). Active database systems. *ACM Comput. Surv.*, 31(1):63–103.

Pimentel, M. and Fuks, H. (2012). *Sistemas Colaborativos*. Elsevier Editora Ltda. ISBN: 85-7383-260-6.

Qiao, Y., Zhong, K., Wang, H., and Li, X. (2007). Developing event-condition-action rules in real-time active database. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 511–516, New York, NY, USA. ACM.

Simon, F., Dos Santos, L. A., and Hara, S. C. (2008). Um sistema de auditoria baseado na análise de registros de log. *Escola Regional de Banco de Dados (ERBD'2008)*.

Smeaton, A. F. and Callan, J. (2005). Personalisation and recommender systems in digital libraries. *International Journal on Digital Libraries*, 5(4):299–308.

Zaniolo, C., Ceri, S., Faloutsos, C., Snodgrass, R., Subrahmanian, V. S., and Zicari, R. (1997). *Advanced Database Systems*. Morgan Kaufmann Publishers. ISBN: 85-7383-260-6.

sbbd:02

Avaliação da Localidade de Dados Intermediários na Execução Paralela de *Workflows BigData*

Douglas Ericson M. de Oliveira[1,2,3], Cristina Boeres[1], Angelo Fausti Neto [4], Fábio Porto[2]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)

[2]Data Extrem Lab (DEXL) – Laboratório Nacional de Computação Científica (LNCC)

[3]Faculdades de Educação Tecnológica do Estado do Rio de Janeiro (FAETERJ-Petrópolis)

[4]Laboratório Interinstitucional de e-Astronomia (LIneA)

doliveira@ic.uff.br, boeres@ic.uff.br, fporto@lncc.br, angelofausti@linea.gov.br

Abstract. *Processing huge data sets has been a challenge in science and business. Systems implementing the MapReduce programming model have obtained a great success and being followed by more expressive languages and refined processing techniques. In this paper, we are interested in evaluating the impact on data locality on the execution of Big Data processing in scientific workflows. We observe that, due to the huge size of intermediate results, reducing data transfer between producer-consumer activity pipeline must be a performance target. In order to evaluate this, we implement a real scientific workflow from astronomy using Hadoop and Spark. We show experimentally that data locality outperforms free distributed allocation and confirm the advantages of using RAM memory for storing intermediate results, when the latter fits in the former.*

Resumo. *O processamento de grandes datasets tem se mostrado um desafio para as áreas de ciência e da indústria. Sistemas implementando o modelo de programação MapReduce têm obtido grande sucesso e têm sido sucedidos por sistemas com linguagens mais expressivas e técnicas de processamento mais refinadas. Neste artigo, estamos interessados em avaliar o impacto da localidade de dados na execução do processamento de grandes volumes de dados por workflows científicos. Observa-se que, devido ao grande volume de dados produzidos entre as etapas do workflow, reduzir o volume de transferência entre atividades que estabelecem um fluxo de produtores-consumidores deve ser um objetivo para melhoria de desempenho deste tipo de aplicações. De forma a avaliar essa hipótese, implementou-se um workflow científico real da área de astronomia nos sistemas Hadoop e Spark. Mostra-se, de forma experimental, que a obtenção de localidade de acesso em arquivos intermediários produz execuções com melhor desempenho que aquelas em que a alocação é livre. Adicionalmente, confirmam-se os ganhos de desempenho quando o tamanho dos arquivos permite seu armazenamento em memória principal.*

1. Introdução

Processar grandes volumes de dados de forma eficiente tem se tornado uma questão a ser solucionada por empresas, ciência, governo, entre outros. Na área científica, pesquisas em diversas disciplinas tais como astronomia, biologia e física, veem o processo experimental

sendo transposto para o ambiente computacional baseado em dados. Instrumentos, como telescópios e sequenciadores de DNA, e simulações computacionais estão gerando um grande volume de dados a serem analisados.

Cientistas têm usado *workflows* para exprimir computacionalmente análises e experimentos científicos sobre dados *in-silico* [Romano 2008]. Tais *workflows* científicos definem um processo computacional composto por um conjunto de atividades estruturadas segundo uma ordem parcial e obedecendo uma dependência de dados entre as mesmas. Desta forma, uma atividade recupera o resultado em dados da atividade da qual depende.

O processamento de grandes volumes de dados, tradicionalmente, é realizado por sistemas de gerência de bancos de dados (SGBD). Em SGBDs relacionais, alcança-se desempenho satisfatório para processamento de dados segundo as operações da álgebra relacional, cuja semântica é conhecida. Adicionalmente, otimiza-se o armazenamento e acesso aos dados através de técnicas de distribuição de dados por discos e pela criação de índices de suporte às consultas, apenas para citar alguns princípios elementares de melhoria de desempenho [Shasha e Bonnet 2002] em SGBDs. Estamos entretanto, em um novo contexto. Primeiramente, *workflows* científicos apresentam atividades associadas a programas, cuja semântica é de *caixa-preta* (*i.e. desconhecida pela máquina de workflow que a executa*). Em segundo lugar, cada vez mais, adotam-se soluções baseadas no armazenamento de dados em sistemas de arquivos, fora do controle dos SGBDs. Finalmente, considerando-se o grande volume de dados a ser processado, estratégias adotam paralelismo de instâncias do *workflow* e particionamento dos dados por um *cluster*.

Neste sentido, estratégias como as implementadas por sistemas tais como Hadoop [White 2009] e Spark [Zaharia et al. 2010b] obtêm eficiência na execução de *workflows* científicos sobre grandes volumes de dados a partir do particionamento dos dados em um *cluster* e do processamento de tarefas em cada uma das partições. Nestes sistemas, *workflows* científicos precisam ser mapeados para os modelos computacionais (ex. *MapReduce*) de cada sistema, perdendo a perspectiva global da execução e, por consequência, reduzindo a possibilidade de obtenção de uma estratégia de execução de maior eficiência. Neste contexto, pode-se indagar sobre qual estratégias adotar para se obter um eficiente processamento de grandes volumes de dados por *workflows* científicos.

De forma a explorar esta questão, dividiu-se a investigação em dois estágios. Em um primeiro estágio realizado em [Oliveira et al. 2014], investigou-se o consumo de dados fonte, referindo-se a primeira atividade do *workflow* que acessa os dados a partir do meio de armazenamento em que foram disponibilizados às aplicações. O segundo estágio corresponde a dados resultantes do processamento de atividades que são consumidos pelas atividades subsequentes no *workflow* científico, chamados neste trabalho de arquivos intermediários.

Neste trabalho, interessou-se pelas estratégias de execução de *workflows* científicos que favoreçam a localidade dos dados intermediários. Considerou-se o modelo de execução paralelo *MapReduce* [Dean e Ghemawat 2008], conforme implementado pelos sistemas Hadoop e Spark (ibid). Exploraram-se, em particular, as implicações sobre a co-alocação ou não de atividades, em um *workflow* científico, envolvidas na produção-consumo de arquivos intermediários, bem como o tratamento especial para estes arquivos

adotado pela técnica RDD (*Resilient Distributed Dataset*) [Zaharia et al. 2012], de uso de memória RAM.

Nossos experimentos apontam para uma estratégia global de execução de *workflows* científicos que privilegie a localidade de referência das atividades em relação aos dados. A localidade de referência é um conceito reconhecido no processamento de consultas distribuídos [Özsu e Valduriez 2011]. Quando aplicado na avaliação de *workflows* científicos, observam-se ganhos de até 40% sobre a mesma execução em sistemas como o Hadoop onde tal localidade, em relação a dados intermediários, pode não ocorrer em função da ativação de *jobs* distintos para atividades de um mesmo *workflow* científico. Para sistemas como o Spark, o uso de memória para manutenção de arquivos intermediários traz, de fato, uma vantagem. No entanto, para arquivos intermediários que não cabem na memória disponível, a necessidade de gravação em disco leva à execuções piores do que os cenários de gravação em disco como avaliado em [Gu e Li 2013].

O restante do artigo está estruturado da seguinte forma: Na seção 2 apresentamos os trabalhos relacionados. A seção 3 apresenta as características e funcionamento das alternativas de execução investigadas. Estas por sua vez são avaliadas na seção 4. Finalmente, a seção 5 apresenta as conclusões.

2. Trabalhos Relacionados

O modelo *MapReduce* [Dean e Ghemawat 2008] para execução paralela de tarefas privilegia localidade de dados na alocação de tarefas. Sua implementação mais difundida, o sistema de execução de tarefas distribuídas Hadoop [White 2009], permite o armazenamento dos dados nos recursos que realizam o processamento, utilizando-se do sistema de arquivos distribuídos HDFS [HDFS 2012]. Neste contexto, o algoritmo de escalonamento FIFO (*First-In First-Out*), implementado pelo escalonador do sistema, leva em consideração a localidade dos dados [Zaharia et al. 2010a].

No algoritmo de escalonamento FIFO do Hadoop existem três níveis de localidade. Quando um nó trabalhador envia uma mensagem "disponível", o nó mestre tenta encontrar uma tarefa que possua seus dados de entrada armazenados localmente. Se encontrar tal tarefa, então a localidade a nível de nó é alcançada e a tarefa é assinalada a este nó. Se não encontrar tarefa nesta condição, o escalonador tenta encontrar uma tarefa cujos dados de entrada estejam armazenados no mesmo *rack* do nó demandante, obtendo assim a localidade no nível-*rack*. Se ainda assim falhar, uma tarefa é escolhida aleatoriamente e atribuída ao nó tendo dados acessados fora do *rack*, com perda de localidade. Observa-se que este algoritmo privilegia a localidade, porém possui limitações por causa da estratégia gulosa FIFO.

Em [Zaharia et al. 2010a], o *delay scheduling* foi utilizado para melhorar a localidade de dados do *Hadoop*. Neste trabalho foi projetado o HFS (*Hadoop Fair Scheduler*) que considera tanto a localidade quanto o *fair sharing*. Se uma tarefa não pode ser escalonada em um nó que tenha os dados, o escalonamento é adiado por um tempo. Este algoritmo possui um bom desempenho quando várias tarefas podem ser concluídas em um curto período de tempo. Uma peculiaridade deste algoritmo é a configuração do tempo de adiamento (*delay time*). Se este tempo for muito pequeno, não haverá nenhuma tarefa local a ser atribuída. Por outro lado, se o tempo de adiamento for muito grande, pode levar a estagnação da tarefa e afetar o desempenho do sistema. Portanto, para obter a

melhor localidade de dados, deve-se observar atentamente a configuração do tempo de adiamento.

Em [Zaharia et al. 2010b] é proposto o *Spark*, um *framework* para processamento paralelo de aplicações, assim como o Hadoop. A principal diferença entre os sistemas *Spark* e o *Hadoop*, do ponto de vista de localidade de dados, é a que no primeiro o armazenamento dos dados intermediários entre tarefas de um *workflow* é realizado em memória principal (RAM). No *Hadoop*, por outro lado, dados intermediários devem, obrigatoriamente, ser persistidos em um sistema de arquivos (local ou distribuído). Os resultados experimentais mostram significativa melhora em desempenho apresentada pelo Spark, realçando a importância do tratamento de dados intermediários.

Por outro lado, avaliar o uso da memória principal é importante, uma vez que as aplicações neste contexto processam grandes volumes de dados. Esta comparação foi realizada em [Gu e Li 2013], onde compara-se o desempenho em função da quantidade de memória utilizada. Concluiu-se que o Spark apresenta um melhor desempenho enquanto existe memória RAM disponível suficiente. Porém, no caso contrário, o uso de espaço em *swap* tem como consequência a diminuição de desempenho, sendo superado por seu competidor.

Vários trabalhos atacam o problema da localidade de dados como em [Ranganathan e Foster 2002], [Palanisamy et al. 2011], [Abad et al. 2011], [Zhang et al. 2011] e [Seo et al. 2009]. Estes trabalhos procuram melhorar a localidade de dados avaliando cada atividade em separado, e não as atividades relacionadas, como em *workflows*. Isto difere da proposta sendo apresentada neste trabalho que pretende avaliar como sistemas de processamento de grandes volumes de dados tratam a localidade de dados, em particular quando tratam de *workflows*, uma vez que possuem mais de uma atividade.

HaLoop [Bu et al. 2010] é um *framework MapReduce* para execução de aplicações iterativas. O Hadoop cria uma tarefa separada para cada iteração, e isto leva a um *overhead* causado principalmente pelo acesso ao disco para armazenamento dos dados intermediários das tarefas. O HaLoop trata aplicações iterativas que possuem como característica a maior parte dos dados não sofrer alteração entre uma iteração e outra. Para isso, o HaLoop realiza um *caching* dos dados inalterados minimizando a E/S. Diferentemente deste trabalho, onde os dados são totalmente processados e transformados ao longo da execução do *workflow* não-iterativo. Um ponto em comum é o fato do HaLoop também utilizar a localidade para escalonar as atividades do *workflow*. O HaLoop escalona a tarefa subsequente no recurso que contém a partição de dados que será necessária a tarefa. Este escalonamento dinâmico realiza a ativação do *framework* entre todas as atividades do *workflow*, trazendo um *overhead* como demonstrado nos experimentos demonstrados na seção 4.1.

Na mesma classe de aplicações o HyMr [Ruan et al. 2012] realiza uma integração do Twister [Ekanayake et al. 2010] com o Hadoop, utilizando o Twister para gerenciamento das atividades iterativas e o Hadoop para as demais atividades (não-iterativas). Além disso, utiliza o HDFS para armazenamento dos dados das atividades iterativas adicionando tolerância à falhas.

Diferentemente, este trabalho cria o conceito de unidade de trabalho local, que

corresponde ao fragmento do *workflow* que deve ser garantidamente executado em um mesmo nó. De forma a implementar este conceitos em *frameworks* em que não há uma avaliação global das atividades envolvidas em um *workflow* científico, está sendo usada a máquina de execução de *workflows* QEF (*Query Evaluation Framework*) [Porto et al. 2007]. QEF é uma máquina de execução algébrica que pode ser estendida com operadores correspondendo às atividades de um *workflow* científico. A definição no QEF de um fragmento local de *workflow* garante que as atividades que comunguem da localidade sejam executadas como uma atividade, do ponto de vista do sistema paralelo de execução, como o *Hadoop* ou *Spark*. Este artifício permitiu, neste trabalho, avaliar o impacto de se garantir explicitamente a localidade de acesso a dados intermediários compartilhados por atividades em um *workflow* científico executando sobre os sistemas paralelos em questão.

3. Modelo de Avaliação do Problema

De forma a explorar as implicações do tratamento de arquivos intermediários em *workflows* científicos, definiu-se um cenário de avaliação. Este é composto de um *workflow* científico real, fornecido pela equipe do laboratório LIneA, chamado de *SkyMap*.

3.1. *Workflow Científico SkyMap* - Caso de Uso

O *workflow* científico *SkyMap* tem por objetivo produzir um histograma da região do céu investigada. Cada objeto celeste identificado no levantamento astronômico aparece como um ponto na posição indicada por sua coordenada esférica. O sistema de coordenadas esféricas adotado considera duas dimensões: *right ascention - ra* e *declination- dec*, que identificam cada ponto na esfera celeste que envolve a terra, de forma similar a latitude e longitude. O conjunto de objetos celestes identificados em um levantamento astronômico forma um catálogo, em que cada entrada corresponde a um objeto. O *workflow SkyMap* acessa um catálogo astronômico produzido pelo levantamento *Dark Energy Survey - www.darkenergysurvey.org*. Em nossos experimentos, o catálogo utilizado contém cerca de 30 milhões de objetos celestes. Estes objetos são acessados pelo *workflow* a partir de um arquivo armazenado de forma distribuída no sistema de arquivos HDFS. O *workflow SkyMap* produz o histograma do céu pintando cada um dos objetos celestes contidos no catálogo em uma representação 2D da região de interesse do céu. O processo é realizado modelando um *workflow* com as seguintes atividades, conforme apresentado na Figura 1-(A):

(1) **Leitura e Tratamento dos dados**: recuperação dos dados a partir do catálogo e tratamento para adequação ao formato esperado pelo programa escrito em Python, *SkyMap*. O resultado desta atividade é armazenado em um arquivo intermediário.

(2) **SkyMap**: programa escrito em Python que, para cada objeto do catálogo obtido na etapa anterior, colore o ponto respectivo em um histograma da região do céu sendo analisada. O programa realiza a leitura do arquivo gerado na atividade anterior e gera o histograma parcial em um arquivo de saída, segundo um formato específico.

(3) **SkyMapAdd**: este programa realiza a consolidação dos histogramas parciais, produzindo um arquivo único com o histograma completo.

Observe que o *workflow SkyMap* descrito acima pode ser mapeado para o modelo *MapReduce*, conforme apresentado na Figura 1-(A). Nesta figura, as atividades (1) e (2)

são associadas a processos do tipo *MAP*, enquanto a atividade (3) seria mapeada para um processo do tipo *Reduce*. O *workflow SkyMap*, apesar de simples, oferece oportunidades de investigação no tratamento de localidade de dados para o arquivo intermediário compartilhado entre as atividades (1) e (2).

3.2. Exploração de Alternativas para o tratamento de arquivos intermediários

A partir do *workflow* real *SkyMap*, discutem-se alternativas para o tratamento de localidade de dados associada a arquivos intermediários em *workflows* científicos. Consideram-se três alternativas de execução das tarefas (1) e (2) do *workflow SkyMap* que se comunicam por meio de um arquivo intermediário.

As avaliações exploram alternativas de mapeamento do *workflow* para os sistemas *Hadoop* e *Spark*. As variações adotadas são apresentada a seguir :

(1) Hadoop: Mapeamento do *workflow* no modelo *MapReduce* adotado pelo sistema Hadoop, ou seja, cada atividade do *workflow* é mapeada para uma das funções, *Map* ou *Reduce*. Neste modelo, os dados intermediários devem ser armazenados no sistema de arquivos distribuído do Hadoop, HDFS [HDFS 2012].

(2) Hadoop/QEF-Local: Mapeamento do *workflow* para o modelo *MapReduce* adotado pelo sistema Hadoop. De forma a garantir a execução em um mesmo nó das atividades que compartilham um arquivo intermediário, utiliza-se a máquina QEF para processar o fragmento constituído por essas atividades. A função *Map* ativa a execução da máquina QEF, garantindo o tratamento local ao fragmento do *workflow*. Adicionalmente, o arquivo intermediário é armazenando no sistema de arquivos local. Esta alternativa é denominada neste trabalho de *HaQoop* local.

(3) Hadoop/QEF-HDFS: Semelhante ao modelo anterior, baseado em Hadoop e QEF, quanto à alocação das atividades em um mesmo nó. Os dados intermediários, no entanto, são armazenados no sistema de arquivos distribuído HDFS. Será denominado no decorrer do trabalho como *HaQoop* HDFS.

(4) Spark: Implementação do *workflow* no Spark, neste modelo os dados intermediários são armazenados em memória RAM através dos RDDs [Zaharia et al. 2012].

3.3. Discussão sobre as Alternativas

(1) Alternativa Hadoop: Na Figura 1-(C) apresenta o mapeamento adotado nesta alternativa. Sua implementação no *framework* Hadoop requer o mapeamento do *workflow* em dois *jobs*/tarefas. Sendo assim a atividade (1) corresponde a um *job* que só possui a etapa *MAP*, as atividades (2) e (3) compõem o segundo *job* com as etapas *Map* e *Reduce* respectivamente.

Em uma execução paralela da atividade (1), cada *Map* realiza a leitura de sua respectiva partição de dados e os processa preparando-os segundo o formato esperado pelo programa *SkyMap*. O armazenamento do resultado produzido pela atividade (1) se dá em um sistema de arquivos distribuído (HDFS). Neste modelo, o *Map* implementando a atividade (2) pode ser iniciado em qualquer nó, local ou não em relação ao nó que produziu o arquivo intermediário a ser consumido. O mesmo processo ocorre na atividade (3) na etapa *Reduce*. Neste caso, é obrigatório o armazenamento dos dados produzidos pela atividade (2) no HDFS, pois o *Reduce* é processado por somente uma máquina escolhida

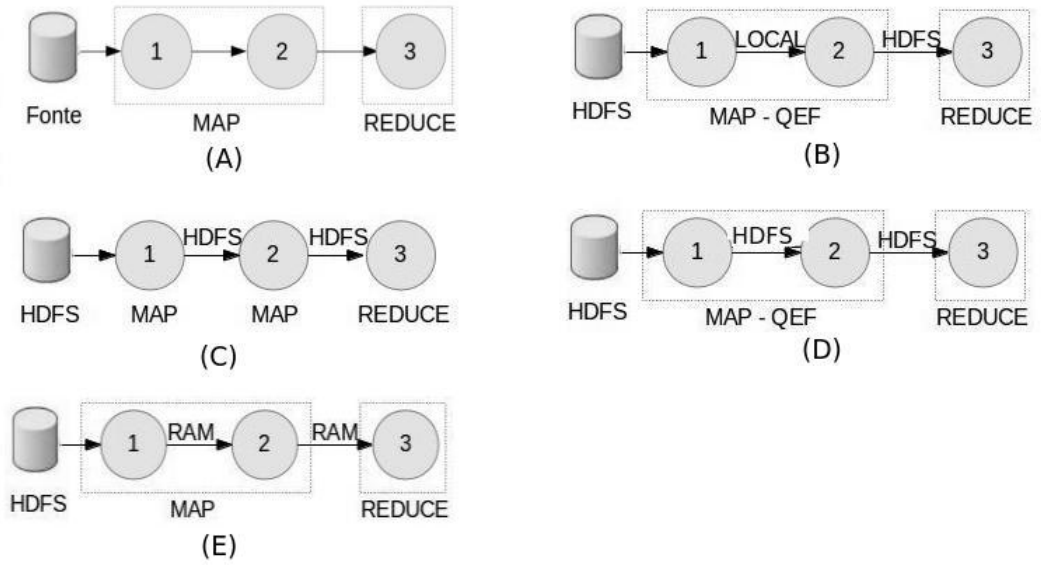


Figura 1. *Workflow* da Aplicação *SkyMap* e Alternativas de Execução

aleatoriamente pelo Hadoop. A máquina escolhida para executar o *Reduce* deve ser capaz de recuperar os dados produzidos pelo *Map*, neste caso realizado pelo sistema de arquivo distribuído HDFS.

A discussão acima realça o problema da ativação de *jobs* diferentes, pelo *framework* Hadoop. Dois elementos entram em cena nesta discussão: a alocação das atividades que compartilham arquivos intermediários, assim como, o armazenamento destes arquivos que deve ser realizada no HDFS.

(2) Alternativa HaQoop: A Figura 1-(B) apresenta este mapeamento, considerando sua implementação no *framework* Hadoop segundo um mapeamento do *workflow* em um único *Map* e um *Reduce*. Sendo assim, as atividades (1) e (2) são integradas via QEF em um único *Map* e a atividade (3) compõe a etapa de *Reduce*. Adicionalmente, garante-se o acesso local aos dados intermediários gravando-se os mesmos no sistema de arquivos local. Nesta alternativa, a execução deste conjunto de atividades ocorre garantidamente de forma local, diferentemente da alternativa *Hadoop*. A execução local de atividades se traduz na localidade de acesso aos arquivos intermediários envolvidos.

(3) Alternativa HaQoop - HDFS: O modelo apresentado na Figura 1-(D) apresenta alternativa semelhante a HaQoop-Local. O mapeamento da alternativa no *framework* Hadoop se traduz, igualmente, em um único *Map* e *Reduce*. Com isso as atividades (1) e (2) continuam fazendo parte da etapa *Map* e a (3) compondo a etapa *Reduce*. Esta alternativa explora as consequências da execução local de atividades quando os dados intermediários compartilhados não são necessariamente acessados localmente. Para isso, considera-se o armazenamento de arquivos intermediários no sistema distribuído HDFS.

(4) Alternativa Spark: Na Figura 1-(E) apresenta-se o mapeamento visando a execução do *workflow* no *framework* Spark. O objetivo da avaliação desta alternativa é investigar o efeito na gravação de arquivos intermediários em memória RAM.

4. Experimentos

Nesta seção são descritos os experimentos realizados segundo as alternativas de mapeamento apresentados na seção 3. Para a realização dos testes foi preparado um ambiente de processamento com as seguintes características: 11 máquinas físicas cada uma com processador Intel Xeon ES4607, 2.2GHz, 24 núcleos reais por processador, com possibilidade de *Hyper-threading*, 128GB memória e 15TB de disco rígido. Um ambiente com 11 máquinas virtuais de criadas com o *hypervisor* KVM (*Kernerl-based Virtual Machine*) sendo 1 mestre e 10 trabalhadores, uma por máquina real cada uma com 2 núcleos, 12GB memória, 200GB de disco rígido. Os *Frameworks* utilizados foram Hadoop, QEF, Spark, conforme já mencionados.

Os testes foram feitos com as seguintes variações no número de *Maps*:2, 4, 6, 8, 10 e 20. Embora as máquinas utilizadas para os testes possuam mais de um núcleo, a fase de *Map* inicia um único processo em cada máquina utilizando somente um núcleo. Para cada ambiente e variação de *Maps* foram realizadas 15 execuções consecutivas. Foram extraídos os valores máximos de cada execução e obtida a média desses valores.

4.1. Comparação do tempo total de execução

A Figura 2 (A) apresenta uma avaliação do tempo total de execução do workflow entre as alternativas de avaliação. Aparece em realce a diferença de desempenho entre o Hadoop e o HaQoop. Por exemplo, com 20 *Maps* percebe-se claramente que a diferença entre ambas as alternativas é bem pequena nas três atividades que compõem o *workflow*, porém o tempo total de execução apresenta uma diferença muito grande entre eles.

A justificativa para este resultado está no custo de inicialização do segundo *Map* na alternativa Hadoop. O Hadoop realiza duas chamadas *Map* e a *Reduce*. Com isso, o *framework* Hadoop é solicitado entre as atividades (1) e (2) no modelo Hadoop. O HaQoop realiza somente um *Map* uma vez que as atividades (1) e (2) fazem parte do fragmento de execução local executado pelo QEF. Além da garantia da localidade de dados isto também beneficia o desempenho do *workflow* pelo fato de permitir atividades de um mesmo tipo serem unidas em um único fragmento e executadas como uma só.

Uma análise de cada atividade modelada pelo *workflow* é avaliada separadamente. Em todos os casos, observando a Figura 2, é visto que a alternativa Spark teve o melhor desempenho em todas as atividades e consequentemente no tempo de execução total. Percebe-se que a utilização dos RDDs que mantém os dados intermediários na memória RAM beneficia o desempenho do *workflow*. Sabe-se que a memória RAM possui maior velocidade do que a memória secundária, porém uma menor capacidade de armazenamento. Nos experimentos aqui realizados, a quantidade de dados em cada partição era alocada devidamente na memória disponível. Esta relação de desempenho e capacidade de armazenamento é avaliada em [Gu e Li 2013] para quantidades de dados bem maiores.

4.2. Leitura e Tratamento dos dados

Na Figura 2 (B) apresentam-se os tempos de execução da primeira atividade - Leitura e Tratamento dos Dados. As diferenças nesta atividade entre as alternativas de execução avaliadas são: gravação dos dados produzidos por esta atividade e utilização do QEF.

É possível observar um melhor desempenho do HaQoop com relação aos demais, a partir da execução com 6 *Maps*. Porém, nas execuções com 2 e 4 *Maps*, o desempenho

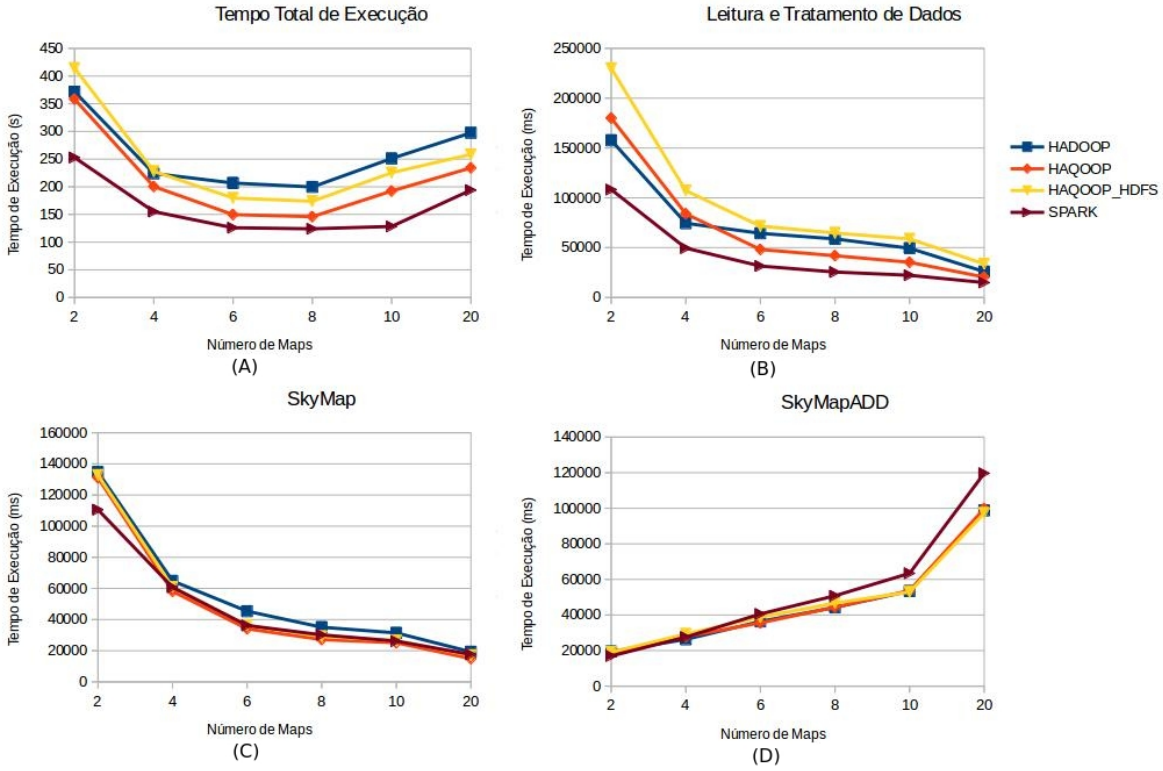


Figura 2. Tempo de Execução com as alternativas - Hadoop, HaQoop, HaQoop-HDFS e Spark

inferior do HaQoop com relação ao Hadoop justifica-se pelo fato de possuir uma camada de *software* adicional (*i.e. QEF*). Com isso, o HaQoop produz um custo ligeiramente maior de processamento do que no Hadoop simples, impactando no desempenho das execuções com 2 e 4 *Maps*.

Na alternativa Hadoop, a gravação é realizada no HDFS, porém localmente. O HDFS atua como um intermediário na gravação dos dados locais de cada *Map*. Consequentemente, a medida em que aumenta-se o número de *Maps* paralelos, aumenta também a concorrência no HDFS como intermediário único na gravação dos dados de vários *Maps*. Pode-se verificar que nas execuções com 2 e 4 *Maps* o Hadoop possui um desempenho superior ao HaQoop, mas o impacto da concorrência entre Maps supera aquele devido ao QEF quando o número de *Maps* aumenta, como pode-se observar na Figura 2 (B).

O HaQoop-HDFS une as duas desvantagens apresentadas acima. A desvantagem do Hadoop, representada pela concorrência na gravação dos dados, e a do HaQoop causada pela utilização do QEF. Com isso, observa-se um desempenho inferior comparado as demais em todas as variações realizadas.

4.3. SkyMap

Na Figura 2 (C) apresentam-se os tempos de execução da atividade *SkyMap*. As diferença na execução desta atividade, entre as alternativas discutidas, estão no modelo de obtenção dos dados intermediários, gerados pela atividade (1), e na utilização do QEF, nas estratégias HaQoop e HaQoop-HDFS. É possível observar um melhor desempenho do HaQoop-local com relação aos demais, pois os dados são lidos diretamente a partir do

sistema de arquivos local. Apesar do melhor desempenho, o HaQoop não garante a tolerância à falhas, em cada atividade, fornecida pelos demais através do HDFS.

No HaQoop-HDFS, os dados são obtidos a partir do HDFS, que prioriza o armazenamento local. Com o QEF sendo invocado pelo *Map*, garante-se que a atividade (2) será executada localmente ao mesmo conjunto de dados da atividade (1) . O desempenho levemente inferior com relação ao HaQoop deve-se somente ao fato de utilizar o HDFS como um intermediário nesta obtenção dos dados.

4.4. SkyMapAdd

Na Figura 2 (D) apresenta-se os tempos de execução somente da atividade *SkyMapAdd*, que é realizada como uma tarefa *Reduce*. Esta atividade possui a mesma implementação em todas as estratégias analisadas. Com isso constata-se que o tempo de execução é semelhante para todas as estratégias. Uma vez que só um nó irá executar a etapa *Reduce*, e este só é definido em tempo de execução, os dados de entrada devem estar no HDFS permitindo assim que independentemente do nó escolhido este tenha acesso aos dados.

Ainda, pode-se perceber que o desempenho desta atividade piora a medida que aumenta o número de *Maps*. Este problema está diretamente relacionado a estratégia de particionamento dos dados de entrada. Nestes experimentos optou-se por uma particionamento balanceado dos dados sem se importar com a localização espacial do objeto. Por exemplo, possuindo 10 nós trabalhadores e 30 milhões de objetos, cada recurso irá receber 3 milhões de objetos.

A atividade paralela *SkyMap* recebe os objetos correspondentes a partição, e o intervalo espacial (ra e dec) em que estes objetos se encontram, gerando um histograma parcial proporcional ao tamanho do intervalo. Como o particionamento não considerou a localização espacial, o intervalo foi definido como o mínimo e o máximo possível para todas as atividades paralelas *SkyMap*. Portanto, o tamanho dos histogramas produzidos por cada *Map* é o mesmo independente da quantidade de objetos processada. Por exemplo, possuindo 1 nó trabalhador e 30 milhões de objetos, será gerado um histograma com 10 MB. Em outro caso, com 10 nós e 30 milhões de objetos, cada nó irá processar 3 milhões de objetos, porém será produzido 10 histogramas cada um com 10 MB.

A atividade *SkyMapAdd* recebe como entrada estes histogramas parciais. Pode-se concluir que a quantidade de arquivos de entrada desta atividade é diretamente relacionada ao grau de paralelismo. Quanto maior o grau de paralelismo maior a quantidade de histogramas gerados e maior será o tempo de processamento do *SkyMapADD*, o que pode se tornar um gargalo.

5. Conclusão

O desenvolvimento das ciências *in-silico* tem apresentado novos desafios no processamento de grandes volumes de dados. Os *workflows* científicos tem sido utilizados para exprimir computacionalmente análises e experimentos científicos. Atualmente, *frameworks* de execução de aplicações como Hadoop e Spark tem aumentado a eficiência na execução de *workflows* sobre grandes volumes de dados. Nestes sistemas, perde-se a perspectiva global da execução e reduz-se a possibilidade de obtenção de uma estratégia de execução de maior eficiência. Neste trabalho, foram avaliadas quatro alternativas de execução

para se obter um eficiente processamento de grandes volumes de dados por *workflows* científicos.

No Hadoop verificou-se que a execução do *workflow* utilizando dois *Maps* influenciaram o desempenho. Além disso, o último *Map* pode ser executado em um nó diferente do que foi utilizado para execução do primeiro *Map*, ou seja, caso algum *Map* não alcance a localidade na execução da segunda atividade isto implica em uma transferência de dados. Por estas razões o Hadoop possui um desempenho levemente inferior quando comparado ao HaQoop-HDFS. Conclui-se que a unidade de trabalho local influencia diretamente na execução da aplicação. Ela garante várias atividades executando sobre um mesmo *Map* e a localidade de dados, verificada no HaQoop-HDFS e HaQoop. Além da importância da localidade de dados demonstrada, este trabalho também apresenta a questão da estratégia de particionamento dos dados de entrada. Neste, optou-se pelo balanceamento igual sem levar em consideração a posição espacial de cada objeto. Este fato impactou negativamente o desempenho da atividade *SkyMapAdd*. Como próximo trabalho iremos analisar estratégias de particionamento que considerem a posição espacial para melhorar a eficiência desta classe de aplicações.

6. Agradecimentos

Este trabalho foi parcialmente financiado pelo CNPq, Bolsa de Produtividade em Pesquisa, Processo:309494/2012-5.

Referências

Abad, C. L., Lu, Y., and Campbell, R. H. (2011). Dare: Adaptive data replication for efficient cluster scheduling. In *Proceedings of the 2011 IEEE International Conference on Cluster Computing*, CLUSTER '11, pages 159–168, Washington, DC, USA. IEEE Computer Society.

Bu, Y., Howe, B., Balazinska, M., and Ernst, M. D. (2010). Haloop: Efficient iterative data processing on large clusters. *Proc. VLDB Endow.*, 3(1-2):285–296.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113.

Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., and Fox, G. (2010). Twister: A runtime for iterative mapreduce. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 810–818, New York, NY, USA. ACM.

Gu, L. and Li, H. (2013). Memory or time: Performance evaluation for iterative operation on hadoop and spark. In *10th IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, HPCC/EUC 2013, Zhangjiajie, China, November 13-15, 2013*, pages 721–727.

HDFS (2012). http://hadoop.apache.org/docs/stable/hdfs-design.html.

Oliveira, D. E. M., Boeres, C., and Porto, F. (2014). Analise de estrategias de acesso a grandes volumes de dados. In *SBBD Proceedings 2014*.

Özsu, M. T. and Valduriez, P., editors (2011). *Principles of Distributed Database Systems*. Springer, New York, 3 edition.

Palanisamy, B., Singh, A., Liu, L., and Jain, B. (2011). Purlieus: Locality-aware resource allocation for mapreduce in a cloud. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 58:1–58:11, New York, NY, USA. ACM.

Porto, F., Tajmouati, O., Silva, V. F. V. D., Schulze, B., and Ayres, F. V. M. (2007). Qef - supporting complex query applications. In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 846–851, Washington, DC, USA. IEEE Computer Society.

Ranganathan, K. and Foster, I. (2002). Decoupling computation and data scheduling in distributed data-intensive applications. In *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing*, pages 352–358. IEEE Comput. Soc.

Romano, P. (2008). Automation of in-silico data analysis processes through workflow management systems. *Brief Bioinform*, 9(1):57–68.

Ruan, Y., Guo, Z., Zhou, Y., Qiu, J., and Fox, G. (2012). Hymr: A hybrid mapreduce workflow system. In *Proceedings of the 3rd International Workshop on Emerging Computational Methods for the Life Sciences*, ECMLS '12, pages 39–48, New York, NY, USA. ACM.

Seo, S., Jang, I., Woo, K., Kim, I., Kim, J.-S., and Maeng, S. (2009). HPMR: Prefetching and pre-shuffling in shared MapReduce computation environment. In *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, pages 1–8.

Shasha, D. and Bonnet, P. (2002). Database tuning: Principles, experiments, and troubleshooting techniques. In *VLDB*. Morgan Kaufmann.

White, T. (2009). *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 1st edition.

Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., and Stoica, I. (2010a). Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, pages 265–278, New York, NY, USA. ACM.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–2, Berkeley, CA, USA. USENIX Association.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010b). Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA. USENIX Association.

Zhang, X., Zhong, Z., Feng, S., Tu, B., and 0002, J. F. (2011). Improving data locality of mapreduce by scheduling in homogeneous computing environments. In *ISPA*, pages 120–126. IEEE.

Clustering Multivariate Climate Data Streams using Fractal Dimension*

Christian C. Bones[1], Luciana A. S. Romani[2], Elaine P. M. de Sousa[1]

[1]University of São Paulo - USP
São Carlos – SP – Brazil

[2]Embrapa Agriculture Informatics
Campinas – SP – Brazil

{chris,parros}@icmc.usp.br, luciana.romani@embrapa.br

Abstract. *A data stream is a flow of data produced continuously along the time. Storing and analyzing such information become challenging due to exponential growth of the data volume collected. In this context, some methods were proposed to cluster data streams with similar behavior along the time. However, those methods have failed on clustering data flows with more than one attribute, i.e., multivariate flows. This paper introduces a new method to cluster multivariate data streams, based on fractal dimension, reading the data only once. We evaluated our method over real multivariate data streams generated by climate sensors. Not only was our method able to cluster the flows of data, but also identified sensors with similar behavior during the analyzed period.*

1. Introduction

The increasing number of devices and sensors that continuously generate a huge amount of data leads to new challenges and applications. For instance, sensors have been used to monitor the pollution in cities, the level of rivers (to prevent flooding) and the meteorological conditions. This flow of data, generated ad infinitum at a high speed rate, is called data stream.

In this scenario, clustering of data streams becomes an active research topic [Bifet and De Francisci Morales 2014, Zhang et al. 2014, Pereira and de Mello 2014, Widiputra et al. 2011] with applications in several contexts. Its goal is grouping in the same cluster data streams that have similar properties and behavior over time, whereas data streams of different clusters must present dissimilar characteristics. For instance, clustering tecniques could be applied to cluster data streams from climate sensors that have similar behavior along a period of time.

Some of the main challenges on extracting knowledge from data streams are: read data only once; capture and represent data evolution along the time; provide answers as soon as the user demand them. It is also desirable to deal with multidimensional data, considering all the variables involved on each stream, i.e., multivariate data streams. Some methods have been developed to work with multiple data streams [Rodrigues et al. 2008, Chaovalit and Gangopadhyay 2009, Widiputra et al. 2011, Pereira and de Mello 2014]. Their approaches in data streams clustering fall into two main categories that are: *Clustering by example*, in which all data points are clustered independently, regardless of the

*The authors are greatful to CAPES, CNPQ and FAPESP for their financial support.

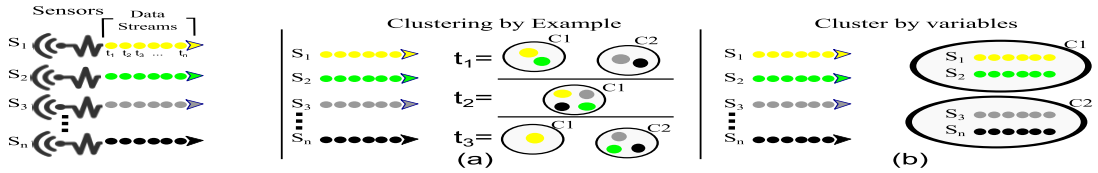


Figure 1. Difference between clustering by examples and clustering by variables.

data sources these data points are from; *Clustering by variable* or *Clustering the entire data stream*, in which a data stream is compared with other data streams and those similar will be clustered into the same cluster. Fig. 1 shows the difference between these two approaches: Fig. 1(a) illustrates clustering by examples at different times t_i such that clusters C_1 and C_2 are created considering the properties of individual data points at time t_1 and, therefor, points from different streams can be at the same cluster. On the other hand, the clustering by variable approach in Fig. 1(b) shows that each cluster is built considering the general properties of the entire flow of data from each sensor.

Although, several online clustering methods have been proposed to process and analyze flows of data in real time [Rehman et al. 2014, Pereira and de Mello 2014, Zhang et al. 2014, Widiputra et al. 2011, Chaovalit and Gangopadhyay 2009, Rodrigues et al. 2008], only few of them try to extract valuable information and group the entire data streams based on their similar behavior over the time [Pereira and de Mello 2014, Widiputra et al. 2011, Chaovalit and Gangopadhyay 2009, Rodrigues et al. 2008]. Furthermore, these methods have failed on clustering data streams that have more than one attribute, i.e., multivariate flows, because they only consider the similarity of attributes independently.

In this work, we are interested in clustering the entire data stream, as we focus on clustering climate sensors that behave similarly along the time. Moreover, we are especially interested in clustering sensors that generate multivariate data streams, taking into account the correlation among the attributes. As climate data streams usually have more than one attribute (e.g. temperature and precipitation) and, in such way, those attributes may have some correlation, considering each attribute as an independent flow is not the best approach for clustering climate sensors.

In this context, this paper proposes the Fractal-based Clustering of Data Streams framework (FCDS), whose the main module is a novel method for clustering multivariate data streams, based on the calculation of fractal dimension. Our method utilizes the fractal dimension, calculated for data streams that have more than one attribute, to cluster data streams that behave similarly along the time. Also, our method keeps the evolution of the data stream by checking cluster membership whenever a new value of fractal dimension is obtained. In other words, our method checks if the data stream is still belonging to the same cluster or if it should be allocated in another one that better describes its behavior in that period of time.

In order to evaluate our method, we used a dataset provided by EMBRAPA Agriculture Informatics - Empresa Brasileira de Pesquisa Agropecuária - Campinas. The dataset includes multivariate data streams from climate sensors of different Brazilian regions. Our results not only indicated that our approach can be a useful method to assist

specialists in analyzing large amounts of climate data, but also helps to identify regions with the same behavior along the time.

The rest of this paper is organized as following. Section 2 presents background concepts and related work. Section 3 describes our approach to cluster data streams. Experimental results are discussed in Section 4 and Section 5 presents final remarks and future work.

2. Background and Related Work

Data stream is an ordered collection of data $s_1, s_2, ...$, generated continuously by one or more sources, that can be read only once [Guha et al. 2003]. In addition, a data stream could have more than one attribute per datum, being called a multivariate data stream. Formalizing these concept, let $\mathbb{S} = \{S_1, \ldots, S_n\}$ be a set of data streams sources where each $S_i = \left\{\vec{d_1}, \ldots, \vec{d_\infty}\right\}$ is a multivariate data stream. Also, each S_i is assumed to contain f attributes such that $\vec{d_i} = [a_1, \ldots, a_f]$ is the set of attributes.

Clustering data stream is a technique used in data mining to perform the grouping of flows of data, so that objects within the same group must have very similar characteristics and properties [Aggarwal et al. 2006]. These characteristics should differ as much as possible from group to group. One of the most common ways to cluster objects is to measure the distance between them [Gama 2010]. However, clustering similar characteristics can be very costly [Guha et al. 2003]. Also, clustering data streams is used to overview the data distribution and as a pre-processing for other algorithms[Gama 2010]. In order to achieve this goal, some basic requirements must be presented in data stream cluster algorithms [Barbará 2002]: (i) Representation of compact size; (ii) Quickly and incremental processing of new data items; (iii) Traceability of changes in groups; (iv) Quick and clear identification of outliers.

Therefore, any new data stream clustering method must be adapted to perform clustering in a continuous, concise and evolving online way entry of the observed sequence. Furthermore, the temporal characteristic of the data stream must also be considered using a small amount of storage space [Guha et al. 2003] and processing time. Such requirements are imposed to contemplate the continuous manner in which the data arrives and the need for analyzing them in real time [Gama 2010].

2.1. Clustering by Variable

Most of the methods proposed in the literature to cluster data streams deal with flows that have only one attribute. Those methods that support more than one attribute do not consider the correlation among attributes to cluster the data streams [Chaovalit and Gangopadhyay 2009, Rodrigues et al. 2008, Rehman et al. 2014, Miller et al. 2014]. They only consider to cluster if all the attributes are similar with all the attributes of other data stream. Leading to results that do not correspond to the behavior of the data streams along the time.

Beyond that, the number of methods that cluster data streams that behave similarly over the time is even more restricted such as the ECM method [Widiputra et al. 2011] and POD-Clus method [Chaovalit and Gangopadhyay 2009]. Thus, this two method will be

detailed in the next sections and they also will be used to compare the achievements of our approach.

ECM

The Evolving Clustering Method (ECM) performs predictions using univariate data streams [Widiputra et al. 2011]. ECM builds local models in two main steps, which are the extraction of relationships between data streams profiles and the detection and clustering recurrent trends when a particular profile emerges. ECM calculates the relationship among data streams profiles using Pearson's correlation [Rodrigues et al. 2008], extracting only the most significant coefficients obtained through tests of statistical significance. Given two data streams a and b the Pearson's correlation is:

$$corr(a,b) = \frac{P - \frac{AB}{n}}{\sqrt{A_2 - \frac{A^2}{n}}\sqrt{B_2 - \frac{B^2}{n}}} \tag{1}$$

where $A = \sum a_i$, $B = \sum b_i$, $A_2 = \sum {a_i}^2$, $B_2 = \sum {b_i}^2$, $P = \sum a_i b_i$, that can be easily upgradeable as soon as each new datum arrives. Then, the ECM calculates the dissimilarity between the data streams a and b by RNOMC (Rooted Normalized One-Minus-Correlation) equation, developed in the ODAC method [Rodrigues et al. 2008] and presented in Equation 2. However, unlike the ODAC, the ECM requires that the whole time series is previously available to perform the calculations offline. Based on a set of decision rules, the algorithm decides how to group the series opting to create, remove, or join groups.

$$rnomc(a,b) = \sqrt{\frac{1 - corr(a,b)}{2}} \tag{2}$$

Because ECM uses the same measure of dissimilarity used by ODAC [Rodrigues et al. 2008], it only detects linear relationships. Another point that limits the application of ECM is how the algorithm decides the union, creating or adding a new element to the cluster, for example, as an element d_j will only be added to the group of d_i if d_j is correlated with all existing elements in d_i, so if d_i has thousand elements and d_j has only one element no correlated with d_i, then d_j will not be added to d_i's cluster. Another disadvantage is that ECM clusters data streams with only one attribute.

POD-Clus

The POD-Clus algorithm (*Probability and Distribution-based Clustering*) [Chaovalit and Gangopadhyay 2009] has four approaches: *clustering by examples* and *clustering the entire data streams* without catching the evolution of the clusters; *clustering by examples* and *clustering the entire data streams* monitoring the evolution of the clusters. Only the latter one is of interest in the context of this work.

The POD-Clus seeks to maintain summaries and discard detailed information of the data points, using normal distributions for this purpose. Each POD-Clus' cluster k receives n data points and stores some average statistics: such as the average μ, standard deviation σ and the updated covariance matrix, whenever new data arrives. These statistics are used to measure the similarity between data streams considering each attribute according to the equation 3. The distance of a muiltivariate data stream S to a cluster

center C is defined as [Kumar and Patel 2007]:

$$D_{SC} = \sum_{f=1}^{F} \frac{(\mu_{Sf} - \mu_{Cf})^2}{\sigma_{Sf}^2}, \quad (3)$$

when S denotes a data stream, C denotes a cluster, f denotes a data feature, μ_{Sf} is the mean of data stream S's feature f, σ_{Sf} is the standard deviation of data streams S's feature f, and μ_{Cf} is the mean of the cluster C's feature f.

POD-Clus also monitors the progress of each cluster, identifying the emergence, the disappearance, the union and the division of the clusters. To detect the appearance of a new group, it maintains a cluster of outliers and when one of these clusters reaches the minimum amount of data streams defined, this cluster becomes a new effective cluster. A cluster disappears when it stops receiving new data by a certain amount of time and old data receives less importance according to a fading factor. To join a cluster to another, it is checked the amount of data which may be in both clusters, generating an overlap between them. If the amount of overlapping data is greater than a predetermined threshold, then the two clusters are merged. To split a group POD-Clus monitors its density and compares it with the normal distribution. If there is significant difference, the group is divided.

The POD-Clus assumes that the whole data stream is derived from a normal distribution that does not guarantee a good representation of the clusters. Another POD-Clus' drawback is the fact that to join a data stream to a cluster all its attributes f have to be similar to the C's features f, disregarding the correlation between the attributes.

2.2. Fractal Dimension

In this work we propose a clustering data stream method based on the calculus of fractal dimension that is used to identify the correlation among the attributes of a data stream in order to capture the data streams' behavior over the time. Then, this behavior will be used to measure the similarity among data streams in order to achieve better clusters results.

A fractal is characterized by the self-similarity property, i.e., it is an object that presents roughly the same characteristics when analyzed over a large range of scales. From the Fractal Theory, the Correlation Fractal Dimension D_2 is particularly useful for data analysis, since it can be applied to estimate the intrinsic dimension of real datasets that exhibit fractal behavior, i.e., exactly or statistically self-similar datasets. The Correlation Fractal Dimension D_2 measures the non-uniform behavior of real data considering both linear and nonlinear attribute correlations [de Sousa et al. 2007]. Therefore, D_2 represents the dimensionality of the dataset regardless of the dimension E of the space defined by its attributes. For instance, a set of points defining a line $z = ax + by + c$ embedded in a three-dimensional space with dimensions $[X; Y; Z]$ (and thus E = 3) has $D_2 = 1$, as there is a linear correlation between its attributes [de Sousa et al. 2007].

A method to measure the fractal dimension of datasets embedded in E-dimensional spaces is the Box-Counting method, which defines D_2 as:

$$D_2 = \frac{\partial log(\sum_i C_{r,i}^2)}{\partial log(r)} \quad r \in [r_1, r_2] \quad (4)$$

where r is the side of the cells in a (hyper) cubic grid that divides the address space of the dataset and $C_{r,i}$ is the count of points in the ith cell.

The work presented in [de Sousa et al. 2007] proposes a technique to detect changes in multidimensional, evolving data streams based on the information of intrinsic behavior provided by the fractal dimension D_2. The authors also present the algorithm SID-meter to continuously measure D_2 over time aimed at monitoring the evolving behavior of the data, such that significant variations in successive measures of D_2 can indicate changes in the intrinsic characteristics, as well as in attribute correlations in the data.

3. Fractal-based Clustering of Data Streams

Our goal is to partitioning the set $\mathbb{S}$, defined in Section 2, in a collection $P = \{C_1, \ldots, C_m\}$ of m disjoint clusters regarding to the fractal dimension analysis of each S_i. Each cluster C is composed of the data stream sources considered similar to each other if they do not exceed an user-defined maximum standard deviation parameter. Our proposal clusters data streams with a similar behavior in an interval of time T_i and also takes into account the correlation among the attributes measured by the fractal dimension D_2, calculated by Equation 4. Our method also follows the evolution of the data streams, where clusters can disappear or be created.

Fig. 2 shows the idea of data stream sources (e.g. sensors) generating new data continuously and send them to the proposed *Fractal-based Clustering of Data Streams* framework (FCDS), which produces evolutive clusters. Notice that for each interval of time T_i, the gathered data are clustered following the fractal dimension of the available data. As new information are received, the clusters are created or rearranged so as to ensure the similarity among their elements and the correlation among the attributes along the time. For instance, from period T_1 to T_2 it is possible to notice a cluster disappearance and from period T_2 to T_3 two new clusters was created.

Let us now illustrate the main components of our proposed framework. The process initiates when a defined number of data stream sources are chosen to be analyzed. As the sources are producing data, they can be directly forwarded to the FCDS framework in order to be processed, as shown in Fig. 3.

The input component of the FCDS is the Sliding Window Module. This module receives the data streams and bounds their information by a sliding window. The sliding window specifies the amount of data buffered for the fractal dimensional calculation. T he window is divided into counting periods (t), and each period has a defined number of events (e) such that each e represents $\vec{d}$ data points. Therefore, $t \times e$ defines the window size l and e also represents its movement step. The window took a default size l, which usually considers either domain experts or the seasonality of the data. But it is possible to define l with different values to achieve different granularities, allowing the incremental calculation of the fractal dimension D_2, as showed in Fig. 4.

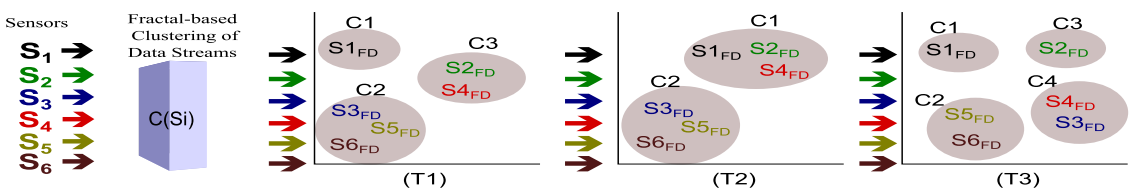


Figure 2. General idea of clustering data streams.

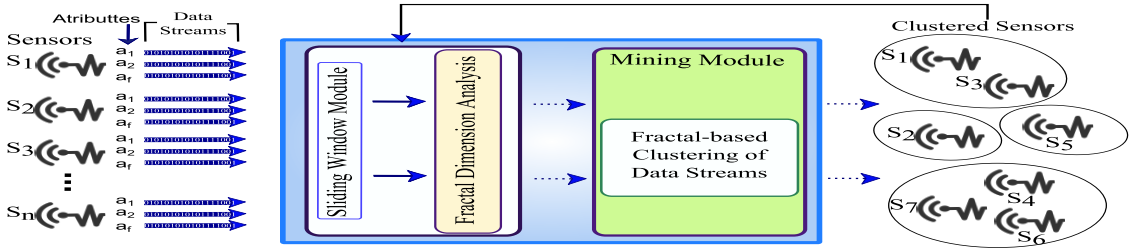


Figure 3. Method of clustering data streams.

Figure 4. Sliding Window of a sensor, counting periods $t = 4$, events $e = 3$.

As soon as there are enough information fulfilling the amount of events e, those data can be now unbuffered for their unique reading. In such case, the sliding window is shifted forward in e units and the region storing the already processed piece of information is released in order to store the continuous incoming data. Additionally, the previously windowed information is dispathed to the module responsible for performing the Fractal Dimension analysis.

The Fractal Dimension Analyzer considers an analog approach applied by SID-Meter [de Sousa et al. 2007] to perform the incremental calculus of D_2. Unlike the methods already proposed in the literature, this kind of technique enables to iterate over data containing more than one attribute. Thus, applying techniques such as the box-counting (Section 2.2), this module transforms a piece of multivariate data stream of size e, which represents the measures, to exactly one point of fractal correlation, summarizing that amount of data and the correlation among their attributes. Our intuition is that using a technique which correlates the involved attributes along the time leads to a better recognition of similar data stream sources than using the raw values of the attributes.

Subsequently, the reduced amount of fractal points is sent to the data mining step. The Mining Module is the main component of the FCDS framework and aims at clustering the correlated points according to their similarity and a user-defined maximum standard deviation (σ_{MAX}). In order to perform the clustering step, let us introduce the Algorithm 1.

The Algorithm 1 iterates over the set X comprising the points of fractal correlation (fd) such that each point is labeled with its respective data stream S_i. Initially, it is verified whether or not the data stream source S_i already belongs to some existing cluster C_j composing the partition P (line 2). Supposing S_i was not clustered yet, it is necessary to find for a cluster to insert S_i into. For this purpose, for each cluster $C \in P$, the boolean procedure `findCluster` (line 8) looks for the cluster C_j with the lowest difference between the centroid of C_j and the analyzed point x. This process follows the model

Algorithm 1: Fractal-based Clustering of Data Streams (FCDS)

Input: The set X containing pairs of streams S_i and its fractal dimension fd;
The partition P of clusters;
The maximum standard deviation σ_{MAX}.
Output: The partition P of clusters with similar data streams.

```
1  foreach x ∈ X do
2      if x.S_i is in a cluster C_j ∈ P then
3          if updateCluster(C_j, x, σ_MAX) = false then
4              if findCluster(x, σ_MAX) = false then
5                  C_new ← createNewCluster(x);
6                  rearrangeCluster(C_j, C_new);
7      else
8          if findCluster(x, σ_MAX) = false then
9              C_new ← createNewCluster(x);
10 return P;
```

introduced in Equation 5, where $C = \{c_1, \ldots, c_k, \ldots, c_n\}$.

$$\Delta(C, x) = \left(\frac{1}{n}\sum_{k=1}^{n} c_k.fd\right) - x.fd \tag{5}$$

Once the cluster C which minimizes the Equation 5 regarding to the element x was obtained, x is assigned to C iff the condition expressed in Equation 6 holds. Otherwise, the new cluster C_{new} containing the element x is created (line 9) and included in the partition P.

$$\sqrt{\frac{1}{n+1}\left(\left(x.fd - \left(\frac{1}{n}\sum_{p=1}^{n} c_p.fd\right)\right)^2 + \sum_{k=1}^{n}\left(c_k.fd - \left(\frac{1}{n}\sum_{p=1}^{n} c_p.fd\right)\right)^2\right)} \leq \sigma_{MAX} \tag{6}$$

Considering the data stream S_i is already part of some cluster, it is necessary to check whether or not the data stream source S_i remains in the previous assigned cluster. In order to employ such verification (line 3), the Mining module re-execute the calculus of Equation 6 regarding to the new value of the fractal dimension $x.fd$. If so, the statistic (function Δ) of the considered cluster C_j is updated. In the case where the left-hand side of Equation 6 exceeds the user-defined maximum standard deviation σ_{MAX}, the algorithm tries to reallocate the element x in an existing cluster C_k so as to minimize the value computed in Equation 5 (line 4) and also hold the condition defined in Equation 6. If there is not such cluster C_k satisfying both conditions, a new one (C_{new}) is created to include the element x (line 5). Now, when creating a new cluster to x, the elements already present in C_j are checked if they are better included in C_{new} (minimizing Equation 5) and inserted in it if they do (line 6). Notice that the `rearrangeCluster` procedure is able to suppress existing clusters if all of their elements are moved and they become empty.

At the end of one iteration over the set X, the partition P is composed of clusters containing the similar data streams sources in relation to the fractal correlation (line 10). As the sources are continuously generating measures, the sliding window shifts forward and, as soon as there are enough data to repeat the process, the Mining module is re-invoked. Therefore, the obtained clusters evolute to reflect the changes in the gathered information along the time.

4. Results

This paper reports a novel method to cluster data streams that behave similarly over the time, based on the correlation of their attributes by the calculus of fractal dimension. For this purpose, we introduced the FCDS framework.

In order to evaluate our proposal, we applied the following methodology: 1) test our approach using a real dataset; 2) assess the quality of the obtained groups by the use of the Silhouette measure; 3) apply the same methodology to the baselines methods, previously described in Section 2.1, in order to compare them with our approach. This methodology was used to analyze three key points: (i) the cohesion of the obtained partition of clusters, (ii) the ability of the FCDS framework to capture similar data streams behavior along the time and (iii) the quality of the generated clusters.

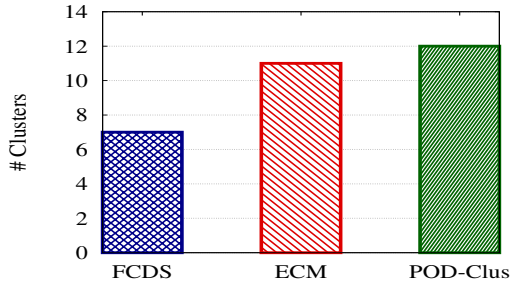
So, we employed a dataset composed of 145 climate sensors (sources of data streams) collecting 3 distinct daily measures (minimum—maximum temperature and precipitation) each one. The considered measures belong to the South and Southeast Brazilian regions, in a period from January 1990 to December 1994. This dataset was obtained in cooperation with the Embrapa Agriculture Informatics (Campinas, SP, Brazil).

In order to measure clusters' quality, we apply the well-known Silhouette. This is a measure commonly employed to evaluated how well the elements are disposed in their respective clusters. The values of Silhouette range from -1 to 1, such that clusters that obtain values above 0.5 are considered of good quality.

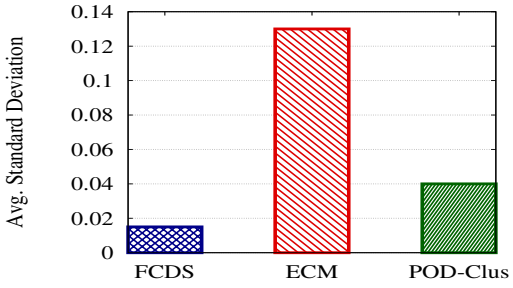
Finally, we compare our results with ECM and POD-Clus methods (both in Section 2.1) once they are the techniques presented in the literature aiming at solving the mainly issues: compact representation; fast data processing; traceability of changes; outliers identification. As mentioned in Section 2.1, only POD-Clus support multivariate data stream. But it only measure the dissimilarity of the attributes without considering the correlation among them. Therefore, in order to deal with the correlation among multiple attributes, presenting in the data streams, we calculate the fractal dimension to each sensor, creating in this way an unidimensional flow to use as input for ECM and POD-Clus. The parameter employed in ECM was correlation threshold equal 0.4. While the parameters for POD-Clus were outlier threshold equal 3 and boundary size equal 0.2. The main results are depicted in Fig. 5.

Fig. 5(a) presents the amount of clusters generated by each considered method. The proposed FCDS technique was able to identify the similarity among the data streams and partitioning them in a fewer number of clusters than the other algorithms. That result suggests our framework better summarized the same amount of information keeping the quality of the clusters, as can be seen in Fig. 5(b).

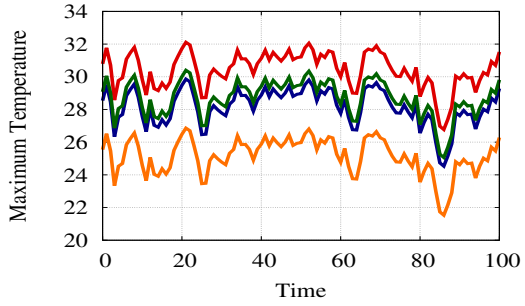
Fig. 5(b) shows the average of the standard deviations among the total number



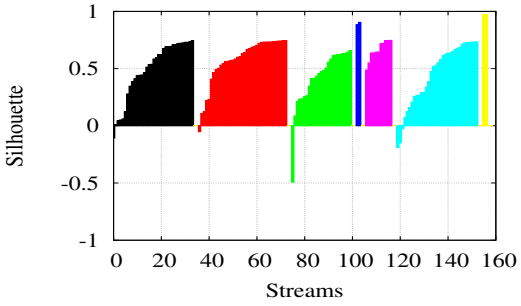
(a) Quantity of clusters generated by each method



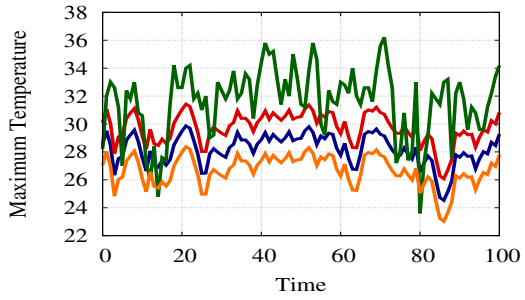
(b) Average of clusters standard deviation



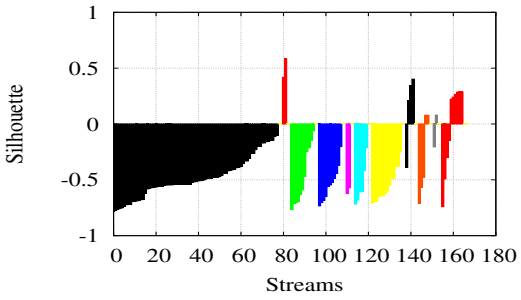
(c) Plot data streams raw data of the FCDS' cluster with the greatest standard deviation



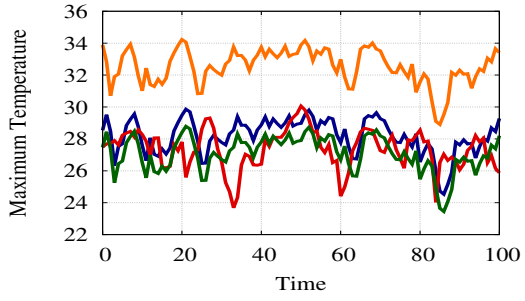
(d) Sillouette of FCDS' clusters



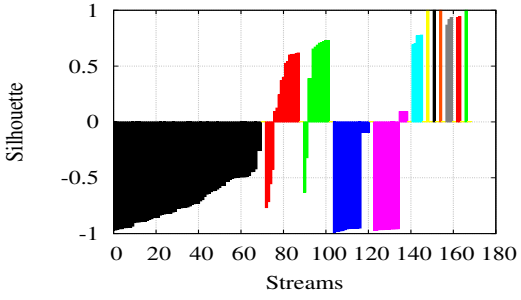
(e) Plot data streams raw data of the ECM' cluster with the greatest standard deviation



(f) Sillouette of ECM' clusters



(g) Plot data streams raw data of the POD-Clus' cluster with the greatest standard deviation



(h) Sillouette of POD-Clus' clusters

Figure 5. Comparative results

of clusters. In that study, the small is the standard deviation the better the elements are clustered. The average of the FCDS's clusters standard deviation was 88.46% less than the result of the ECM method and 62.50% regarding to the POD-Clus. It means that

clustering using FCDS, a data streams is belonging to a cluster that it probably should be in. In addition, it is important to highlight that the POD-Clus method achieved a low average standard deviation due to many generated clusters contain a single data stream, as will be seen further in Fig. 5(h).

Figs. 5(c), 5(e) and 5(g) illustrate the behavior found in the worst cluster (maximum standard deviation) generated by the each considered method. As sampled in Fig. 5(c), the data stream sources were correct included in the same cluster by the FCDS regarding to the correlation among their attributes. Unlike, the ECM method (Fig. 5(e)) was unsuccessful in correct capturing the correlation among the data streams, including streams in clusters which they are not part of. This fact can be confirmed by the two distinct behaviors observed in the same figure. Analogous, the POD-Clus method also misplace data streams in wrong clusters, recurring to the same behavior observed for the ECM method, as depicted in Fig. 5(g).

Fig. 5(d) shows the Sillhoute results obtained by the FCDS technique. The FCDS built clusters with 65% of the data streams Silhouette measures upper to 0.5. Although our method produced fewer clusters than the considered others, that measure indicates that small amount was good enough to represent the behavior of the analyzed dataset.

Fig. 5(f) presents the Silhouette measure to the clusters computed by the ECM method. In this analysis, only one stream was included in the correct cluster. Those results indicate the ECM method produced bad quality clusters when compared to FCDS.

Fig. 5(h) show the the Silhouette values to the clusters generated using the POD-Clus method. When compared with the ECM, the former achieve better results than the latter, correct clustering 29 data streams of a total amount of 145. However, POD-Clus in those 29 data streams, 4 were recognized to be outliers composing a single-element cluster each one. Thus, such behavior affected the standard deviation presented in Fig. 5(b).

Thus, considering the obtained results, the FCDS framework better represented the behavior of different data streams along the time. The approach based on fractals showed promising results to deal with multivariate data streams, helping to capture the behavior of climate streams over time and obtaining fewer clusters with good quality.

5. Conclusion and Future Work

Clustering of data streams is one of the most employed approach to analyze data that is potentially endless and evolves over the time. Nevertheless, the literature provides few methods for clustering the entire data streams. However, such proposals often deal with data streams composed of a single attribute or do not apply appropriate strategies for multivariate flows.

In this paper we introduced a framework to cluster a set of multivariate data streams considering the fractal correlation among their attributes, also complying the basic requirements to cluster data streams. It also takes into account the behavior of distinct streams along the time. The proposed Fractal-based Clustering of Data Streams framework is composed of minor modules, each one responsible for a specific processing of the data. The core of the FCDS framework lies in the computation of the fractal dimension of a piece of the data streams and its subsequent clustering. The use of the fractal dimension allowed to better identify the correlation among the attributes of the data streams. Also,

our method performs the clustering of multivariate data streams supporting their evolution in an incremental way, thereby helping to improve the quality of the clusters.

We performed a set of experimental evaluation of the FCDS framework and compared it againt two recent correlate methods, ECM and POD-Clus, in order to verify the capacity of representing similar data streams behaviors along the time and keeping the quality of the produced clusters.

As an ongoing direction, we intend to analyze other kind of data but the climate ones and improve the recognition of outliers so as to avoid clusters with a single element.

References

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2006). A framework for on-demand classification of evolving data streams. *IEEE TKDE*, 18(5):577–589.

Barbará, D. (2002). Requirements for clustering data streams. *ACM SIGKDD Explorations Newsletter*, 3(2):23–27.

Bifet, A. and De Francisci Morales, G. (2014). Big data stream learning with samoa. In *2014 IEEE Int. Conf on Data Mining Workshop*, pages 1199–1202.

Chaovalit, P. and Gangopadhyay, A. (2009). A method for clustering transient data streams. In *Proc. ACM SAC '09*, pages 1518–1519, New York, NY, USA. ACM.

de Sousa, E. P., Traina, A. J., Jr, C. T., and Faloutsos, C. (2007). Measuring evolving data streams behavior through their intrinsic dimension. *New Generation Computing*, 25(1):33–60.

Gama, J. (2010). *Knowledge Discovery from Data Streams*. Chapman and Hall Book, Boca Raton, Florida, USA.

Guha, S., Meyerson, A., Mishra, N., Motwani, R., and O'Callaghan, L. (2003). Clustering data streams: Theory and practice. *IEEE TKDE*, 15(3):515–528.

Kumar, M. and Patel, N. R. (2007). Clustering data with measurement errors. *Computational Statistics & Data Analysis*, 51(12):6084–6101.

Miller, Z., Dickinson, B., Deitrick, W., Hu, W., and Wang, A. H. (2014). Twitter spammer detection using data stream clustering. *Information Sciences*, 260:64–73.

Pereira, C. M. M. and de Mello, R. F. (2014). Ts-stream: clustering time series on data streams. *Journal of Intelligent Information Systems*, 42(3):531–566.

Rehman, M. Z., Li, T., Yang, Y., and Wang, H. (2014). Hyper-ellipsoidal clustering technique for evolving data stream. *Knowledge-Based Systems*, 70(0):3–14.

Rodrigues, P. P., Gama, J., and Pedroso, J. P. (2008). Hierarchical clustering of time-series data streams. *IEEE TKDE*, 20(5):615–627.

Widiputra, H., Pears, R., and Kasabov, N. (2011). Multiple time-series prediction through multiple time-series relationships profiling and clustered recurring trends. In Huang, J., Cao, L., and Srivastava, J., editors, *Advances in Knowledge Discovery and Data Mining*, pages 161–172. Springer, Shenzhen, China.

Zhang, X., Furtlehner, C., Germain-Renaud, C., and Sebag, M. (2014). Data stream clustering with affinity propagation. *IEEE TKDE*, 26(7):1644–1656.

sbbd:04

Consulta Espacial Preferencial por Palavra-chave

João Paulo Dias de Almeida e João B. Rocha-Junior

[1]Pós-graduação em Computação Aplicada
Universidade Estadual de Feira de Santana
Feira de Santana - Bahia - Brasil

jpalmeida.uefs@gmail.com, joao@uefs.br

Abstract. *With the popularity of devices that are able to annotate data with spatial information (latitude and longitude), the processing of spatial queries has received a lot of attention from the research community recently. In this paper, we study a new query type named Top-k Spatial Keyword Preference Query that selects objects of interest based on the textual relevance of other spatio-textual objects in their spatial neighborhood. This paper introduces this new query type, presents three algorithms for processing the query efficiently and performs an experimental evaluation to study the performance of the algorithms proposed.*

Resumo. *Com a popularidade de dispositivos capazes de anotar dados com coordenadas espaciais (latitude e longitude), o processamento de consultas espaciais tem recebido bastante atenção da comunidade científica recentemente. Este artigo apresenta uma nova consulta, chamada Consulta Espacial Preferencial por Palavra-chave, que seleciona objetos de interesse de acordo com a relevância textual de outros objetos espaço-textuais presentes na sua vizinhança espacial. Este artigo introduz esta nova consulta, apresenta três algoritmos para processá-la de forma eficiente e avalia o desempenho dos algoritmos propostos através de um estudo experimental, utilizando bases de dados reais.*

1. Introdução

O volume de dados espaciais tem crescido significativamente nos últimos anos, o que explica o interesse por novas técnicas capazes de processar esses dados de forma eficiente [Cao et al. 2012]. Facebook, Google Maps, Twitter e Waze são exemplos de aplicações que processam informações espaciais para fornecer serviços úteis para o usuário.

A maioria desses dados espaciais estão associados a uma informação textual. Por exemplo, algumas mensagens do Twitter (texto) enviados a partir de *Smartphones* possuem uma coordenada espacial (latitude e longitude). Uma grande parcela dos mais de 4 bilhões de objetos espaciais armazenados no OpenStreetMap (www.osm.org) estão associados a um texto descritivo. Estes objetos que possuem informações espaciais e textuais são comumente referenciados como objetos espaço-textuais [Vaid et al. 2005].

Para processar consultas de forma eficiente em grandes bases de dados compostas por objetos espaço-textuais, faz-se necessário utilizar índices híbridos [Chen et al. 2013, Cong et al. 2009, Rocha-Junior et al. 2011]. Estes índices combinam métodos de acesso espaciais, como R*-tree [Beckmann et al. 1990]; e textuais, como os Arquivos Invertidos [Zobel and Moffat 2006], para processar consultas de forma eficiente.

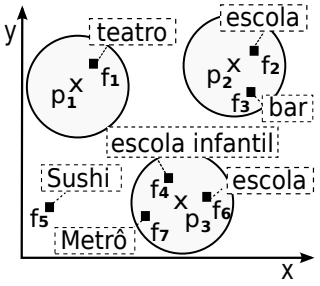


Figura 1. Objetos (p) e objetos de referência (f) com um texto.

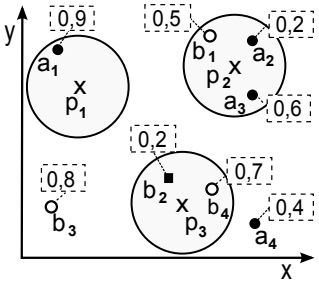


Figura 2. Objetos (p) e objetos de referência (a e b) com um escore.

O objetivo deste artigo é apresentar uma nova consulta para extrair informações de bases de dados espaço-textuais, a consulta Espacial Preferencial por Palavra-chave (EPPC). Diferente da consulta espaço-textual tradicional [Cong et al. 2009, Rocha-Junior et al. 2011] que retorna objetos próximos a uma dada localização; a consulta EPPC retorna objetos de interesse com base em outros objetos espaço-textuais presentes em sua vizinhança espacial.

Dado um conjunto de *objetos espaciais de interesse* (ex: hotéis), um conjunto de *objetos espaço-textuais de referência* (ex: bares, restaurantes e shoppings), um *critério de vizinhança espacial* (ex: 100m de distância dos objetos de interesse) e um *conjunto de palavras-chaves de busca* (ex: "cantina italiana"); a consulta EPPC retorna os k melhores objetos espaciais de interesse, onde o escore de cada objeto de interesse é a maior relevância textual (similaridade textual para as palavras-chaves de busca) entre os objetos espaço-textuais de referência que atendem ao critério de vizinhança espacial.

Exemplo. A Figura 1 apresenta um espaço contendo objetos espaciais de interesse p (ex. apartamentos) e objetos espaço-textuais de referência f (ex. estabelecimentos diversos). O círculo em volta de cada objeto de interesse representa o critério de vizinhança espacial. Assim, um usuário interessado em alugar um apartamento próximo a um objeto relevante para as palavras-chave "escola" e "infantil", teria como retorno os objetos p_2 e p_3, sendo p_3 o objeto mais relevante (top-1), visto que ele possui na sua vizinhança espacial, um objeto (f_4) mais relevante para as palavras-chave de busca que p_2.

Várias aplicações podem se beneficiar desta consulta para prover serviços de localização mais avançados para os usuários. Por exemplo, uma aplicação de turismo pode retornar os k melhores pontos de metrô (objetos de interesse) próximos a estabelecimentos (objetos espaço-textuais de referência) relevantes para o conjunto de palavras-chave "museu de arte a céu aberto", facilitando assim, a locomoção do usuário. Aplicações governamentais também podem se beneficiar. Por exemplo, dado um conjunto de ruas (objetos de interesse), e tweets (objetos espaço-textuais de referência) enviados pelos cidadãos a partir de seus *Smartphones*, é possível identificar ruas com maior incidência de criminalidade utilizando de palavras-chaves como "roubo, assalto e tiro".

Infelizmente, processar a consulta EPPC de forma eficiente em grandes bases de dados é complexo. Uma forma trivial para processar essa consulta requer selecionar todos os objetos espaço-textuais presentes na vizinhança espacial de cada objeto de interesse, computar a relevância textual de cada um destes objetos para as palavras-chave de busca, atribuir o escore ao objeto de interesse, repetindo este processo para todos os objetos

de interesse até obter os k melhores. Em bases de dados grandes, este método é bastante custoso. As principais contribuições deste trabalho são: 1) propor a consulta EPPC; 2) apresentar algoritmos para processar esta consulta de forma eficiente e 3) avaliar os algoritmos propostos através de experimentos em bases de dados reais.

O restante deste artigo é organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 contém uma especificação da consulta EPPC; a Seção 4 apresenta os algoritmos para processar a consulta EPPC; os experimentos realizados e resultados obtidos são discutidos na Seção 5; e a Seção 6 faz as considerações finais.

2. Trabalhos Relacionados

A pesquisa na área de consultas espaço-textuais pode ser dividida em três etapas. Na primeira etapa, as consultas tinham como objetivo recuperar documentos relevantes para as palavras-chaves de busca, utilizando a informação da localização para filtrar documentos irrelevantes [Zhou et al. 2005]. Por exemplo, para responder à consulta "padaria na Barra da Tijuca", verificava-se quais documentos eram relevantes sobre padarias no bairro Barra da Tijuca, utilizando técnicas de Recuperação de Informação.

Na segunda etapa, com a popularização do GPS, o foco passou a ser em identificar objetos espaciais relevantes para as palavras-chave. As pesquisas não focaram na relevância textual dos objetos e sim na busca booleana, indicando se eles tinham as palavras-chaves ou não [Felipe et al. 2008]. Por exemplo, na consulta "retorne todos os objetos que estão a 100m de distância da coordenada (x,y) e que são relevantes para as palavras 'praia flamengo"', os objetos espaço-textuais que tem ambos os termos "praia" e "flamengo" são retornados como igualmente relevantes, todos os demais são descartados.

Na terceira etapa (atual), o foco passou a ser na localização precisa dos objetos espaço-textuais e na relevância textual dos mesmos para as palavras-chave de busca. As consultas passaram a computar a relevância textual precisa dos objetos espaço-textuais, retornando apenas os mais relevantes. A principal consulta é a Espacial por Palavra-chave [Cong et al. 2009, Rocha-Junior et al. 2011], que dada uma localização (latitude e longitude) e um conjunto de palavras-chave, ela retorna os k objetos espaço-textuais mais relevantes, levando-se em consideração a distância entre a localização da consulta e os objetos, e a relevância textual dos objetos para as palavras-chaves de busca.

A consulta EPPC foi inspirada na consulta Espacial Preferencial tradicional [Yiu et al. 2007]. Dado um conjunto de objetos espaciais de interesse P, um conjunto de objetos espaciais de referência F_i e um critério de vizinhança espacial; a consulta Espacial Preferencial tradicional retorna os k melhores objetos $p \in P$, onde o escore de cada objeto p é o maior escore entre os objetos de referência $f \in F_i$ que atendem ao critério de vizinhança espacial. O escore de cada objeto de referência é dado a priori.

Exemplo. A Figura 2 apresenta um espaço contendo objetos espaciais de interesse p (ex. hotéis) e objetos espaciais de referência a (ex. cafés) e b (ex. bares). Os objetos de referência estão associados a um escore pré-definido. O círculo em volta de cada objeto de interesse representa o critério de vizinhança espacial. Assim, um usuário interessado em um hotel próximo a um bom café, terá como retorno os objetos p_1 e p_2, sendo p_1 melhor que p_2, porque o objeto de referencia a_1 presente na vizinhança espacial de p_1 tem um escore maior que a_3 (melhor escore entre os restaurantes na vizinhança de $p2$).

Na consulta Espacial Preferencial tradicional, o escore dos objetos espaciais de referência é dado a priori, assim é possível construir índices que materializam parte dos resultados e podem responder a essa consulta de forma eficiente [Rocha-Junior et al. 2010]. Na consulta EPPC, o escore dos objetos de referência não é dado a priori. O escore dos objetos espaciais de referência é computado no momento da consulta, a partir da relevância textual (similaridade) entre as palavras-chaves de busca e o texto associado a cada objeto espaço-textual, tornando o processamento eficiente desta consulta mais complexo.

Recentemente, Tsatsanifos e Vlachou [Tsatsanifos and Vlachou 2015] apresentaram uma consulta similar a EPPC. Entretanto, esta consulta não considera que o escore dos objetos é computado puramente pela relevância textual entre as palavras-chaves de busca e o texto associado aos objetos de referência. Além da informação textual, os objetos possuem um escore pré-definido (similar a consulta Espacial Preferencial tradicional) e os algoritmos propostos se beneficiando deste escore para processar a consulta.

3. Especificação da Consulta Espacial Preferencial por Palavra-chave

A consulta *Espacial Preferencial por Palavra-chave* (EPPC) Q é formada por um conjunto de palavras-chave $Q.D$, pela definição da vizinhança espacial de interesse $Q.\psi$, e pela quantidade $Q.k$ de objetos de interesse que se deseja obter como resposta: $Q = (Q.D, Q.\psi, Q.k)$.

Dado um conjunto de objetos de interesse P, onde cada objeto $p \in P$ possui uma coordenada espacial $p = (p.x, p.y)$; e um conjunto de objetos espaço-textuais de referência F, onde cada objeto $f \in F$ possui uma coordenada espacial $(f.x, f.y)$ e um texto $f.D$, $f = (f.x, f.y, f.D)$. A consulta Q retorna os k objetos de interesse contidos em P que possuem os maiores escores. O escore $\tau^{Q.\psi}(p)$ de um objeto de interesse $p \in P$ é a maior relevância textual (similaridade textual) entre os objetos espaço-textuais de referência f que atendem ao critério de vizinhança espacial $Q.\psi$..

O usuário pode optar entre três formas possíveis de especificar o critério de vizinhança espacial $Q.\psi$: seleção espacial ($Q.\psi = rng$), vizinho mais próximo ($Q.\psi = nn$) ou influência ($Q.\psi = inf$) [Yiu et al. 2007].

- Seleção espacial $\tau^{rng}(p)$, dado um raio r:

$$\tau^{rng}(p) = max\ \{\ \theta(f.D, Q.D) \mid f \in F : dist(p, f) <= r\ \}$$

- Vizinho mais próximo $\tau^{nn}(p)$:

$$\tau^{nn}(p) = max\ \{\ \theta(f.D, Q.D) \mid f \in F, \theta(f.D, Q.D) > 0, \forall\ v \in F : dist(p, f) \leq dist(p, v)\ \}$$

- Influência $\tau^{inf}(p)$, dado um raio r:

$$\tau^{inf}(p) = max\ \{\ \theta(f.D, Q.D) \cdot 2^{-dist(p,f)/r} \mid f \in F\ \}$$

onde $\theta(f.D, Q.D)$ é a função cosseno que retorna a relevância textual (similaridade textual) entre o texto do objeto de referência $f.D$ e o conjunto de palavras-chave $Q.D$ [Zobel and Moffat 2006, Rocha-Junior et al. 2011], enquanto $dist(p, f)$ é a distância Euclidiana entre um objeto p e um estabelecimento f.

Quando o usuário opta pela *seleção espacial* ($Q.\psi = rng$), o escore de um objeto de interesse $p \in P$ é definido pela maior relevância textual $\theta(f.D, Q.D)$ entre os objetos $f \in F$ que atendem ao critério de vizinhança espacial, ($dist(p, f) \leq r$). Ao optar por *vizinho mais próximo* ($Q.\psi = nn$), o escore é definido pelo objeto de referência mais próximo de p que seja textualmente relevante $\theta(f.D, Q.D) > 0$. Caso existam vários objetos de referência f com a mesma proximidade espacial do objeto de interesse p, o escore de p é a maior relevância textual $\theta(f.D, Q.D)$ entre os objetos f que possuem a mesma menor distância. O critério de vizinhança *influência* ($Q.\psi = inf$) define o escore do objeto de interesse p levando em consideração a relevância textual entre $f.D$ e $Q.D$ e a distância entre p e f, quanto maior a distância menor o escore (influência).

Todos os algoritmos apresentados neste artigo podem ser adaptados para atender os três critérios de vizinhança espacial. Por questões de espaço, optou-se por apresentar os algoritmos considerando apenas a *seleção espacial* (*rng*).

4. Algoritmos propostos para processar a consulta EPPC

Nesta seção são apresentados três algoritmos para processar a consulta EPPC. Em todos os algoritmos, os objetos de interesse P estão armazenados em uma R*-tree [Beckmann et al. 1990]. O primeiro algoritmo (IFA) utiliza o Arquivo Invertido adaptado para indexar o texto dos objetos de referência F. Os outros dois algoritmos (SIA e SIA$^+$), utilizam o índice espaço-textual S2I [Rocha-Junior et al. 2011] para indexar os objetos de referência F.

Arquivo Invertido adaptado. Um arquivo invertido [Zobel and Moffat 2006] armazena, para cada termo do vocabulário, uma lista com os ids dos objetos (documentos) que possuem o termo e o *impacto do termo*[1] para o objeto. Dessa forma, dado um termo, é possível localizar a lista de objetos que contém esse termo rapidamente e calcular o escore textual do objeto. Para computar o critério de vizinhança espacial, optou-se por armazenar em cada tupla da lista, além do id do objeto e do impacto do termo, as coordenadas espaciais do objeto. Assim, quando uma tupla é retornada, é possível verificar se o objeto atende ao critério de vizinhança espacial.

S2I. O S2I [Rocha-Junior et al. 2011] é um dos principais índices espaço-textuais [Chen et al. 2013]. O S2I funciona de maneira semelhante a um Arquivo Invertido. Para cada termo, o S2I armazena tuplas contendo id dos objetos que contêm o termo e o impacto do termo. As tuplas dos termos mais frequentes são armazenados em um índice espacial (aR-tree [Papadias et al. 2001]), enquanto que as tuplas dos termos menos frequentes são armazenados em uma lista (similar a lista utilizada pelo Arquivo Invertido modificado). Sendo assim, para os termos mais frequentes, é possível selecionar os objetos que atendem ao critério de vizinhança espacial de forma eficiente.

4.1. *Inverted File Based Algorithm*

O *Inverted File Based Algorithm* (IFA) utiliza um Arquivo Invertido adaptado para localizar os k melhores objetos de interesse. O escore de um objeto de interesse p é o

[1] O impacto do termo reflete a importância do termo para o documento, em geral, quanto maior a frequência do termo maior o impacto. Em uma consulta com mais de um termo, a relevância textual do documento é obtida somando o impacto para cada termo [Zobel and Moffat 2006, Rocha-Junior et al. 2011].

maior escore textual $f.\theta$ entre os objetos de referência $f \in F$ relevantes para as palavras-chave de busca $\theta(f.D, Q.D) > 0$, cuja distância para p seja menor ou igual que o raio $Q.r$, $dist(p, f) \leq Q.r$. Para identificar os objetos de referência f relevantes para as palavras-chaves $Q.D$, é necessário recuperar as listas invertidas de cada termo $t \in Q.D$ ($IF.list(t)$). Estas listas são acessadas em ordem crescente de id do objeto de referência f, facilitando o cálculo do escore textual de f.

Os objetos f são retirados da lista seguindo o padrão iterador ($IF.list(t).next()$). Ao retirar um objeto de referência f de uma lista invertida para t, o escore textual dele só reflete o escore para o termo t (impacto do termo t em f). Caso esse mesmo objeto f, seja localizado em outra lista de um termo $t' \neq t$ e $t' \in Q.D$, o escore textual de f deve ser acrescido do escore parcial de t'. Em outras palavras, o escore textual de um objeto de referência é a soma dos escores parciais de todos os termos da consulta que estão presentes na descrição textual do objeto de referência. Após computar o escore textual de um objeto de referência f, o algoritmo verifica se f atende ao critério de vizinhança espacial ($dist(p, f) \leq Q.r$) e se o escore atual de p é menor que o escore textual de f ($p.score < f.\theta$), atualizando o escore de p com o escore de f em caso afirmativo.

O Algoritmo 1 apresenta o IFA. O algoritmo computa o escore de cada objeto $p \in P$ (linhas 3-27), inicialmente o escore de p é zero (linha 4). O algoritmo recupera um iterador (linha 6) que tem acesso a todos os objetos da lista invertida do termo t, em ordem crescente de id. Então o algoritmo armazena, em uma *heap* H, uma entrada e composta por dois atributos $e.f$ e $e.l$ ($e = \{e.f, e.l\}$). O atributo $e.f$ é o primeiro objeto da lista t ($iterator.next()$), e o atributo $e.l$ é uma referência para o iterador da lista ($iterator$) (linhas 7-9). O objetivo da *heap* H é armazenar, para cada lista de um termo t, o objeto de referência f com menor id ($e.f$) que ainda não tenha sido visitando e um ponteiro para acessar os demais itens da lista (iterador, $e.l$), permitindo localizar de forma eficiente os objetos de referência que estão armazenadas em mais de uma lista.

O algoritmo continua retirando a entrada da lista cujo o objeto de referência $e.f$ tenha o menor id (linha 11). A função $nextEntry(H)$ retira uma entrada da *heap* ($e \leftarrow heap.pollFirst()$) e armazena na *heap* uma nova entrada $\{e.l.next(), e.l\}$ composta pelo próximo objeto de referência da lista ($e.l.next()$) e pelo ponteiro para o iterador da lista ($e.l$), caso a lista ainda tenha objetos de referência a serem visitados. Enquanto e for diferente de $null$ e enquanto a *heap* H não estiver vazia (linhas 12-20), o algoritmo computa o escore de p. Para computar o escore de p, é preciso computar o escore textual do objeto de referência f, somando os escores parciais de f em cada lista que ele aparece. Como as listas estão ordenadas pelo id dos objetos de referência, basta verificar se o objeto de referência está presente no topo de mais de uma lista (linhas 14-17).

Por exemplo, em uma consulta por dois termos t_1 e t_2, um objeto de referência f_1 que tenha t_1 e t_2 na sua descrição textual vai aparecer nas duas listas invertidas: $IF.list(t_1)$ e $IF.list(t_2)$. Como os objetos das listas estão ordenados em ordem crescente de id, f_1 (cujo $id = 1$) vai estar no topo das duas listas. Sendo assim, basta retirar f_1 das duas listas e computar o escore textual de f_1 somando os escores parciais que f_1 tem em cada lista. Se a descrição textual de f_1 tivesse apenas o termo t_1, f_1 não estaria na lista t_2 e o escore textual dele seria composto apenas pelo escore parcial obtido através da lista invertida de t_1.

Algoritmo 1: *Inverted File Based Algorithm* (IFA)

```
   Input: Q = (Q.D, Q.r, Q.k) //O critério de vizinhança rng é o raio Q.r.
   Output: Heap contendo os k melhores objetos de interesse.
1  M ← ∅ //Heap contendo os k melhores objetos de interesse
2  H ← ∅ //Heap que mantém as entradas e ordenadas pelo id do objeto de referência
3  for each p ∈ P do
4      p.score ← 0
5      for each t ∈ Q.D do
6          iterator ← IF.list(t).iterator()
7          if iterator.hasNext() then
8              H.add({iterator.next(), iterator})
9          end
10     end
11     e ← nextEntry(H)
12     while e ≠ null AND H ≠ ∅ do
13         e' ← nextEntry(H)
14         while e' ≠ null AND e.f = e'.f do
15             e.f.θ ← e.f.θ + e'.f.θ e' ← nextEntry(H)
16         end
17         updateScore(p, e.f)
18         e ← e'
19     end
20     updateScore(p, e.f)
21     if |M| < k OR p.score > M.peekMin().score then
22         M.add(p)
23         if |M| > k then
24             M.removeMin()
25         end
26     end
27 end
28 return M
```

Uma vez que o escore textual do objeto de referência tenha sido computado, utiliza-se a função *updateScore(p, e.f)* para atualizar o escore de p. Esta função verifica o critério de vizinhança espacial, checando se a distância entre p e f é menor ou igual que o raio $Q.r$ e se o escore atual de p é menor que o escore textual de f ($p.score < f.\theta$), atualizando o escore de p em caso afirmativo. Após computar o escore de p, o algoritmo atualiza a *heap* M que armazena os melhores objetos (linhas 22-27), adicionando p a M, caso o tamanho de M ($|M|$) seja menor do que k ou o escore de p seja maior que o escore do objeto em M com menor escore (linha 22). O algoritmo remove o objeto de M com menor escore caso o tamanho de M supere k (linhas 24-26). O algoritmo repete até que a *heap* H esteja vazia ou $e = null$, retornando os k melhores objetos armazenados em M.

4.2. *Spatially Indexed Algorithm*

O *Spatially Indexed Algorithm (SIA)* utiliza um índice híbrido S2I ao invés de um arquivo invertido. Assim como o IFA, o SIA computa o escore de cada objeto de interesse $p \in P$ para obter os k melhores objetos. Entretanto, ao invés de obter uma lista com os objetos de referência que possuem um termo da consulta $Q.D$, o S2I permite selecionar os objetos de referência que são textualmente e espacialmente relevantes de forma mais eficiente.

Algoritmo 2: *Spatially Indexed Algorithm* (SIA)

Input: $Q = (Q.D, Q.r, Q.k)$ //O critério de vizinhança rng é o raio $Q.r$.
Output: Heap contendo os k melhores objetos de interesse.
1 $M \leftarrow \emptyset$ //*Heap* contendo os k melhores objetos de interesse
2 $H \leftarrow \emptyset$ //*Heap* que mantém as entradas e ordenadas pelo id do objeto de referência
3 **for *each*** $p \in P$ **do**
4 $\quad p.score \leftarrow 0$
5 $\quad$ **for *each*** $t \in Q.D$ **do**
6 $\quad\quad iterator \leftarrow S2I.search(t, p.x, p.y, Q.r)$ *linhas 7-9 do algoritmo IFA*
7 $\quad$ **end**
8 $\quad$ *linhas 11-27 do algoritmo IFA*
9 **end**
10 **return** M

O Algoritmo 2 apresenta o SIA. O algoritmo começa inicializando o escore de cada objeto p com zero (linha 4). Para cada termo t da consulta ($t \in Q.D$), o algoritmo acessa o $S2I$ para recuperar um iterador que acessa os objetos de referência relevantes para t (linha 6), ordenados em ordem crescente de id. Entretanto, ao invés de retornar todos os objetos textualmente relevantes como o IF, o $S2I$ retorna apenas os objetos f que são textualmente relevantes e cujo a distância para p seja menor ou igual que o raio $Q.r$ ($S2I.search(t, p.x, p.y, Q.r)$) (linha 6). Se o termo t é frequente, os objetos de referência f que possuem o termo t na descrição textual estão armazenados em um índice espacial (aR-tree [Papadias et al. 2001]), que permite selecionar os objetos relevantes para os critérios de vizinhança espacial da consulta de forma eficiente. O restante do algoritmo SIA é idêntico ao IFA.

4.3. *Optimized Spatially Indexed Algorithm*

O IFA e o SIA computam o escore de cada objeto de interesse p em P, mantendo os k objetos com maiores escores em uma *heap* M. O SIA^+ é uma variação do SIA, que ao invés de computar o escore de cada objeto de interesse separadamente, ele calcula o escore de um conjunto V de objetos de interesse concorrentemente. Os objetos $p \in P$ escolhidos para compor um conjunto V são objetos espacialmente próximos. Por questões de simplicidade, optou-se por computar concorrentemente os objetos $p \in P$ que estão armazenados na mesma folha da R*-tree [Beckmann et al. 1990] que armazena os objetos P[2]. O objetivo do SIA^+ é reduzir o I/O, percorrendo o S2I apenas uma vez para calcular o escore de todos objetos armazenados em V.

O Algoritmo 3 apresenta o *Optimized Spatially Indexed Algorithm* (SIA$^+$). Para cada $V \in P$ onde V é composto por um conjunto de objetos espacialmente próximos, o algoritmo inicializa todos os escores dos objetos $p \in V$ com zero (linha 4). O algoritmo então acessa o S2I para pegar apenas os objetos de referência que podem contribuir com o cálculo do escore de um objeto $p \in V$ (linha 6). Todo objeto de referência f cuja distância para o MBR (*Minimum Bounding Rectangle*), retângulo mínimo que engloba todos os objetos espaciais $p \in V$, seja inferior ou igual a $Q.r$ podem colaborar com o escore de um objeto $p \in V$, visto que f pode atender ao critério de vizinhança de p. As linhas 9-18 são idênticas ao IFA e ao SIA, exceto as linhas 16 e 18, que ao invés de computar o escore

[2]Uma outra solução seria utilizar um algoritmo de *cluster* para agrupar os objetos próximos espacialmente, onde cada V seria composto pelos objetos armazenados no mesmo *cluster*.

Algoritmo 3: *Optimized Spatially Indexed Algorithm* (SIA+)

Input: $Q = (Q.D, Q.r, Q.k)$ //O critério de vizinhança rng é o raio $Q.r$.
Output: Heap contendo os k melhores objetos de interesse.

```
1  M ← ∅ //Heap contendo os k melhores objetos de interesse
2  H ← ∅ //Heap que mantém as entradas e ordenadas pelo id do objeto de referência
3  for each V ∈ P do
4      ∀p ∈ V, p.score ← 0
5      for each t ∈ Q.D do
6          iterator ← S2I.search(t, V.MBR, Q.r)
7          linhas 7-9 do algoritmo IFA
8      end
9      e ← nextEntry(H)
10     while e ≠ null AND H ≠ ∅ do
11         e' ← nextEntry(H)
12         while e' ≠ null AND e.f = e'.f do
13             e.f.θ ← e.f.θ + e'.f.θ
14             e' ← nextEntry(H)
15         end
16         updateScore(V, e.f) e ← e'
17     end
18     updateScore(V, e.f)
19     for each p ∈ V do
20         if |M| < k OR p.score > M.peakMin().score then
21             M.add(p)
22             if |M| > k then
23                 M.removeMin()
24             end
25         end
26     end
27 end
28 return M
```

de um único objeto p, o SIA$^+$ computa o escore de todos os objetos $p \in V$. O mesmo ocorre nas linhas 19-26, onde o SIA$^+$ precisa verificar, para cada $p \in V$, se p pode ser inserido na *heap* M que mantém os k melhores objetos.

5. Avaliação Experimental

Esta seção faz uma avaliação dos algoritmos propostos (IFA, SIA e SIA$^+$) para processar a consulta EPPC. Todos os algoritmos foram implementados em Java, utilizando a biblioteca XXL[3]. Todos os experimentos descritos nesta seção foram executados em um computador com processador Intel: 2.0 GHz e 4 GB de memória.

Cada experimento avalia o impacto de uma única variável, enquanto as outras são mantidas fixas. Em cada experimento, o tempo de resposta e o I/O (*page fault*) são coletados. Para cada experimento foram realizadas 200 consultas EPPC. As palavras-chaves das consultas são obtidas pegando termos aleatoriamente e sem repetição entre os 500 termos mais frequentes de cada base de dados. O raio para todas as consultas foi fixado em aproximadamente[4] 200m.

[3]http://dbs.mathematik.uni-marburg.de/Home/Research/Projects/XXL

[4]A função utilizada para calcular a distância não leva em consideração a inclinação do planeta terra. A mesma função foi utilizada em todos os algoritmos, portanto não interfere nos resultados obtidos.

Parâmetros	Valores
Número de resultados (k)	1, **5**, 10, 15
Palavras-chave	1, **3**, 5, 7
conjunto de dados	São Paulo, **Veneza**, Londres

Tabela 1. Variáveis estudadas com os padrões estão destacados em negrito.

A Tabela 1 apresenta as variáveis estudadas. Os valores em negrito são os valores utilizados como padrão. Todas as figuras desta seção são apresentadas em escala logarítmica devido a grande diferença entre os resultados apresentados pelos algoritmos.

5.1. Bases de dados

As bases de dados espaço-textuais utilizadas nos experimentos foram obtidas através do Mapzen[5] no dia 22/05/2015. O Mapzen mantém dados de diversas regiões do planeta, coletados do OpenStreetMap[6]. Para este artigo, foram selecionados objetos espaço-textuais das cidades de São Paulo, Veneza e Londres. Os objetos espaço-textuais, em cada uma das bases, foi separado em conjunto de objetos de interesse P e conjunto de objetos de referência F. Os objetos $p \in P$ são todos os objetos espaço-textuais cuja categoria no OpenStreetMap é *hotel*.

Bases de dados	$\|P\|$	$\|F\|$	#termos únicos	#total de termos
São Paulo	127	87418	21653	252404
Veneza	504	167958	14678	408747
Londres	1341	463066	56569	1198649

Tabela 2. Características das bases de dados.

A Tabela 5.1 apresenta as características de cada uma das bases de dados: número de objetos de interesse $|P|$; número de objetos de referência, $|F|$; número de termos únicos (#termos únicos) que aparecem no vocabulário (coleção) e o número total de termos (#número total de termos).

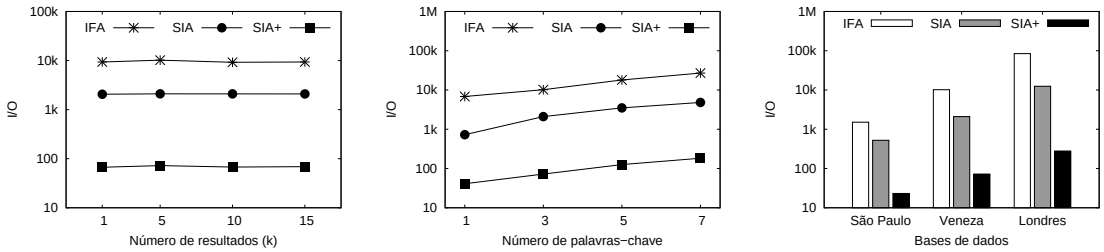
5.2. Estudo sobre acesso a dados em arquivo

Esta seção avalia o número de páginas lidas em disco por cada algoritmo. Todos os algoritmos foram implementados para ler páginas de 4MB e não utilizam cache de páginas em memória. A Figura 3 apresenta o número de páginas lidas (I/O) em diferentes cenários.

A Figura 3(a) apresenta o número de páginas lidas, para diferentes valores do número de resultados solicitados (k). O SIA e o SIA$^+$ acessam menos páginas do que o IFA, o que demonstra a importância em utilizar um índice híbrido para diminuir o acesso a objetos que não atendem ao critério de vizinhança espacial. O SIA$^+$ apresenta os melhores resultados, visto que ele acessa o índice uma única vez para computar o escore de um conjunto de objetos concorrentemente.

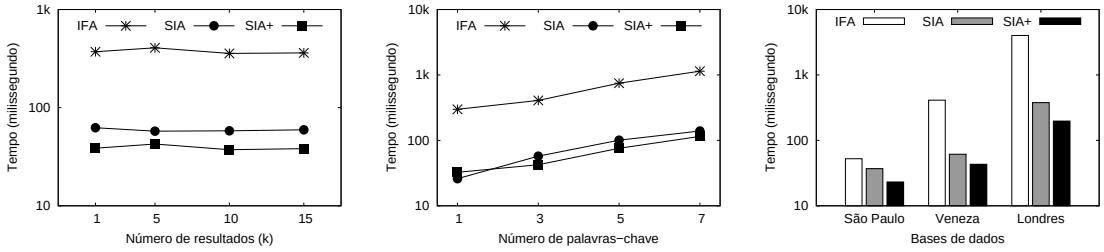
A Figura 3(b) apresenta o número de páginas lidas, variando o número de palavras-chave por consulta. Todos os algoritmos são impactados por essa variável, visto

[5]http://mapzen.com/metro-extracts
[6]http://www.osm.org.



(a) Número de Resultados (k). (b) Número de palavras-chave (c) Variando a cardinalidade

Figura 3. Resultados dos algoritmos em números de páginas lidas.



(a) Número de Resultados (k). (b) Número de palavras-chave. (c) Variando a cardinalidade

Figura 4. Resultado dos algoritmos em tempo de resposta.

que aumentar um termo significa acessar mais dados no momento da consulta. Entretanto, existe uma diferença clara em favor do SIA$^+$, que acessa 2 ordens de magnitude menos páginas que o IFA.

Finalmente, a Figura 3(c) apresenta o I/O para três diferentes bases de dados. Comprovando que os resultados são consistentes mesmo em bases de dados diferentes e de diferentes tamanhos.

5.3. Estudo sobre o tempo de resposta

Esta seção avalia o tempo de resposta para realizar a consulta por cada algoritmo. O tempo de resposta está sendo medido em milissegundos (Figura 4).

A Figura 4(a) apresenta o tempo de resposta, variando o número de resultados k. SIA e SIA$^+$ apresentaram melhores resultados que o IFA. Entretanto a diferença entre o SIA o SIA$^+$ que ocorre em número de páginas lidas não se traduz plenamente no tempo de resposta. A principal razão é o que o SIA$^+$ requer maior processamento para processar dados concorrentemente, visto que nem todos os objetos de referência retornado para o grupo são relevantes, fazendo com que mais cálculos de distância sejam executados.

A Figura 4(b) apresenta o tempo de resposta, enquanto varia o número de palavras por consulta. O SIA e SIA$^+$ apresentam tempos de resposta similares, enquanto que o IFA é uma ordem de magnitude mais lento que os dois. Este experimento comprova que apesar do SIA acessar mais páginas que o SIA$^+$, o tempo de resposta deles é similar. O que demonstra a importância em adotar técnicas para reduzir também o processamento e não apenas o I/O caso o objetivo principal seja otimizar o tempo de resposta.

Finalmente, a Figura 4(c) apresenta o tempo de resposta para três diferentes bases de dados. Este gráfico comprova a eficiência do SIA e SIA$^+$ em reduzir tempo de resposta,

demonstrando que quanto maior a base, maior o ganho destas abordagens.

6. Conclusões

Neste artigo é apresentada uma nova consulta denominada consulta Espacial Preferencial por Palavra-chave (EPPC). Foram apresentados três novos algoritmos para processar esta consulta: IFA, SIA e o SIA$^+$. O IFA utiliza um Arquivo Invertido adaptado para indexar os objetos espaço-textuais de referência. Como o IF é um índice de indexação textual que não tem recursos para filtrar objetos de uma determinada localização espacial, os resultados desta abordagem foram inferiores quando comparado as outras duas abordagens que utilizam um índice hibrido (S2I), uma vez que a consulta EPPC necessita conhecer a relação espacial entre os objetos, e o S2I permite obter essa relação espacial com maior eficiência. O processamento concorrente do escore de objetos de interesse, proporcionou ao SIA+ um resultado superior em tempo de resposta e principalmente em I/O.

Referências

Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD*, pages 322–331. ACM.

Cao, X., Chen, L., Cong, G., Jensen, C. S., Qu, Q., Skovsgaard, A., Wu, D., and Yiu, M. L. (2012). Spatial keyword querying. In *ER*, pages 16–29. Springer.

Chen, L., Cong, G., Jensen, C. S., and Wu, D. (2013). Spatial keyword query processing: an experimental evaluation. volume 6, pages 217–228. VLDB Endowment.

Cong, G., Jensen, C. S., and Wu, D. (2009). Efficient retrieval of the top-k most relevant spatial web objects. volume 2, pages 337–348. VLDB Endowment.

Felipe, I. D., Hristidis, V., and Rishe, N. (2008). Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *ICDE*, pages 16–16. IEEE.

Papadias, D., Kalnis, P., Zhang, J., and Tao, Y. (2001). Efficient OLAP operations in spatial data warehouses. In *SSTD*, pages 443–459. Springer.

Rocha-Junior, J. B., Gkorgkas, O., Jonassen, S., and Nørvåg, K. (2011). Efficient processing of top-k spatial keyword queries. In *SSTD*, pages 205–222. Springer.

Rocha-Junior, J. B., Vlachou, A., Doulkeridis, C., and Nørvåg, K. (2010). Efficient processing of top-k spatial preference queries. volume 4, pages 93–104. VLDB Endowment.

Tsatsanifos, G. and Vlachou, A. (2015). On processing top-k spatio-textual preference queries. In *EDBT*, pages 433–444. ACM.

Vaid, S., Jones, C. B., Joho, H., and Sanderson, M. (2005). Spatio-textual indexing for geographical search on the web. In *SSTD*, pages 218–235. Springer.

Yiu, M. L., Dai, X., Mamoulis, N., and Vaitis, M. (2007). Top-k spatial preference queries. In *ICDE*, pages 1076–1085. IEEE.

Zhou, Y., Xie, X., Wang, C., Gong, Y., and Ma, W.-Y. (2005). Hybrid index structures for location-based web search. In *CIKM*, pages 155–162. ACM.

Zobel, J. and Moffat, A. (2006). Inverted files for text search engines. volume 38, page 6. ACM.

sbbd:05

Contribuição de Pesquisa entre Comunidades como Indicador de Influência

Thiago H. P. Silva, Lais M. A. Rocha
Mirella M. Moro, Ana Paula Couto da Silva

Departamento de Ciência da Computação, Universidade Federal de Minas Gerais
Belo Horizonte – MG – Brasil

{thps,laismota,mirella,ana.coutosilva}@dcc.ufmg.br

Resumo. *No contexto de extração de informações, este artigo propõe uma nova métrica de influência científica a partir de dados bibliográficos. Em particular, o objetivo é medir o grau de influência de pesquisadores através da avaliação de relações entre comunidades, que são formadas a partir dos eventos em que publicam. Parte-se do princípio que cada pesquisador tem uma comunidade-base em que ele/ela apresenta maior influência. Quando tal pesquisador trabalha em uma comunidade diferente (além da comunidade-base), leva novos conhecimentos a essa comunidade e há transferência de influência, aumentando a qualidade global das comunidades. Ao medir tal transferência, pretendemos medir a influência de pesquisadores nas respectivas comunidades.*

Abstract. *In the context of information extraction, this paper proposes a new influence metric derived from bibliographic data. Specifically, the goal is to measure the degree of influence of researchers by evaluating the links between communities, which are formed by their publication venues. Each researcher has a base community where he/she presents greater influence. Then, when such researcher works on a different community (besides the base community), he/she takes new knowledge to that community and transfers influence, which improves the global quality of the communities. By pondering such transfer, we measure the influence of researchers in their and across communities.*

1. Introdução

Uma tarefa central em bancos de dados é extrair informações relevantes a partir de seus dados. Nas últimas duas décadas, a comunidade científica tem tentado definir métricas para extrair conhecimento (geralmente na forma de rankings) a partir de dados bibliográficos, a chamada *bibliometria*. Métricas tradicionais incluem volume de artigos produzidos e seu número de citações; e uma das métricas mais utilizadas é o *h-index*, o qual combina as duas informações [Hirsch 2005]. Porém, todas essas métricas puramente quantitativas têm sofrido grandes questionamentos (e.g., [Hicks et al. 2015]).

Para contrapor essas métricas centradas no pesquisador [Bollen et al. 2009], uma solução é avaliar como o indivíduo se relaciona com seus pares e as comunidades definidas pelos mesmos. Especificamente, estudos recentes de redes sociais acadêmicas têm avaliado o desempenho de cada pesquisador individualmente e agrupado em times, comunidades, programas de pós-graduação entre outros [Freire and Figueiredo 2011, Lopes et al 2011, Newman 2004]. Outras análises exploram

aspectos qualitativos como a influência dos veículos de publicação [Garfield 1999], comunidades [Silva et al 2014, Silva et al. 2015], redes de colaborações [Brandão et al. 2014], perfis acadêmicos [Gonçalves et al 2014, Lima et al 2015], e outros.

Considerando a perspectiva social, alguns estudos focam em analisar como posições privilegiadas na rede são definidos de diferentes maneiras. Por exemplo: Granovetter (1973) introduziu a ideia de *weak ties*; Newman (2000) explorou o número de caminhos mais curtos entre pares de nós para medir a influência sobre o fluxo de informações entre os indivíduos; e Burt (2004) chamou de *brokers* as pessoas que constroem capital social ao se posicionarem em pontos estratégicos na rede (*structural holes*). De maneiras diferentes, tais estudos enfatizam a importância de construir pontes e de conectar nós distantes da rede social. Ou seja, em redes sociais acadêmicas, pesquisadores que conectam diferentes grupos devem trazer mais influência a esses grupos. A hipótese base é explorada em outros contextos (e.g., em dados econômicos), e nosso estudo mostra que esta também pode ser aplicada como indicador de influência na área de Computação.

Este trabalho contribui para esta discussão em dois aspectos: (*i*) mostra que é possível ampliar o conceito de construção de pontes ao explorar a relação pesquisador-comunidade e (*ii*) propõe uma nova estratégia que mensura a influência que, ao mesmo tempo, lida com o problema da discrepância entre comunidades ao projetar a influência sob uma mesma comunidade. Especificamente, analisamos a propagação de influência entre *comunidades*, que são formadas por pesquisadores que compartilham interesses comuns pela publicação de artigos na mesma conferência. Para isso, definimos que cada pesquisador possui uma *comunidade base* como sendo aquela em que ele possui o melhor desempenho. Então, seguimos os conceitos de diversidade e novidade [Burt 2004], e consideramos que quando um pesquisador trabalha em uma comunidade diferente (além da comunidade base), ele/ela transfere novos conhecimentos de seu contexto para outros domínios. Ou seja, há transferência de novas ideias, técnicas e metodologias que resultam em contribuições para a evolução da qualidade de pesquisa como um todo.

O novo indicador proposto, chamado **3c-index** (*Cross-Community Contribution*), utiliza uma função genérica de *score* para avaliar o grau de influência do pesquisador em diferentes comunidades, e depois projeta tal valor em sua comunidade base. As principais vantagens são: *flexibilidade*, pois o *3c-index* permite o uso de quaisquer funções de score (e.g., número de citações, volume, h-index, socialibilidade ou número de coautores, volume de produção recente e métricas de redes complexas); e *equidade*, pois releva os diferentes perfis das comunidades envolvidas através da projeção em uma única comunidade. Nossa análise também mostra que o *3c-index* traz resultados diferentes nos ranqueamentos em relação aos índices tradicionais e identifica quais são os pesquisadores que estão em buracos estruturais, i.e., aqueles que mais atuam em diferentes partes da rede tornando-a mais forte. A nossa validação experimental também reporta que os resultados do *3c-index* superam índices tradicionais na tarefa de ranquear pesquisadores com premiações dadas por suas contribuições e inovações em suas comunidades.

Finalmente, é importante notar que o 3c-index não deve ser utilizado sozinho. O seu objetivo principal é fornecer uma análise *complementar* cujo resultado não pode ser diretamente aferido a partir de uma simples consulta SQL aos dados bibliográficos. Segundo Hicks et al. (2015) , é necessário que seja utilizado por especialistas junto a outras avaliações qualitativas.

2. Trabalhos relacionados

É sempre difícil avaliar pessoas de acordo com algum critério. Neste trabalho focamos na relação social entre os pesquisadores e suas comunidades. Um dos primeiros trabalhos que lidam com a perspectiva social foi realizado por Granovetter (1973) através da ideia de *weak ties* como sendo as relações que unem diferentes partes da rede através da construção de pontes. Newman (2000) considerou o papel de um pesquisador na rede e explorou o número de caminhos mais curtos de um nó central para outros nós e, assim, mediu a influência sobre o fluxo de informações entre os indivíduos (níveis de *brokering*). De forma similar, Burt (2004) chama de *brokers* as pessoas que constroem capital social ao se posicionarem em pontos estratégicos na rede (*structural holes*). Nosso trabalho contribui para esta discussão ao analisar a influência propagada por pesquisadores entre comunidades, i.e., mostramos que podemos ampliar o conceito da construção de pontes ao explorar a relação pesquisador-comunidade.

Considerando a área de Computação, Freire e Figueiredo (2011) exploraram as colaborações *externas* de indivíduos e grupos. Eles identificaram os indivíduos e grupos influentes através da intensidade de seus relacionamentos com indivíduos de fora do grupo. De forma similar, Silva et al. (2014) exploraram o conceito de *comunidade* para ranquear veículos de acordo com o grau das relações externas de seus membros. Os resultados indicam que pesquisadores que publicam em outras comunidades possuem maior probabilidade de introduzir novas ideias de sua competência para outros contextos e, sendo assim, tais pesquisadores são considerados altamente influentes por ligarem diferentes partes da rede (ou seja, funcionam como *weak ties* ou *brokers* da rede). Em relação ao fluxo de contribuições entre comunidades, Silva et al. (2015) propuseram quatro métricas para mensurar a dinâmica entre pesquisadores em comunidades (*Permanency*, *Migration*, *Exclusivity* e *Plurality*). Aqui, propomos avaliar pesquisadores explorando o vínculo de um pesquisador com seu conjunto de comunidades e, a partir disso, explorar o grau de influência que pode ser transmitida entre diferentes comunidades.

Em termos de análise da produtividade, é necessário considerar os diferentes padrões de publicação existentes. Por exemplo, Gonçalves et al. (2014) caracterizaram a produtividade de pesquisadores e concluíram que há, realmente, perfis diferentes e bem definidos. Comunidades de pesquisadores também possuem padrões distintos quanto ao número de membros, áreas de pesquisa, taxas de publicações, etc. Para lidar com tal problema, Silva et al. (2015) estabeleceram propriedades (*Equality*, *Relativity* e *Temporality*) que devem ser discutidas na metodologia para uma comparação mais justa. Aqui, nós definimos a *comunidade base* do pesquisador como aquela onde ele possui o melhor desempenho, e a influência é medida na *perspectiva de cada pesquisador em relação à sua comunidade base, buscando uma avaliação relativamente mais justa*.

No contexto de ranqueamento, Silva et al. (2015) propuseram uma estratégia baseada em *grupos similares* formados por membros com características comuns em relação ao vínculo temporal com os veículos de publicação, argumentando que há mais justiça ao comparar perfis semelhantes. Lima et al. (2013) criaram uma estratégia genérica para pesquisadores pertencentes a múltiplas áreas através da projeção da produtividade sob uma mesma perspectiva. Similarmente, nós aplicamos uma estratégia de ranqueamento baseado em percentis para mapear os valores de diferentes comunidades para uma *comunidade base* do pesquisador. Além disso, corrigimos os valores ao definir o grau de

Tabela 1. Grau de influência entre comunidades de Christos Faloutsos.

Comunidade	Percentil	$infd_i$
SIGKDD	0.995	0
SIGMOD	0.92	0.07
SIGMETRICS	0.68	0.32
SIGAPP	0.48	0.52

influência existente entre as comunidades sob a perspectiva da contribuição de cada autor. Assim, quanto maior o grau de influência, maior será a probabilidade de que haja transferência de conhecimento do pesquisador para comunidades distintas.

3. Contribuição entre Comunidades

Pesquisadores podem ser agrupados em uma ou mais áreas de competência [Lima et al 2013, Lima et al 2015]. Então, a transferência de conhecimento entre áreas é fundamental para contribuir com a evolução da Ciência, pois possibilita aplicações de ideias conhecidas e amplamente utilizadas em um contexto para resolver problemas em outros domínios [Sun et al 2013]. Neste trabalho definimos a métrica **3c-index** que identifica a transferência de conhecimento entre diferentes comunidades. O novo índice mede a influência dos pesquisadores de acordo com as suas especialidades (definido como *grau de influência*) para outras comunidades.

Uma característica importante do 3c-index consiste em lidar com justiça ao comparar comunidades com diferentes padrões de publicação. Para isso, definimos o conceito de *comunidade base* de um pesquisador como aquela em que ele possui o melhor desempenho. A escolha da comunidade base é definida em termos de percentis, i.e., de acordo com as posições dos pesquisadores no ranqueamento. Formalmente, p_i^c é o percentil do pesquisador *i* na comunidade *c* definido por: $p_i^c = \frac{l_i^c + 0.5e_i^c}{N^c}$, onde N^c é o número de pesquisadores na comunidade c, l_i^c e e_i^c o número dos pesquisadores com valores no ranqueamento inferiores ou iguais ao do pesquisador i, respectivamente. Por exemplo, para uma comunidade com 100 pesquisadores com scores distintos (sem empates), o pesquisador na posição 10 possui $l_i^c = 89$ e $e_i^c = 1$, resultando no percentil de 89.5%. Após o cálculo de p_i^c para todas as comunidades em que o pesquisador publica, defini-se *comunidade base* b_i como sendo aquela onde o pesquisador possui o maior valor para o percentil, i.e., $b_i = \text{argmax}_{c \in C} p_i^c$.

Tendo definido a *comunidade base* de um pesquisador, mede-se então o grau de influência $infd_i$ como a diferença entre os percentis (i.e., $infd_i = b_i - p_i^c$). Por exemplo, considerando todos os autores que publicam em um evento promovido por um SIG[1] como uma comunidade científica, pode-se identificar a influência propagada por seus membros para as demais comunidades. Para ilustrar, a Tabela 1 mostra o grau de influência propagada pelo pesquisador *Christos Faloutsos*: sua comunidade base é a SIGKDD, no qual p_i^c apresenta maior valor; SIGMOD evidencia grau similar; e com graus moderados de contribuições estão SIGAPP e SIGMETRICS. Note que $infd_i = 0$ para a comunidade base pois, por definição, não há transferência de conhecimento neste caso. Assim, o grau de influência captura os relacionamentos importantes dos pesquisadores que atuam como *pontes* entre comunidades.

[1]Association for Computing Machinery Special Interest Groups. `http://www.acm.org/sigs`

Dado que comunidades científicas possuem perfis distintos (i.e., número de artigos, taxas de publicações e citações, etc), é necessário um fator de normalização. Conforme o modelo de ranqueamento proposto por Lima et al. [Lima et al 2013], nossa estratégia considera os valores dos percentis dos pesquisadores *projetados* para sua comunidade base. O *3c-index* de um pesquisador i é então definido como

$$3c{-}index(i) = score(b_i) + 2\sum_{c \in C}(b_i - p_i^c)score(f_b(p_i^c)),$$

onde $score(x)$ é o valor no ranqueamento na posição de percentil x, $f_b(x)$ é a função de projeção que mapeia o percentil x para o percentil correspondente no ranqueamento da comunidade base b. Dessa forma, o valor final consiste no valor obtido na sua comunidade base ($score(b_i)$) somado aos valores do ranqueamento provenientes de contribuições em comunidades externas. Note que a estratégia de ranqueamento é genérica e possibilita o uso de qualquer função de *score* como: número de citações, volume de publicações, h-index, número de alunos formados, número de coautores, métricas de análise de redes complexas, entre outros.

4. Metodologia

Esta seção apresenta a metodologia a ser utilizada na avaliação do nosso novo índice. Especificamente, o processo de avaliação é apresentado em duas perspectivas. Na primeira (Seção 5), endossamos a existência da troca de contribuição entre comunidades, investigamos as formações das comunidades em relação aos membros que a possuem como *comunidade base*, e avaliamos a independência entre a nossa proposta e as métricas padrão. Na segunda (Seção 6), validamos a nossa abordagem para ranquear pesquisadores relevantes (vencedores de *ACM Awards*) ao compararmos com as métricas padrão.

Para realizar tais avaliações, primeiro é necessário definir uma base de dados que contenha todas as informações necessárias sobre os pesquisadores a serem avaliados. Para avaliar e validar o índice proposto, também é necessário: definir métricas padrão para função de score e com as quais o desempenho do novo índice será comparado; definir uma tabela verdade para verificar se os resultados do índice fazem sentido; e definir métricas de avaliação considerando o ranqueamento gerado pelo índice. A seguir, define-se cada uma dessas partes.

Base de Publicações. O 3c-index é baseado no conceito de comunidade. Para fins de avaliação experimental, considera-se as comunidades definidas por um sub-conjunto de ACM SIGs. Cada SIG organiza um ou mais eventos científicos sobre tópicos de interesse específicos. Além de promoverem grandes eventos, essas SIGs concedem prêmios a seus membros devido a serviços prestados, contribuições e inovações (*ACM Awards*). Especificamente, são 12 conferências como comunidades, as quais são consideradas de alta qualidade na área de Ciência da Computação e também são alvos de pesquisadores de alta qualidade. A Tabela 2 apresenta as comunidades e suas estatísticas. Para cada evento, foram coletadas suas listas de artigos a partir da DBLP[2] considerando o intervalo de tempo [2001,2010]. Também foram coletadas as citações dos trabalhos na plataforma do *Google Scholar*[3], casando-se 4-tuplas formadas por {título, ano, autores, veículo}. Por fim, a base de dados contém mais de dezoito mil autores, mais de cem mil artigos com

[2]DBLP: `http://www.informatik.uni-trier.de/~ley/db`
[3]GS: `http://scholar.google.com`

Tabela 2. Estatísticas da base de dados compreendendo o intervalo [2001,2010].

SIG	Aut.	Art.	$\frac{\text{Aut.}}{\text{Art.}}$	Cit. (10^{-3})	$\frac{\text{Cit.}}{\text{Aut.}}$	$\frac{\text{Cit.}}{\text{Art.}}$	#Awards
SIGACCESS	849	418	2,03	7,9	9,25	18,79	2
SIGAda	169	135	1,25	0,7	4,21	5,27	15
SIGAPP	6732	3078	2,19	45,2	6,71	14,68	7
SIGCOMM	816	338	2,41	104,7	128,35	309,86	14
SIGCSE	1957	1143	1,71	23,1	11,78	20,17	25
SIGDOC	460	316	1,46	3,0	6,61	9,62	9
SIGIR	2313	1528	1,51	85,2	36,86	55,79	5
SIGKDD	2172	1075	2,02	109,3	50,33	101,69	17
SIGMETRICS	1053	449	2,35	31,2	29,63	69,49	5
SIGMOBILE	748	276	2,71	66,9	89,41	242,31	7
SIGMOD	2196	1098	2	103,9	47,32	94,65	29
SIGUCCS	838	655	1,28	1,7	2	2,56	5
Todas	18511	10509	1,76	582,8	31,49	55,46	137

mais de meio milhão de citações recebidas. As médias são de 1,8 de autores por artigo e de 55,5 citações por artigo.

Métricas padrão. São considerados três índices bibliométricos bem conhecidos: volume de publicações, número de citações e h-index. Tais índices são amplamentes usados em plataformas de busca orientados à pesquisa como *Google Scholar*, *Microsoft Academic Search* e *AMiner*. Como a estratégia de ranqueamento é genérica e possibilita o uso de qualquer função de *score* (Seção 3), a seguir, simplificamos a nomenclatura como 3c-citações, 3c-volume e 3c-h-index para indicar que estamos usando a métrica 3c-index com as funções de *score*, respectivamente, das métricas: número de citações recebidas, volume de publicações e h-index.

Tabela Verdade. Ranquear pesquisadores é um grande desafio devido à falta de um consenso de quais são as métricas ideais para tornar o processo de decisão justo. Alternativas de avaliação consistem em determinar grupos de referências contendo membros com relevância inquestionável em suas avaliações. Por exemplo, Hirsch (2005) usou os vencedores do prêmio Nobel para avaliar sua métrica (h-index), e Lima et al. (2013) usaram os níveis de bolsas governamentais concedidas. De forma similar, nós construímos um *ground-truth* de pesquisadores influentes como sendo os 137 (dos 18.511) pesquisadores vencedores de pelo menos um *ACM Award*. Nós assumimos que esses seletos pesquisadores possuem relevância e impacto inquestionáveis para as comunidades que os premiaram.

Métricas de Avaliação. Para avaliação, são considerados dois tipos de métrica: um para comparar o resultado do ranqueamento, e outro para validar o ranqueamento.

Com o intuito de avaliar o comportamento do 3c-index, consideramos a correlação de Spearman ρ (mede a dependência estatística entre dois rankings) e a distância de Jaccard (interseção sobre união) para mensurar a dissimilaridade entre dois conjuntos. O objetivo principal é verificar a independência entre a nossa abordagem e as métricas padrão, i.e., verificar se a proposta traz novidade com possibilidade de ser usada como forma completementar para uma análise mais completa.

Para melhor investigar a eficácia do 3c-index para ranqueamento, utilizamos a métrica DCG (*Discounted Cumulative Gain*). Esta métrica mede a qualidade do ranqueamento de acordo com uma escala graduada com um fator logarítmico de penalização para itens relevantes recuperados tardiamente [Järvelin and Kekäläinen 2002]. Formalmente,

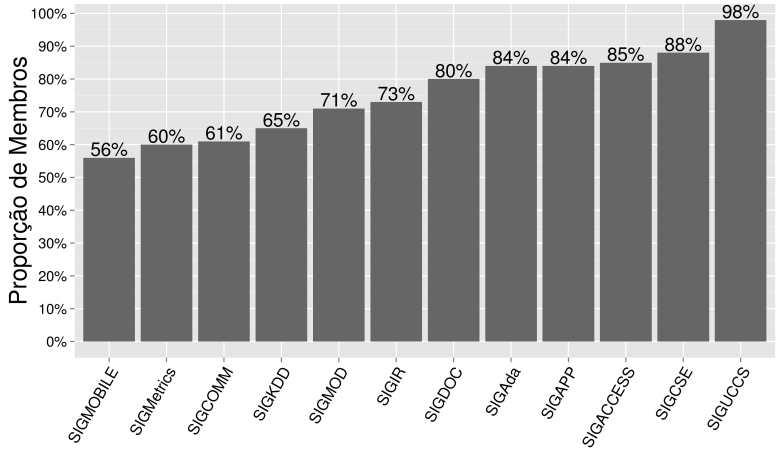


Figura 1. Porcentagem de autores (com mais de um artigo) que possuem a própria comunidade como sua *comunidade base* de acordo com o h-index.

o DCG na posição de ranqueamento k é definido como $DCG@k = g_1 + \sum_{i=2}^{k} \frac{g_i}{log_2(i)}$, onde g_i denota uma relevância binária (i.e., 1 se o pesquisador é vencedor de um *ACM Award*, 0 caso contrário). Nós usamos a versão normalizada (nDCG), que é obtida ao dividirmos o DCG@k pelo melhor possível ranqueamento no mesmo corte k.

5. Análise

Nesta seção investigamos a aplicabilidade do 3c-index para o contexto de comunidades científicas. A Seção 5.1 verifica a dependência das comunidades em relação à formação dos seus membros de acordo com o conceito de *comunidade base*, i.e., estamos interessados em endossar que há evidências o suficiente para aplicarmos a métrica em tais contextos. A Seção 5.2 mostra os pesquisadores mais influentes nos SIGs, bem como o reconhecimento deles pelas comunidades da ACM. Então, a Seção 5.3 compara a nossa proposta com índices tradicionais com objetivo de mostrar a independência entre eles.

5.1. Transferência de Conhecimento

A questão a ser respondida nesta seção é se existe transferência de conhecimento entre comunidades. Uma vez que parte do valor agregado da métrica proposta consiste em quantificar o grau de influência dos pesquisadores em comunidades, temos então que analisar como é composta cada comunidade em relação à participação de membros externos. Consideramos como função de score o percentil de acordo com o ordenamento pelo h-index[4] e que somente os membros com mais de uma publicação em uma conferência formam a comunidade da mesma. A Figura 1 mostra a proporção de participantes em cada SIG que possuem a própria comunidade como base. Por exemplo, 56% dos membros da SIGMOBILE a possuem como sua comunidade base, i.e., ela recebe contribuição externa de 44% dos seus autores. Em contrapartida, a SIGUCCS possui apenas 2% de contribuição externa, indicando que a conferência tende a possuir uma comunidade extremamente fechada (de fato, o foco desta SIG é dar suporte logístico aos serviços de tecnologia da informação para instituições de ensino[5]). As conferências restantes pos-

[4]Foram obtidos valores similares para o número de citações recebidas e volume.

[5]SIGUCCS: `http://www.siguccs.org/`

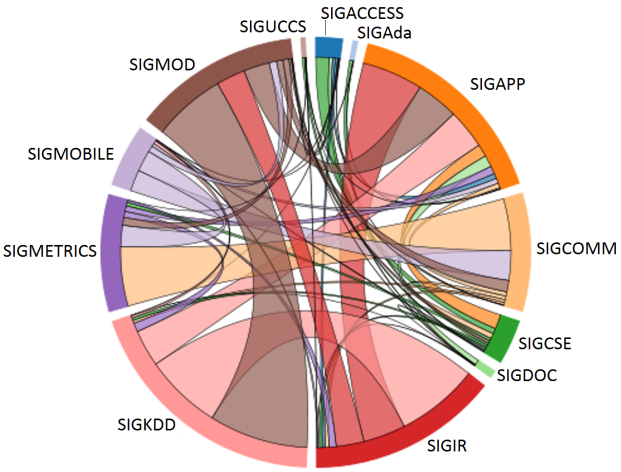


Figura 2. Transferência de Conhecimento entre Comunidades (ACM SIGs).

suem proporções de 12% a 40% de membros externos, ressaltando a existência da troca de conhecimento entre comunidades diferentes.

Para uma visualização mais clara da interação entre as comunidades, a Figura 2 mostra a transferência de influência entre as SIGs. Por exemplo, se a comunidade base de um pesquisador for SIGMOD e ele publicar no SIGKDD, então o seu valor do 3c-index (usando o h-index) é contabilizado como partindo do SIGMOD para o SIGKDD (i.e., nesse caso o valor acumulado somente é atribuído para a SIGKDD que foi o destino da contribuição). Sendo assim, cada parte da visualização circular é proporcional ao acúmulo de contribuições recebidas. As conferências SIGUCCS, SIGAda e SIGDOC possuem os menores valores de transferência de conhecimento, enquanto as conferências SIGKDD, SIGAPP, SIGIR e SIGMOD possuem os maiores valores. O aspecto do gráfico é bem coerente em relação às áreas que possuem uma maior proximidade dos seus programas como SIGMOD/SIGKDD, SIGMOD/SIGIR e SIGCOMM/SIGMETRICS. Finalmente, é possível notar que existe interações entre todas as comunidades.

5.2. Pesquisadores Influentes nos SIGs

Nesta seção verifica-se quais são os pesquisadores considerados mais influentes nos SIGs, confrontando essa lista com as distinções de reconhecimento pelos seus trabalhos de acordo com a ACM. A Tabela 3 mostra os top 10 pesquisadores de acordo com o 3c-index usando como funções de score o número de citações recebidas, volume e h-index. A notação é a seguinte: os pesquisadores que já ganharam premiações devido às suas contribuições e inovações para uma das comunidades (i.e., o prêmio é dado pelo próprio SIG) estão em negrito; e outros reconhecimentos da ACM (*fellow*, *distinguished* e *senior*)[6] estão marcados com símbolos.

Os resultados mostram pesquisadores conhecidos na área de Computação. Em especial, o 3c-volume e o 3c-h-index têm em suas primeiras colocações os pesquisadores agraciados com prêmios dados por suas comunidades (cinco primeiros para volume e quatro primeiros para o h-index). Este resultado é importante pois há apenas 137 de 18.511 pesquisadores de nossa base que ganharam tal premiação. Na próxima subseção compa-

[6]ACM Membership: `http://awards.acm.org/grades-of-membership.cfm`

Tabela 3. Pesquisadores melhor ranqueados de acordo com 3x-index nos SIGs.

Posição	3c-citações	3c-volume	3c-h-index
1º	**Scott Shenker**♯	**W. Bruce Croft**♯	**Jiawei Han**♯
2º	Ion Stoica♯	**Christos Faloutsos**♯	**Christos Faloutsos**♯
3º	M. Frans Kaashoek♯	**Surajit Chaudhuri**♯	**Scott Shenker**♯
4º	David R. Karger♯	**Jiawei Han**♯	**W. Bruce Croft**♯
5º	Sylvia Ratnasamy	**Scott Shenker**♯	ChengXiang Zhai‡
6º	Mark Handley	ChengXiang Zhai‡	**Surajit Chaudhuri**♯
7º	Paul Francis	Philip S. Yu♯	Wei-Ying Ma‡
8º	Richard M. Karp♯	Zheng Chen†	Zheng Chen†
9º	**Jon M. Kleinberg**♯	Divesh Srivastava♯	Philip S. Yu♯
10º	Dina Katabi♯	Leif Azzopardi	Divesh Srivastava♯

Em negrito os ganhadores do ACM Awards, ♯*ACM fellow*, ‡*ACM distinguished scientist*, †*ACM senior member*.

ramos o 3c-index em relação à composição de pesquisadores nas primeiras posições do ranqueamento (i.e., se são semelhantes), bem como os seus posicionamentos.

5.3. Comparação com Métricas Padrão

Outra questão a ser respondida é se existe independência entre o índice proposto e as métricas padrão. Esta questão é importante para mostrar que a nossa estratégia captura características diferentes e, dessa forma, pode ser usada de forma *complementar* com outros índices. Primeiro consideramos o posicionamento dos pesquisadores de acordo com cada métrica. Para isso, usamos o coeficiente de Spearman que mede a intensidade da ordem (ao invés do valor) entre dois ranqueamentos. Em seguida, exploramos a formação do ranqueamento de cada métrica em relação à presença dos mesmos pesquisadores, i.e., verificamos se há dissimilaridade entre os conjuntos de pesquisadores que compõem as primeiras posições. Tal dissimilaridade é medida por 1 menos a divisão entre a interseção e união de dois conjuntos (Jaccard).

A Tabela 4 mostra as correlações de Spearman entre índices tradicionais e seus correspondentes na versão proposta 3c-index. Aqui, considera-se a correlação forte e muito forte para $\rho \geq 0.7$, moderada para $0.4 \leq \rho < 0.7$ e fraca para $\rho < 0.4$. Em relação à métrica do número de citações recebidas e o 3c-index (C e $3c$), não há correlação para o topo do ranqueamento até a décima posição[7], valores moderados na comparação até a quadragésima posição, e valores fracos para as demais. As correlações do volume (V e $3c$) e do h-index (H e $3c$) em relação aos seus correspondentes na métrica 3c-index são semelhantes, com valores moderados para o topo do ranking (posições até 10 e 20) e valores fracos para as demais. De acordo com tal posicionamento dos pesquisadores em cada ranqueamento, há um grau de *independência* entre as métricas mostrada pelas correlações fracas e moderadas.

Para investigar se a formação do ranqueamento é similar em relação à presença dos mesmos pesquisadores, a Tabela 5 mostra as distâncias de Jaccard entre a métrica proposta e os índices tradicionais. Exceto em relação às citações ($d_J(C, 3c)$) para 20 pesquisadores e para o h-index ($d_J(H, 3c)$) para 10 pesquisadores, todas as configurações possuem dissimilaridade de pelo menos 24%. Desta forma, as métricas também são independentes em relação à formação dos conjuntos de pesquisadores nas primeiras posições.

[7]Os 10 primeiros na métrica de citações estão posicionados entre os 16 primeiros pelo 3c-index, sendo somente dois pesquisadores com as mesmas posições nos dois ranqueamentos.

Tabela 4. Correlações de Spearman entre citações (C), volume (V), h-index (H) e seus 3c-index ($3c$). Ranqueamentos ordenados pela métrica original.

Posições	C e $3c$	V e $3c$	H e $3c$
10	0,04	0,57	0,67
20	0,56	0,67	0,66
40	0,45	0,27	0,12
60	0,27	0,13	0,06
80	0,30	0,09	0,13
100	0,34	0,21	0,16

Tabela 5. Distância de Jaccard para os conjuntos com 10, 20, 50 e 100 pesquisadores entre as métricas citações (C), volume (V), h-index (H), e seus 3c-index ($3c$).

#Pesq.	$d_J(C, 3c)$	$d_J(V, 3c)$	$d_J(H, 3c)$
10	0,33	0,46	0,18
20	0,10	0,26	0,40
40	0,33	0,40	0,43
60	0,29	0,48	0,38
80	0,24	0,43	0,40
100	0,28	0,35	0,40

6. Validação Experimental

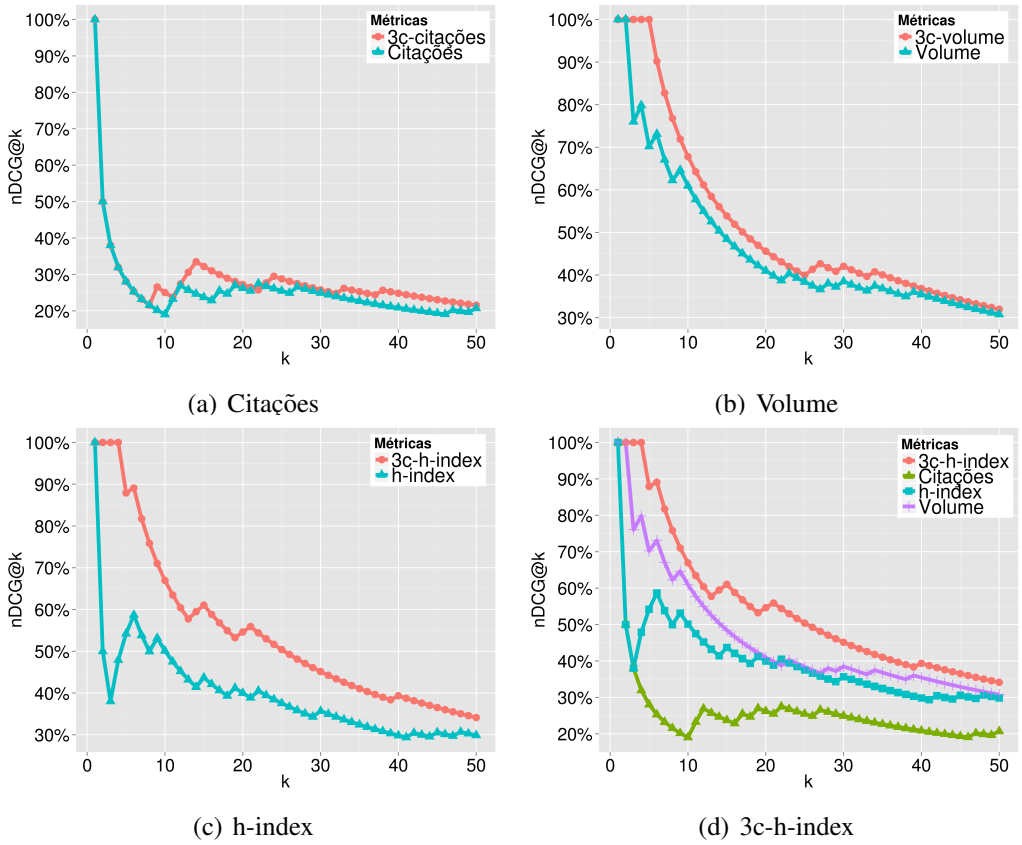
Após compararmos o comportamento do ranqueamento do 3c-index com as *métricas padrão*, nesta seção validamos o uso da métrica para ranquear pesquisadores considerados relevantes. Especificamente, a tarefa consiste em ranquear 18.511 pesquisadores com o objetivo de que as primeiras posições sejam ocupadas pelos 137 pesquisadores ganhadores de pelo menos um *ACM Awards*.

A Figura 3 compara o ranqueamento produzido pelo 3c-index com o número de citações, volume de publicações e h-index. Todos os gráficos consideram o valor *nDCG@k* obtido a cada corte *k*, variando da 1ª a 50ª posição (i.e., $k = 10$ mostra a comparação das métricas para ranquear somente os 10 primeiros pesquisadores). A figura mostra que o 3c-index supera consistentemente os índices tradicionais na maioria dos casos. Especificamente, considerando o número de citações recebidas (Figura 3(a)), há um empate entre as métricas nas primeiras posições até a 8ª posição e, então, o 3c-index supera ou é igual ao seu correspondente (exceto para a 22ª posição). Em relação ao volume de publicações (Figura 3(b)), há um empate entre as métricas nas primeiras duas posições e, a partir desta posição, a nossa proposta supera o seu correspondente.

É importante notar que as métricas baseadas no acúmulo de citações tendem a ser sensíveis a *outliers*, já que um só trabalho pode atrair muitas citações e, desta forma, podem existir ruídos nas avaliações que consideram conjuntos de artigos. O número de artigos publicados também é sensível quando, por exemplo, um autor pode assumir as primeiras posições ao ter um vasto volume em veículos específicos (i.e., maior taxa de aceitação e/ou menor qualidade). Como estamos considerando veículos de alto nível, o viés tende a ser diminuído.

Para contornar tais problemas, o h-index tenta controlar a sensibilidade ao considerar o volume e o número de citações recebidas ao mesmo tempo. De fato, o melhor desempenho comparativo da nossa proposta consiste na versão dada pelo h-index (Figura 3(c)), onde o 3c-index tem o melhor desempenho ao deixar nas primeiras quatro posições apenas pesquisadores relevantes e, desta forma, tende a superar o seu correspondente até o fim. A Figura 3(d) mostra o comparativo entre 3c-index (com o h-index como sua função de score) e as três métricas padrão resumindo o bom desempenho da nossa estratégia.

Os resultados inferiores das métricas tradicionais indicam que elas ranqueiam os pesquisadores premiados tardiamente (em posições mais afastadas do topo). Portanto, medir o impacto da transferência de conhecimento entre comunidades e lidar com os padrões de publicação de comunidades distintas é uma boa estratégia para ranquear pesquisadores influentes em comunidades.



(a) Citações (b) Volume

(c) h-index (d) 3c-h-index

Figura 3. Comparação entre 3c-index e os *baselines* de acordo com nDCG.

7. Conclusão

Neste trabalho apresentamos um novo índice de influência baseado na perspectiva social dos pesquisadores chamado 3c-index, que explora o grau de influência que pesquisadores exercem sobre comunidades. A proposta é robusta para lidar com o problema de ranqueamento justo de membros de comunidades com padrões distintos, bem como flexível no uso de qualquer função de score (seja baseada em volume, citações, métricas de redes complexas, etc). A análise experimental mostrou independência do 3c-index das métricas padrões em relação ao ranqueamento e formação do ranqueamento nas primeiras posições. Portanto, tais resultados endossam o uso da métrica de forma *complementar* para uma análise mais completa. Por fim, a nova abordagem teve o melhor desempenho para resolver um problema real de ranqueamento dos pesquisadores mais influentes (reconhecimento explícito por inovações e contribuições) em comunidades reconhecidas na área de Computação.

Como trabalho futuro, pretendemos aplicar o novo índice para investigar a influência entre as comissões especiais da SBC. Porém, existe a dificuldade extra de encontrar todas as informações com relação aos prêmios distribuídos pelas mesmas. O índice proposto também pode ser usado para identificar quais são os principais pesquisadores brasileiros na área de Computação ao se posicionarem bem entre instituições [Brandão and Moro 2012]. Outro estudo a ser realizado é a verificação do potencial de influência para utilização em sistemas de recomendação.

Agradecimentos. Trabalho parcialmente financiado por CNPq e FAPEMIG, Brasil.

Referências

Bollen, J., Van de Sompel, H., Hagberg, A., and Chute, R. (2009). A principal component analysis of 39 scientific impact measures. *PLoS ONE*, 4(6):e6022.

Brandão, M. A. and Moro, M. M. (2012). Recomendação de colaboração em redes sociais acadêmicas baseada na afiliação dos pesquisadores. In *SBBD*, pages 73–80.

Brandão, M. A., Moro, M. M., and Almeida, J. M. (2014). Experimental evaluation of academic collaboration recommendation using factorial design. *JIDM*, 5(1):52–63.

Burt, R. S. (2004). Structural Holes and Good Ideas. *American Journal of Sociology*, 110(2):349–399.

Freire, V. P. and Figueiredo, D. R. (2011). Ranking in collaboration networks using a group based metric. *J. Braz. Comp. Soc.*, 17(4):255–266.

Garfield, E. (1999). Journal impact factor: a brief review. *Canadian Medical Association Journal*, 161(8):979–980.

Gonçalves et al, G. D. (2014). Characterizing scholar popularity: A case study in the Computer Science research community. In *JCDL*, pages 57–66, London, UK.

Granovetter, M. S. (1973). The Strength of Weak Ties. *American Journal of Sociology*, 78(6):1360–1380.

Hicks, D., Wouters, P., Waltman, L., de Rijcke, S., and Rafols, I. (2015). Bibliometrics: The Leiden Manifesto for research metrics. *Nature*, 520(7548):429–431.

Hirsch, J. E. (2005). An index to quantify an individual's scientific research output. *PNAS*, 102(46):16569–16572.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Lima et al, H. (2013). Aggregating Productivity Indices for Ranking Researchers Across Multiple Areas. In *JCDL*, pages 97–106, Indianapolis, USA.

Lima et al, H. (2015). Assessing the profile of top brazilian computer science researchers. *Scientometrics*, 103(3):879–896.

Lopes et al, G. R. (2011). Ranking Strategy for Graduate Programs Evaluation. In *ICITA*, pages 253–260, Sydney, Australia.

Newman, M. E. (2004). Who Is the Best Connected Scientist? A Study of Scientific Coauthorship Networks. 650:337–370.

Silva, T. H. P., Moro, M. M., and Silva, A. P. C. (2015). tc-index: a New Research Productivity index based on Evolving Communities. In *TPDL*, Poznań, Poland.

Silva et al, T. H. P. (2014). Community-based Endogamy as an Influence Indicator. In *JCDL*, pages 67–76, London, UK.

Silva et al, T. H. P. (2015). Authorship Contribution Dynamics on Publication Venues in Computer Science: an Aggregated Quality Analysis. In *SAC*, Salamanca, Spain.

Sun et al, X. (2013). Social dynamics of science. *Sci. Rep.*, 3(1069).

sbbd:06

Correlação Probabilística de Bancos de Dados Governamentais

Clícia Pinto[1], Robespierre Pita[1], Pedro Melo[1], Samila Sena[1], Marcos Barreto[1]

[1]Departamento de Ciência da Computação
Instituto de Matemática – Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros, s/n – CEP: 40.110-170 – Salvador – BA – Brasil

cliciasp1@gmail.com, marcosb@ufba.br

Abstract. *Statisticians and epidemiologists need to use data stored in databases from the National Unified Health System (SUS), "Bolsa Família", and Cadastro Único to perform studies on the effects of social programmes held by the government in the incidence of some diseases in poor families. To support such studies, we designed a methodology and implemented a pipelining for data quality assessment, data conditioning, probabilistic linkage, and accuracy ascertainment. This paper presents our methodology and discuss some case studies in controlled and non-controlled scenarios.*

Resumo. *Dados contidos no Sistema Único de Saúde, no Programa Bolsa Família e no Cadastro Único são usados por estatísticos e epidemiologistas em estudos sobre os efeitos de programas sociais na incidência de doenças em famílias pobres. Para suportar tais estudos, nós desenvolvemos uma metodologia e implementamos um pipelining para a análise da qualidade dos dados, o pré-processamento, a correlação destas bases e a averiguação da acurácia dos resultados. Este artigo apresenta esta metodologia e discute estudos de casos em cenários controlados e não-controlados.*

1. Introdução

Na Saúde, um conjunto significativo de aplicações requerem a integração de dados de sistemas médicos e hospitalares objetivando prover suporte para a análise de políticas públicas e a incidência de doenças, o monitoramento de intervenções e o acompanhamento do tratamento de pacientes. Tais aplicações apresentam um grande volume de dados a serem integrados e processados para a geração de informações.

Este trabalho conjunto entre a Universidade Federal da Bahia, a FioCruz Bahia, a Universidade de Brasília, a *London School of Hygiene and Tropical Medicine* e o *Farr Institute* objetiva avaliar a efetividade do Programa Bolsa Família em relação a mortalidade infantil e a incidência de doenças (tuberculose, hanseníase e AIDS). Os dados advêm das bases do Cadastro Único, dos pagamentos do Bolsa Família e do Sistema Único de Saúde (SUS). Tais dados devem ser integrados para a composição de uma "coorte"[1] composta por famílias beneficiárias do Bolsa Família, a qual é comparada por métodos estatísticos às famílias não-beneficiárias [Keyes and Galea 2014].

[1]Conjunto de dados usado em estudos epidemiológicos.

A construção da coorte demanda o tratamento de questões ligadas à qualidade dos dados, à segurança de armazenamento, à preservação de privacidade, à padronização de dados e ao processamento eficiente sem comprometer a acurácia dos resultados. uma metodologia para tratar tais questões vem sendo desenvolvida desde 2013 e implementada como um *pipelining*, o qual emprega o ambiente de execução Spark [Zaharia et al. 2010] e algumas ferramentas estatísticas (SPSS e R).

Este artigo está organizado como segue: a Seção 2 descreve os programas sociais e os dados governamentais relacionados a este projeto. Na seção 3 é descrita a metodologia e a implementação realizada. Estudos de casos são discutidos na Seção 4. Alguns trabalhos relacionados são comentados na Seção 5 e algumas conclusões e atividades em andamento são apresentadas na Seção 6.

2. Dados governamentais e programas sociais

O governo brasileiro mantém diversos programas sociais, os quais selecionam seus beneficiários de acordo com dados socioeconômicos armazenados no Cadastro Único [MDS 2015a]. Ao ingressar no Cadastro Único, cada componente da família recebe um NIS (Número de Identificação Social), o qual deve ser renovado a cada dois anos e serve para identificar este indivíduo no âmbito dos programas sociais.

O Bolsa Família é um dos maiores programas de transferência condicional de renda no mundo [MDS 2015b]. Através dele, famílias registradas no Cadastro Único consideradas pobres ou extremamente pobres, de acordo com a renda mensal, recebem diferentes quantias em dinheiro e, em contrapartida, devem obedecer a um conjunto de condicionalidades (manter as crianças em escolas, realizar vacinas e exames etc). Somente recebem este benefício aqueles indivíduos que possuem um NIS válido.

Na Saúde Pública, o programa Saúde da Família (PSF) e o Sistema Único de Saúde (SUS) são as duas principais estratégias do governo para a oferta gratuita de serviços de saúde à população. O SUS possui aproximadamente 40 bases de dados[2], as quais englobam desde dados administrativos até dados específicos sobre órgãos, transplantes, internações hospitalares, doenças contagiosas etc. No contexto da epidemiologia, algumas destas bases são particularmente relevantes, conforme descrito na Tabela 1.

Tabela 1. Bases de dados governamentais fornecidas ao projeto.

Bases de dados	Cobertura
SIH (hospitalizações)	1998 a 2011
SINAN (notificações de doenças)	2000 a 2010
SIM (mortalidade)	2000 a 2012
SINASC (nascidos vivos)	2001 a 2012
CadÚnico (dados socioeconômicos)	2007 a 2012 (exceto 2010)
PBF (pagamentos do Bolsa Família)	2007 a 2013

As dificuldades técnicas no uso destas bases estão relacionadas à heterogeneidade dos dados, à inexistência de atributos identificadores comuns (chaves) a todas as bases e à baixa qualidade dos dados. Somam-se a isso os requisitos de segurança e privacidade associados aos dados de saúde pública. A heterogeneidade aparece em atributos que

[2] `http://www2.datasus.gov.br/`

armazenam a mesma informação, mas são nomeados ou modelados de formas diferentes em cada base. A baixa qualidade dos dados é reflexo de erros de digitação ou falta de informações quando do preenchimento de dados nestas bases. Como o SUS provê acesso gratuito à saúde, diversos registros são parcialmente preenchidos com dados de moradores de rua ou de crianças com documentação incompleta, por exemplo.

A inexistência de atributos-chave comuns a todas as bases do SUS obriga que os relacionamentos (correlações) sejam feitos de forma probabilística, com base em atributos que efetivamente identifiquem as pessoas em cada base. A escolha destes atributos é relativamente complexa, visto que alguns atributos têm uma qualidade muito baixa ou podem ser ambíguos (um atributo NOME com muitos valores parecidos, por exemplo), o que não contribui significativamente para a correlação. Outras dificuldades técnicas envolvem a preservação da privacidade e o tratamento de registros duplicados.

A construção da coorte populacional requer a integração das diversas versões da base do Cadastro Único. A Tabela 2 ilustra o resultado preliminar de uma coorte, considerando as bases do Cadastro Único entre 2007 a 2012 (conforme Tabela 1). O volume de dados na coorte é de 103 milhões de registros: 66 milhões de pessoas cadastradas em 2007 acrescidas daquelas que tiveram ao menos um registro nos anos seguintes (até 2012). Se a base de 2010 for inserida na coorte[3], a expectativa é que o total de registros aumente entre 7% e 10%.

Tabela 2. Volume de dados da coorte populacional do CadÚnico.

Ano	Quantidade de pessoas (em milhões)
2007	70.359.183
2008	78.769.510
2009	84.850.684
2011	95.359.028
2012	103.003.121

Este volume de dados da coorte deverá então ser relacionado com as bases do SUS e do Bolsa Família para os estudos epidemiológicos que se deseja conduzir. Entre o CadÚnico e o PBF é possível realizar a correlação determinística com base no NIS. Entretanto, as bases do SUS requerem o uso de métodos probabilísticos, os quais são tarefas computacionalmente intensivas dado o grande número de comparações entre registros que precisam executar.

3. A metodologia de correlação probabilística

A metodologia desenvolvida nesta cooperação emprega técnicas para a análise da qualidade dos dados, a preparação destes dados, a correlação entre registros e a averiguação da acurácia dos resultados. Um *pipelining* foi implementado a partir desta metodologia, conforme ilustra a Figura 1.

Os arquivos de dados são fornecidos num formato CSV criptografado. Cada arquivo (base) é analisado para a escolha dos atributos para a correlação. Estes atributos passam por uma etapa de preparação na qual são executadas rotinas de padronização,

[3]Esta base não foi fornecida pelo MDS pois houve a troca da versão 6 para a 7 do CadÚnico e, com isso, algumas inconsistências nos dados de 2010.

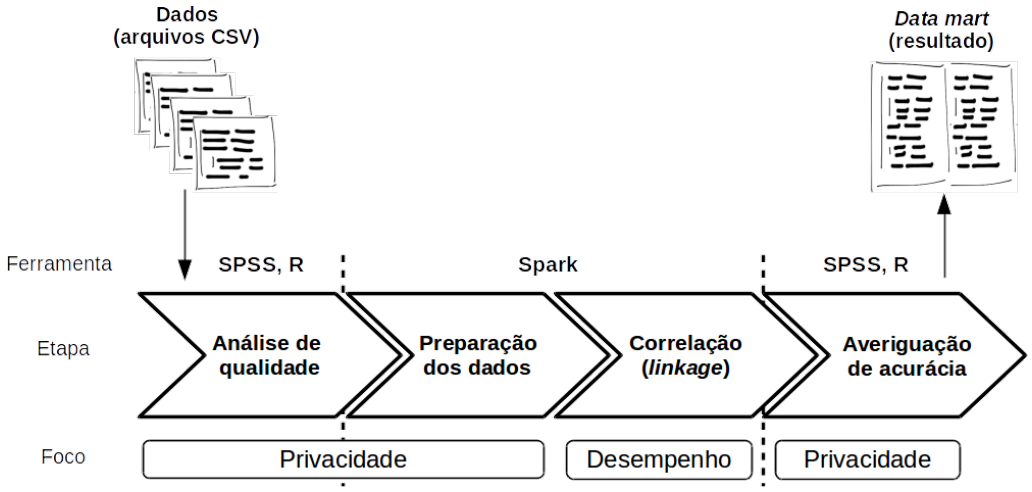


Figura 1. *Pipelining* para a correlação probabilística.

limpeza e anonimização de atributos identificadores. Esta etapa gera também os blocos de registros a serem comparados durante a correlação. Os resultados (*data marts*) passam pela avaliação de acurácia. Se aprovados, estão prontos para uso; do contrário, o *pipelining* deve ser ajustado e executado novamente.

O Spark permite que as etapas computacionalmente intensivas (preparação dos dados e correlação de registros) sejam executadas em curto espaço de tempo sobre um conjunto de recursos computacionais distribuídos. Uma descrição detalhada da implementação deste *pipelining* no Spark é apresentada em [Pita et al. 2015], com um estudo de caso da correlação do Cadastro Único com a base de hospitalizações (SIH).

3.1. Análise da qualidade dos dados

Esta etapa permite conhecer os dados e identificar padrões que auxiliem na escolha dos atributos para a correlação. Os atributos candidatos (chaves, identificadores etc) são analisados em relação à incidência de valores ausentes (*missing*) e aqueles com menores índices são escolhidos. Esta etapa considera ainda outros dois aspectos: a quantidade de atributos a serem usados na correlação e as especificidades de cada base de dados.

A quantidade de atributos influencia diretamente o tempo de processamento e a acurácia dos resultados. Um conjunto maior de atributos podem gerar resultados com melhor acurácia, embora aumente o tempo gasto em comparações durante a correlação. Dada a heterogeneidade das bases, pode ser necessário selecionar um conjunto específico de atributos em cada uma. O desafio passa a ser, portanto, encontrar o conjunto mínimo de atributos comuns entre todas as bases de modo a garantir que os resultados gerados na correlação sejam confiáveis em termos de acurácia.

A análise descritiva realizada nas bases de dados envolvidas no projeto resultou na escolha dos atributos NOME, ULTIMO_NOME, DT_NASC, SEXO e MUNIC_RES para o processo de correlação. Estes atributos apresentam melhor qualidade (pouco ou nenhum valor ausente) e estão presentes em todas as bases do SUS e no CadÚnico. A escolha dos atributos também leva em conta as informações que são importantes para os estudos que se pretende realizar sobre a coorte.

3.2. Preparação dos dados

Os atributos escolhidos na etapa de análise de qualidade passam por rotinas de preparação antes do processo de correlação. Uma primeira rotina corresponde à padronização dos valores destes atributos: substituição dos valores ausentes por valores padronizados, padronização de datas, remoção de caracteres especiais, conversão para maiúsculas e remoção de duplo espaçamento. Após isso, são executadas rotinas para a construção de blocos e para a anonimização de atributos identificadores.

A abordagem padrão para a construção de blocos é agrupar registros que possuam uma mesma chave identificadora no mesmo bloco. Dessa forma, garante-se que serão comparados somente aqueles blocos que possuam chaves idênticas nas diversas bases. Apesar de reduzir a quantidade de comparações, a blocagem pode diminuir a acurácia dos resultados. Os erros de preenchimento podem fazer com que registros que deveriam ficar num mesmo bloco sejam colocados em blocos distintos, nunca sendo comparados.

A metodologia desenvolvida usa duas estratégias de blocagem. No método "tradicional", os atributos NOME e DATA_NASC são utilizados para a construção dos blocos. Para minimizar os efeitos dos erros de preenchimento e tentar garantir que registros semelhantes sejam alocados no mesmo bloco, um método "alternativo" utiliza a blocagem por predicados [Hernandez and Stolfo 1998], o qual considera uma combinação de atributos para compor as chaves que irão gerar os blocos. Assim, a inserção de um registro num determinado bloco pode ser garantida pelo valor contido em um dos atributos que compõem a chave. O predicado usado é *(NOME ˆ MUNIC_RES) v (SOBRENOME ˆ ANO_NASC)*. Em ambos os métodos, os atributos NOME e SOBRENOME são codificados foneticamente com o algoritmo Metaphone [Binstock and Rex 1995].

A anonimização de atributos identificadores é feita através de filtro de Bloom [Bloom 1970], o qual corresponde a um vetor binário de tamanho *n* inicialmente zerado. O valor de cada atributo é decomposto em bigramas (pares de caracteres), os quais passam por um conjunto de funções *hash* que determinam uma posição específica no vetor a ser alterada de 0 para 1. Cada bigrama pode alterar *k* posições, dependendo da quantidade de funções *hash* usadas. Este método apresenta altos níveis de preservação de privacidade e de acurácia, pois não permite falsos positivos: dois conjuntos de atributos exatamente iguais sempre produzirão filtros iguais. O filtro usado na metodologia possui 110 bits (50 bits para NOME, 40 bits para DATA_NASC e 20 para MUNIC_RES) e duas funções *hash*. Uma descrição detalhada da implementação do filtro de Bloom e a avaliação de outros tamanhos e composições do filtro são discutidas em [Pita et al. 2015].

3.3. Correlação de registros

O método clássico de correlação foi proposto em [Fellegi and Sunter 1969] e consiste em usar um conjunto comum de atributos presentes em registros de diferentes bases para determinar se estes registros referem-se a um mesmo objeto. Na metodologia desenvolvida, a correlação é feita de forma determinística entre as bases CadÚnico e PBF (através do NIS) e de forma probabilística entre o CadÚnico e as bases do SUS.

Para verificar a semelhança entre dois registros (filtros de Bloom), deve-se considerar a quantidade de bits 1 e a posição em que ocorrem. Neste trabalho, utiliza-se o Índice de Sorensen (ou Dice), dado por *(2*h / a + b)*, sendo *h* a quantidade de 1's em posições comuns aos dois vetores e *a* e *b* a quantidade de 1's existentes no primeiro e no

segundo vetor, respectivamente. Ao se comparar dois filtros, um valor *Dice=1* é gerado quando os dois vetores forem completamente iguais, decrementando até 0 caso hajam diferenças. Neste trabalho, os valores gerados foram normalizados entre 0 e 10000.

De forma semelhante à construção de blocos, dois métodos de correlação são empregados. No método tradicional, os filtros de Bloom de 110 bits cada são comparados em sua totalidade e um único valor de Dice é gerado. No método alternativo são feitas comparações individuais para cada atributo: SEXO e MUNIC_RES de forma determinística e NOME e DATA_NASC de forma probabilística. As comparações geram valores (exatos ou de Dice), os quais são avaliados com base em tabelas de regras para se decidir sobre o pareamento. Esta estratégia foi adotada no intuito de explorar novas técnicas e melhorar a acurácia dos resultados apresentados pelo método tradicional.

Pontos de corte devem ser definidos para que se possa decidir sobre o pareamento de registros. Em geral, as metodologias de correlação consideram que pares de registros acima de um ponto de corte superior são "verdadeiros positivos", enquanto que pares de registros abaixo de um ponto de corte inferior são "verdadeiros negativos". Todos aqueles pares de registros com índice de semelhança entre os dois pontos de corte são "falsos positivos" ou "falsos negativos", sendo passíveis de verificação manual.

Para todos os pares verdadeiros positivos, os registros completos de ambas as bases de dados são recuperados e agregados num *data mart*, o qual é submetido à análise de acurácia, de modo a validar a eficácia da metodologia implementada. Um vez validado, o *data mart* está pronto para ser usado.

3.4. Análise da acurácia dos resultados

A ausência de um "padrão-ouro" para a correlação probabilística dificulta a escolha dos pontos de corte, pois não existem valores considerados "ótimos" ou adequados a todos os casos. Dois outros fatores tornam esta etapa crítica: i) não se pode inferir previamente o resultado de correlações envolvendo as bases CadÚnico e do SUS para os diferentes desfechos (doenças) que se deseja estudar e ii) não se pode fazer a verificação manual devido ao grande volume de dados das bases.

Dado este contexto, a estratégia adotada para averiguar a acurácia é empregar bases de dados controladas e avaliar diferentes pontos de corte (superior e inferior). Este processo não gera necessariamente um "padrão-ouro", mas garante que a rotina de correlação empregue valores de ponto de corte que gerem os melhores resultados. Uma vez validada, a correlação é gradativamente empregada em bases não-controladas, variando-se a quantidade de registros em cada base e avaliando-se os resultados.

Em um trabalho prévio [Pita et al. 2015], o *pipelining* implementado foi avaliado em termos de eficiência (tempo de execução) na correlação entre as bases do Cadastro Único e de hospitalizações (SIH) para o ano de 2011. Neste artigo, o enfoque está na avaliação da acurácia, considerando métricas tradicionais (sensibilidade, especificidade e valor preditivo positivo) aplicadas a cenários controlados e não-controlados.

4. Estudos de casos

Dois estudos de casos são discutidos: a correlação de bases de dados de rotavírus (cenário controlado) e a correlação do CadÚnico com as bases de hospitalizações (SIH) e de notificações de doenças (SINAN), correspondendo ao cenário não-controlado.

4.1. Bases controladas: rotavírus

Bases controladas são aquelas em que os dados são conhecidos e é possível determinar quantos registros serão combinados na correlação. Os resultados produzidos com bases controladas servem como referência para a análise dos resultados obtidos em bases não-controladas (caso do CadÚnico, PBF e do SUS).

Neste trabalho, a base controlada é composta por 486 registros de crianças atendidas em unidades hospitalares por diarréia e que apresentaram exames laboratoriais com resultados positivos para o rotavírus, somados a 200 registros de outras crianças retirados aleatoriamente de outra base de dados. A segunda base contém 9.678 registros de crianças atendidas por diversos problemas de saúde, inclusive diarréia, em dez unidades hospitalares. O objetivo foi avaliar se a correlação seria capaz de identificar todos os 486 casos conhecidos dentre os 9.678 registros.

Para aproximar do contexto das bases não-controladas, foram simulados quatro cenários com modificações nos atributos NOME e DATA_NASC em diferentes proporções: de 5,15% a 11,3% do total de 486 registros. Foram analisados os dois métodos (tradicional e alternativo) com e sem o uso de blocagem. A quantidade de pares verdadeiramente positivos identificados em cada cenário é ilustrada na Tabela 3.

Tabela 3. Acurácia do cenário controlado: nº de verdadeiros positivos.

	Cenário 1 (10,3%)	**Cenário 2 (11,3%)**	**Cenário 3 (10,3%)**	**Cenário 4 (5,15%)**
Tradicional (sem blocos)	482	481	479	482
Tradicional (com blocos)	444	332	466	458
Alternativo (sem blocos)	482	482	480	486
Alternativo (com blocos)	482	482	472	486

É possível observar que o uso de blocos diminui a acurácia dos resultados (conforme discutido anteriormente), principalmente no método tradicional, para o qual as diferenças observadas em todos os cenários são significativas. Esta interferência é menor no método alternativo, devido ao uso da blocagem por predicados. Neste método, os resultados obtidos foram bastante próximos, evidenciando a eficácia da estratégia de blocagem e da correlação baseada na comparação individual de atributos. É interessante observar que os resultados obtidos no método tradicional sem o uso de blocos se aproximam dos resultados obtidos com o método alternativo, o que evidencia que ambos os métodos são confiáveis e eficazes num contexto de correlação probabilística.

A avaliação da acurácia também objetiva auxiliar na escolha dos pontos de corte para os métodos de correlação. Para tanto, as medidas de sensibilidade e de valor preditivo positivo (VPP) foram calculadas para os quatro cenários simulados. A sensibilidade é definida como a proporção de pares verdadeiros entre os 486 casos conhecidos. O VPP é calculado como a proporção de pares verdadeiros entre a soma dos pares verdadeiros e dos pares falsos. Um alto VPP indica alta proporção de pares verdadeiros encontrados pelo método. Como exemplo, a Tabela 4 ilustra os valores de sensibilidade e VPP para o método tradicional, com e sem o uso de blocos, no cenário 1.

Com *Dice= 8600*, foi encontrada sensibilidade de 91.4% na abordagem com blocagem e 99.0% na abordagem sem blocagem. O valor acima (8800) apresenta sensi-

Tabela 4. Sensibilidade e VPP (método tradicional – cenário 1).

	Com blocagem		Sem blocagem	
Dice	**Sensibilidade (%)**	**VPP (%)**	**Sensibilidade (%)**	**VPP (%)**
10000	69.3	100.0	8.8	100.0
9800	71.2	100.0	12.8	100.0
9600	75.3	100.0	59.5	100.0
9400	79.4	100.0	86.6	100.0
9200	82.3	100.0	95.3	100.0
9000	86.4	100.0	98.1	100.0
8800	91.4	100.0	98.8	100.0
8600	91.4	100.0	99.0	100.0
8400	91.4	100.0	99.2	99.8
8200	91.4	100.0	99.2	99.8
8000	91.4	100.0	99.2	99.8
7000	91.4	100.0	99.2	98.2

bilidade e VPP muito próximos, sugerindo que um ponto de corte entre 8600 e 8800 possa ser escolhido para ambos os métodos. Os demais cenários foram analisados, de modo a escolher um ponto de corte que gerasse os melhores resultados em cada um. A partir daí, o desafio foi definir pontos de corte que pudessem ser aplicados a todos os cenários e gerassem resultados com alta acurácia. Com base nos testes realizados, foram adotados os valores 8700 e 9600 como pontos de corte inferior e superior, respectivamente.

4.2. Bases não-controladas: SIH e SINAN - tuberculose

Neste estudo de caso foram escolhidas as bases de hospitalizações (SIH) e de notificações de doenças (SINAN) para os casos de tuberculose no ano de 2011. A amostragem foi feita com dados do estado de Sergipe, o qual tem o menor número de registros nas bases envolvidas, permitindo que se faça uma verificação manual dos dados e se conheça quantos registros do SIH e do SINAN devem ser encontrados como verdadeiros positivos. O propósito foi tentar estabelecer um "padrão-ouro" com base em uma amostra pequena dos dados e usá-lo na avaliação dos resultados obtidos com amostras maiores (outros estados) e com a totalidade de registros das bases.

Existem 17 casos de internações por tuberculose em Sergipe em 2011. Dessa forma, espera-se que os métodos de correlação entre as bases SIH e CadÚnico identifiquem corretamente estes pares. A Tabela 5 ilustra os resultados obtidos pelo método tradicional, enquanto que a Tabela 6 ilustra os resultados para o método alternativo.

Os resultados obtidos para o método tradicional indicam que um valor *Dice*$\geq$*9.400* é capaz de capturar mais de 80% de pares verdadeiramente positivos em ambos os casos (com e sem blocagem). Somente um par não foi corretamente identificado neste caso. No método alternativo, com um valor *Dice*$\geq$*7.000*, todos os pares esperados são corretamente identificados.

Na base SINAN, existem 227 ocorrências para o estado de Sergipe relativas à tuberculose em 2011. Dessa forma, espera-se que ambos os métodos de correlação identifiquem todos estes pares. As Tabelas 7 e 8 ilustram os resultados.

Para o método tradicional, um valor *Dice*$\geq$*8,800* é capaz de identificar todos os

Tabela 5. SIHxCadÚnico — mét. trad.

Dice	Com blocagem			Sem blocagem		
	Pares ligados	Verd. pos. N	Verd. pos. %	Pares ligados	Verd. pos. N	Verd. pos. %
≤10000	1	1	100,0	1	1	100,0
≥9800	12	12	100,0	12	12	100,0
≥9600	15	14	93,3	15	14	93,3
≥9400	18	15	83,3	18	15	83,3
≥9200	22	16	72,7	23	16	69,6
≥9000	24	16	66,7	25	16	64,0
≥8800	32	16	50,0	32	16	50,0

Tabela 6. SIHxCadÚnico — mét. alt.

Dice	Com blocagem			Sem blocagem		
	Pares ligados	Verd. pos. N	Verd. pos. %	Pares ligados	Verd. pos. N	Verd. pos. %
≥8900	16	16	100,0	16	16	100,0
≥7000	34	17	50,0	43	17	39,5

Tabela 7. SINANxCadÚnico — mét. trad.

Dice	Com blocagem			Sem blocagem		
	Pares ligados	Verd. pos. N	Verd. pos. %	Pares ligados	Verd. pos. N	Verd. pos. %
≥9800	18	18	100,0	18	18	100,0
≥9600	128	128	100,0	128	128	100,0
≥9400	174	172	98,9	178	173	97,2
≥9200	214	204	95,3	226	206	91,2
≥9000	254	217	85,4	288	220	76,4
≥8800	327	226	69,1	385	227	59,0

Tabela 8. SINANxCadÚnico — mét. alt.

Dice	Com blocagem			Sem blocagem		
	Pares ligados	Verd. pos. N	Verd. pos. %	Pares ligados	Verd. pos. N	Verd. pos. %
≥8900	169	169	100,0	172	171	99,4

227 pares quando não se usa a blocagem. Somente um par não é identificado com o uso de blocagem. No método alternativo, uma média de 25% dos pares esperados não são identificados em ambos os casos (com e sem blocagem), o que evidencia a necessidade de se usar valores de ponto de corte mais baixos na expectativa de se identificar uma quantidade maior de pares verdadeiramente positivos.

Ambos os métodos foram avaliados para as métricas de sensibilidade e VPP para as duas correlações realizadas (SIH e SINAN). Em ambos os casos, o uso de blocagem apresentou resultados muito semelhantes às execuções sem blocagem. Novamente, o objetivo foi auxiliar na escolha de pontos de corte que pudessem ser aplicados em ambos os métodos na correlação entre quaisquer bases do SUS e a base do Cadastro Único.

A diferença observada nos valores dos pontos de corte para cada método de correlação evidencia a complexidade de se estabelecer um "padrão-ouro" de comparação e pontos de corte (inferior e superior) únicos para todos os casos. Com base nos resultados obtidos no cenário controlado, os mesmos valores (8700 e 9600) vem sendo adotados no cenário não-controlado para a correlação da totalidade dos registros das bases SIH e SINAN, bem como de outras bases (SIM e SINASC).

5. Trabalhos relacionados

Na Saúde, diversos trabalhos utilizam a correlação para avaliar o impacto de políticas ou encontrar padrões nos dados. Em [Teixeira et al. 2006], registros do SIH e do SIM foram correlacionados de forma probabilística para a identificação de mortes por causas mal definidas. Foi usada a blocagem por chaves e o RecLink [Camargo Jr. and Coeli 2006] para a reclassificação de registros dúbios.

Trabalhos anteriores avaliaram a efetividade dos programas Bolsa Família e Saúde da Família na mortalidade infantil [Rasella et al. 2013] e na ocorrência de hanseníase [Nery et al. 2014]. Porém, tais trabalhos utilizaram estudos "ecológicos" (por amostragem) em vez de coortes, além de ferramentas tradicionais de estatística (SPSS e Stata) sobre conjuntos de dados relativamente pequenos.

A paralelização de algoritmos de correlação é discutida em [Santos 2008], demonstrando bons resultados de escalabilidade. Abordagens baseadas em MapReduce têm contribuído para o suporte de aplicações de correlação, mas ainda em caráter muito inicial ([Huang 2011], [Kolb et al. 2012] e [Merelli et al. 2014]).

Abordagens computacionais para a preparação de dados são ainda pouco discutidas na literatura, embora afetem diretamente a acurácia. Em [Randall et al. 2013], os autores argumentam que a etapa de limpeza pode representar até 75% do esforço total da correlação. A preservação da privacidade, com enfoque em filtros de Bloom, é discutida em [Schnell et al. 2009]. Aspectos de acurácia na correlação de dados anonimizados são discutidos em [Durham et al. 2012] e [Hagger-Johnson et al. 2014].

Além do RecLink, outras ferramentas de dados podem ser destacadas. O German RLC[4] provê diversas ferramentas para correlação: o Merge Toolbox provê deduplicação e pareamento probabilístico com base na técnica de Distância de Edição para determinar a similaridade de registros. O SafeLink fornece um protocolo para a geração de filtros de Bloom a partir de atributos de entrada e o TDGen permite a inclusão de erros nos dados para a validação de métodos de correlação. Por fim, o CALIBER [Denaxas et al. 2012] é uma plataforma que implementa a correlação determinística de registros eletrônicos de bases do governo britânico para suportar estudos baseados em coortes.

Em relação a bases de dados gerenciadas por setores públicos, especificamente no Brasil, pode-se observar a inexistência de referências de correlação probabilística sobre grandes volumes de dados as quais se beneficiem de ferramentas para *big data*, como é o caso do presente trabalho. Do ponto de vista epidemiológico, a principal contribuição é o suporte para estudos baseados em coortes populacionais, as quais permitem o acompanhamento de cada indíviduo da coorte de forma isolada.

6. Conclusões e trabalhos em andamento

A abordagem em *pipelining* empregada neste trabalho procura abordar as principais questões relacionadas à integração de dados da área da Saúde para o suporte de estudos epidemiológicos. Entretanto, a concepção desta ferramenta procura ser genérica o suficiente para permitir a sua adaptação a dados e aplicações de outras áreas.

Os métodos empregados neste trabalho procuram conciliar requisitos de desempenho, segurança e acurácia. O desempenho é resultado da quantidade de atributos usados no filtro de Bloom e da forma como são comparados (em conjunto ou individualmente), do emprego ou não de blocagem e da ferramenta usada para a correlação. A segurança é provida pela anonimização gerada no filtro de Bloom e que se demonstra bastante confiável se comparada a outras abordagens [Schnell et al. 2009]. A acurácia depende dos mesmos fatores que influenciam o desempenho, mas tem uma relação direta com os pontos de corte empregados no cálculo de similaridade. A inexistência de

[4] `http://www.record-linkage.de/`

um "padrão-ouro" para a correlação probabilística torna a análise de acurácia uma etapa fundamental.

Um estudo de caso envolvendo as bases CadÚnico, PBF e SIH foi discutido em [Pita et al. 2015]. A partir destes resultados, o grupo começou a avaliar a acurácia e a realizar melhorias na implementação, algumas delas discutidas neste artigo. O uso de blocagem, apesar de favorecer o tempo ao reduzir a quantidade de comparações, também pode reduzir os níveis de sensibilidade e VPP para alguns casos; o que deve ser cuidadosamente avaliado. A correlação de grandes volumes de dados sem o emprego de blocagem demanda grande capacidade computacional (memória e processamento), o que pode tornar o uso de blocos obrigatório.

A correlação das bases CadÚnico, PBF, SIH, SIM, SINAN e SINASC, para o ano de 2011, está passando pela análise de acurácia com base nos dois métodos descritos. O grupo está avaliando o uso de descritores (metadados) para cada base, de modo a permitir que os métodos de correlação possam inferir os melhores valores de ponto de corte para cada caso. A modelagem dos dados e a escolha de ferramentas para a implementação da coorte populacional descrita na Tabela 2 estão sendo desenvolvidas.

7. Agradecimentos e fomento

Os autores agradecem ao Ministério da Saúde e ao Ministério do Desenvolvimento Social e Combate à Fome pelo fornecimento dos dados. Este projeto é financiado pela Fundação de Amparo à Pesquisa do Estado da Bahia: edital FAPESB-PPSUS 30/2013; pela UFBA: editais PRODOC 2013 e PIBIC 2014; e pela CAPES: edital *Grand Challenge Brazil* – Fundação Bill & Melinda Gates e Programa *Newton Fund – Institutional Links*.

Referências

Binstock, A. and Rex, J. (1995). *Practical Algorithms for Programmers*. Addison Wesley, 1st edition.

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426.

Camargo Jr., K. R. and Coeli, C. M. (2006). RecLink 3: nova versão do programa que implementa a técnica de associação probabilística de registros (*probabilistic record linkage*). *Cadernos de Saúde Coletiva*, 14(02):399–404.

Denaxas, S., George, J., Herrett, E., Shah, A., et al. (2012). Data resource profile: cardiovascular disease research using linked bespoke studies and electronic health records (CALIBER). *International Journal of Epidemiology*, 41:1625–1638.

Durham, E., Xue, Y., Kantarcioglu, M., and Malin, B. (2012). Quantifying the correctness, computational complexity, and security of privacy-preserving string comparators for record linkage. *Information Fusion*, 13(4):245–259.

Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210.

Hagger-Johnson, G., Harron, K., Gonzalez-Izquierdo, A., Cortina-Borja, M., Dattani, N., Muller-Pebody, B., Parslow, R., Gilbert, R., and Goldstein, H. (2014). Identifying possible false matches in anonymized hospital administrative data without patient identifiers. *Health Services Research*, pages n/a–n/a.

Hernandez, M. A. and Stolfo, S. J. (1998). Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Min. Knowl. Discov.*, 2(1):9–37.

Huang, Y. (2011). Record linkage in an Hadoop environment. Technical report, School of Computing, National University of Singapore.

Keyes, K. M. and Galea, S. (2014). *Epidemiology matters: a new introduction to methodological foundations*. Oxford University Press, 1st edition.

Kolb, L., Thor, A., and Rahm, E. (2012). Dedoop: Efficient deduplication with Hadoop. *Proc. VLDB Endow.*, 5(12):1878–1881.

MDS (2015a). Cadastro único para programas sociais do governo federal. [Acesso em 12/05/2015].

MDS (2015b). Programa Bolsa Família. [Acesso em 12/05/2015].

Merelli, I. M., Pérez-Sánchez, H., Gesing, S., and Dagostino, D. (2014). Managing, analysing, and integrating big data in medical bioinformatics: open problems and future perspectives. *BioMed Research International*, 2014(1).

Nery, J. S., Pereira, S. M., Rasella, D., Penna, M. L. F., Aquino, R., Rodrigues, Laura C., M. L., and Penna, G. O. (2014). Effect of the Brazilian conditional cash transfer and primary health care programs on the new case detection rate of leprosy. *PLOS Neglected Tropical Diseases*, 8(11).

Pita, R., Pinto, C., Melo, P., Silva, M., Barreto, M., and Rasella, D. (2015). A Spark-based workflow for probabilistic record linkage of healthcare data. In *Proceedings of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), Brussels, Belgium, March 27th, 2015.*, pages 17–26.

Randall, S. M., Ferrante, A. M., and Semmens, J. (2013). The effect of data cleaning on record linkage quality. *BMC Medical Informatics and Decision Making*, 13.

Rasella, D., Aquino, R., Santos, C. A. T., Paes-Sousa, R., and Barreto, M. L. (2013). Effect of a conditional cash transfer programme on childhood mortality: a nationwide analysis of Brazilian municipalities. *The Lancet*, 382(9886):57–64.

Santos, W. (2008). Um algoritmo paralelo e eficiente para o problema de pareamento de dados. Mestrado, Universidade Federal de Minas Gerais.

Schnell, R., Bachteler, T., and Reiher, J. (2009). Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics and Decision Making*, 9(41).

Teixeira, C. L. S., Klein, C. H., Bloch, K. V., and Coeli, C. M. (2006). Reclassificação dos grupos de causas prováveis dos óbitos de causa mal definida, com base nas Autorizações de Internação Hospitalar no Sistema Único de Saúde, Estado do Rio de Janeiro, Brasil. *Cad. Saúde Pública*, 22(6):1315–1324.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA. USENIX Association.

sbbd:07

Definição de Planos de Execução Distribuídos para Consultas de Junção Espacial usando Histogramas Multidimensionais

Thiago B. de Oliveira[1], **Fábio M. Costa**[1], **Vagner J. do S. Rodrigues**[1]

[1]Instituto de Informática (INF) – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brasil

thborges@ufg.br, {fmc,vsacramento}@inf.ufg.br

Abstract. *Multiway spatial join is a common and heavy query in spatial data processing. This paper presents a cost-based optimizer for multiway spatial queries, based on a novel multidimensional histogram. The histogram is composed by three dimensions: the cardinality, the size of spatial objects and the locality of partitions on a cluster. An algorithm to define efficient distributed execution plans based on these dimensions is proposed and evaluated.*

Resumo. *A junção espacial complexa é uma das consultas mais empregadas e que mais consomem recursos computacionais no processamento de dados espaciais. Este artigo propõe um otimizador de consultas distribuídas de junção espacial que utiliza histogramas com múltiplas dimensões. O histograma é composto por três dimensões: cardinalidade, tamanho dos objetos espaciais e localidade das partições no* cluster. *Um algoritmo para definir planos de execução eficientes baseados nas três dimensões é proposto e avaliado.*

1. Introdução

A quantidade de dados espaciais como imagens geolocalizadas, dados abertos dos governos federais, estaduais e municipais, mapeamento de lojas comerciais e levantamento de dados georreferenciados por entidades governamentais, dentre outros, têm aumentado devido à popularização de dispositivos computacionais dotados de GPS (*Global Positioning System*) e rede de comunicação como celulares, *smartphones* e sensores.

Estes dados são armazenados em sistemas de geoprocessamento, como bancos de dados espaciais e Sistemas de Informação Geográfica (SIG [Bernhardsen 2002]), e processados empregando algoritmos chamados de consultas espaciais. Uma importante consulta é a junção espacial (*Spatial Join*). A junção espacial é um tipo de consulta que encontra elementos correlacionadas em dois ou mais datasets, de acordo com um predicado espacial, como interseção ou proximidade [Brinkhoff et al. 1996]. É denominada *simples* se envolve apenas dois datasets, ou junção espacial *complexa* (*Multiway Spatial Join*), quando envolve três ou mais datasets [Mamoulis and Papadias 2001b].

Uma junção espacial complexa pode ser processada de várias formas diferentes, chamadas planos de execução. Devido a quantidade de planos ser não linear em relação à quantidade de datasets, emprega-se um otimizador de consultas heurístico para escolher um bom plano de execução. Um otimizador de consultas para sistemas distribuídos deve considerar o particionamento dos dados e comunicação entre os servidores, devido ao significativo impacto que esses parâmetros exercem no tempo de resposta dos algoritmos.

Este trabalho foi parcialmente financiado com recursos do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, processo número 473.939/2012-6.

Este artigo apresenta uma proposta de otimizador de consultas espaciais complexas distribuídas. As principais contribuições são:

- Identificação das características dos datasets e da distribuição dos dados, relevantes para o processamento eficiente de junções complexas em sistemas distribuídos;
- Definição de histogramas multidimensionais para organizar essas características, observando a irregularidade (*skewness*) dos datasets espaciais reais;
- Um algoritmo que estima o custo dos planos de execução em ambientes distribuídos, utilizando os histogramas multidimensionais;
- Um novo método de construção da dimensão espacial de um histograma, que melhora a estimativa de custo do processamento das consultas; e
- Um algoritmo guloso de escalonamento de cópias de dados entre os nós do *cluster*, que reduz a quantidade de cópias enquanto preserva o balanceamento.

O restante do artigo está organizado da seguinte forma: Na Seção 2, há uma revisão da literatura sobre processamento de consultas espaciais, métodos de estimativa de custo para escolha de planos de execução e algoritmos distribuídos para o processamento das consultas. Na Seção 3, o conceito de histograma multidimensional é apresentado, juntamente com o levantamento das características relevantes para estimar o custo e definir bons planos de execução. Essa seção também apresenta um algoritmo guloso que escalona as cópias de partições dos datasets e um método heurístico que combina as dimensões dos histogramas para escolha dos planos distribuídos. Na Seção 4, o processo de avaliação dos algoritmos e técnicas propostas é apresentado e os resultados discutidos. Na Seção 5 listamos trabalhos correlatos à nossa proposta e na Seção 6, a conclusão e trabalhos futuros são apresentados.

2. Processamento de Junções Espaciais Complexas

De acordo com [Papadias et al. 1999, Mamoulis and Papadias 2001b], uma junção espacial complexa pode ser representada por um grafo onde cada vértice *v* representa um dataset. A existência de uma aresta (u, v) entre dois vértices *u* e *v* determina que os datasets devem ser correlacionados como parte do processamento da consulta.

Vários planos de execução podem ser especificados para uma consulta. A Figura 1 apresenta uma consulta (1a) e três planos de execução distintos (1b, 1c e 1d), cada um com uma ordem de junção e algoritmos ($\bowtie^a, \bowtie^b, \bowtie^c$ *e* $\bowtie^d$) específicos, determinados a partir da quantidade de datasets e dos resultados intermediários gerados. Os planos preservam a semântica da consulta, mas podem incorrer em tempos diferentes de processamento.

O objetivo de um otimizador de consultas é determinar um plano de execução que reduz o consumo de recursos computacionais e o tempo de resposta da consulta. Uma forma de realizar essa escolha é determinar o custo da consulta utilizando equações que estimam o custo computacional, conforme será detalhado na próxima seção.

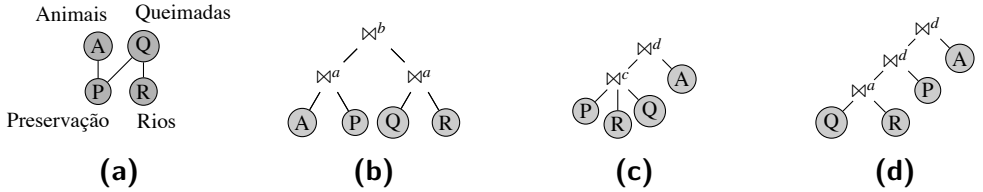


Figura 1. a) Grafo de junção complexa e três planos de execução: (b), (c) e (d).

2.1. Métodos de Estimativa de Custo de Junções Espaciais

Como é complexo determinar o custo computacional de uma consulta antes do processamento, alguns trabalhos definiram equações para estimar o custo computacional de junções espaciais simples [Acharya et al. 1999, Sivasubramaniam 2001] e também junções espaciais complexas [Mamoulis and Papadias 2001a]. Estas equações consideram que o espaço geográfico é preenchido uniformemente pelos objetos e, em um trabalho posterior [Mamoulis and Papadias 2001b], os mesmos autores concluem que estas equações, quando usadas em datasets reais, provocam a escolha de planos de execução ruins.

Uma alternativa proposta por [Mamoulis and Papadias 2001b] foi o uso destas equações em histogramas com duas dimensões espaciais, que representam uma aproximação da densidade do dataset de forma mais granular. O histograma representado na Figura 2b divide o dataset (Figura 2a) em 2500 células de tamanho fixo (50x50). Para construir o histograma, os limites de cada célula são definidos a partir da extensão espacial do dataset. A densidade $D(x)$ é calculada somando a quantidade de objetos cujo centroide está colocalizado com a célula x. A seletividade S de uma junção espacial pode ser estimada a partir dos histogramas (H_r e H_s) de dois datasets, R e S, através da equação $S = \sum D(c_i) * D(c_j) \; \forall \; c_i \in H_r, \; c_j \in H_s | c_i \cap c_j \neq \emptyset$.

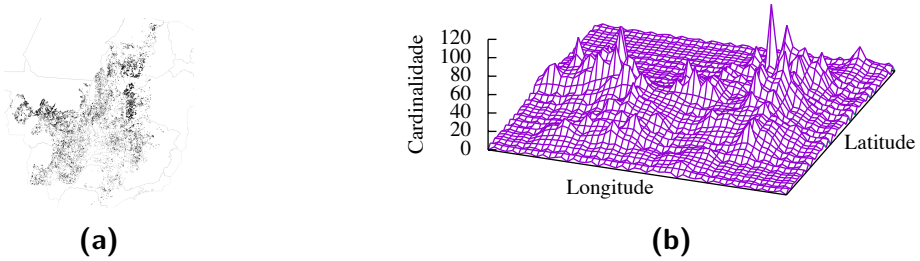


Figura 2. Dataset de desmatamento do cerrado (a). Histograma para o dataset (b).

Em nossos experimentos, constatamos que o centroide, por ser uma informação simplificada de objetos espaciais do tipo polígonos e linhas, conduz a estimativas com erro substancial em relação a execução real e, portanto, neste artigo, propomos e avaliamos um novo método de construção da dimensão espacial, que será detalhado na Seção 3.1.

Existem também outras propostas de histogramas na literatura, que dividem a extensão espacial em uma grade de células de tamanho variável, como *Min-Skew* [Acharya et al. 1999]. Este histograma pode ser empregado para melhorar a estimativa em datasets desuniformes, porém, a complexidade de determinar a seletividade é maior em relação ao histograma de grade com células de tamanho fixo. Escolhemos usar o método da grade fixa devido a sua simplicidade e deixamos para trabalhos futuros esta avaliação.

Em sistemas distribuídos, além da cardinalidade, o volume de comunicação é um parâmetro importante para medir o custo de uma junção espacial complexa. Para determinar este volume é necessário considerar o tamanho dos objetos espaciais (quantidade de pontos) e a técnica de particionamento dos dados, descrita na Seção 2.2.

2.2. Algoritmo para Processamento da Junção Espacial Distribuída

Um otimizador necessita considerar quais algoritmos de junção espacial serão utilizados no processamento das consultas. Em nossa revisão, encontramos algoritmos distribuídos específicos para os seguintes cenários: *i*) Dois datasets indexados: *Replicated Semi-packed Parallel R-Tree* (RSPR) [Mutenda and Kitsuregawa 1999] e *Proximity Area Spa-*

tial Join (PASJ) [de Oliveira et al. 2013]; *ii*) Dois resultados intermediários: *Clone Join* (CJ) [Patel and DeWitt 2000], *Shadow Join* (SJ) [Patel and DeWitt 2000] e *Non-blocking Parallel Spatial Join* (NBPS) [Naughton and Ellmann 2002], e *iii*) Dois ou mais datasets indexados: *Distributed Synchronous Traversal* (DST) [Cunha et al. 2015].

Estes algoritmos também podem ser classificados de acordo com o particionamento dos dados: *i*) particionamento irregular, que divide a extensão espacial do dataset de forma não regular, onde uma partição pode sobrepor outra (RSPR, PASJ, DST), e *ii*) particionamento regular, dividindo a extensão espacial com uma grade de células disjuntas, de mesmo tamanho (CJ, SJ, NBPS). Foi demonstrado em [Zhou et al. 1998] que a técnica de particionamento regular é mais eficiente em sistemas distribuídos, devido ao aumento na quantidade de comparações no processamento do predicado da consulta, causado pelas áreas sobrepostas no particionamento irregular.

Além do particionamento dos dados, é necessário escalonar as partições entre os computadores do *cluster*. O critério de distribuição mais empregado na literatura é o método *Round-Robin*, que privilegia o balanceamento das partições entre os servidores, evitando que várias partições se concentrem em um só servidor e forme um *hotspot* de processamento, problema crítico em sistemas distribuídos. Métodos que privilegiam a redução da comunicação em detrimento do balanceamento também foram propostos, como o método *Proximity Area* [de Oliveira et al. 2013]. O método *Round-Robin* não é uma boa estratátegia de distribuição para junção espacial, mas gera um cenário propício para testes, forçando o otimizador escolher que partições copiar devido à distribuição uniforme e não colocalizada.

Implementamos uma versão modificada do algoritmo CJ que previne a geração de resultados duplicados, ao invés de incluir um operador de distinção no final. A técnica empregada foi o Método do Ponto de Referência, proposta por [Dittrich and Seeger 2000] e também usada no algoritmo NBPS. Esta implementação híbrida consegue se beneficiar da simplicidade do algoritmo CJ e não necessita da etapa de eliminação de resultados duplicados no final. Além do mais, esta modificação torna o CJ um algoritmo não bloqueante, como o NBPS. Algoritmos não bloqueantes formam um *pipeline* durante a execução de junções complexas, reduzindo o tempo total de processamento.

Consideramos que a distribuição das partições é realizada em uma etapa anterior ao processamento das consultas, no momento da inclusão dos dados no sistema. Nesse momento, os histogramas são gerados e os dados são distribuídos. O histograma é usado como método de distribuição e acesso, ou seja, cada servidor do *cluster* armazenará uma ou mais células, formando um índice distribuído. Empregamos a técnica de distribuição de células baseada em *round-robin*, devido ao balanceamento do processamento no *cluster* proporcionado, evitando o surgimento de *hotspots*.

2.3. Seleção de Planos de Execução para Junções Complexas

A escolha do plano de execução da junção complexa pode ser realizada através da enumeração de todos os planos de execução que mantêm a semântica da consulta espacial, e do uso das estimativas de custo de cada junção simples, em etapas sucessivas, considerando o resultado intermediário como um novo dataset em cada etapa. O plano com menor custo total é selecionado para processamento da consulta. Este tipo de otimizador é chamado de otimizador baseado em custo (*Cost-based optimizer*).

Em consultas árvore (*Chain query*), existem $P_n = 1 + 2P_{(n-1)} + \sum_{2 \leq k \leq n-1} P_k P_{(n-k)}$ planos de execução possíveis para n datasets e $P_2 = 1$. Consultas com ciclos (*Cycle Query*) e consultas clique (*Clique Query*) possuem ainda mais planos possíveis. Um algoritmo exaustivo avalia todos os planos possíveis para determinar o melhor plano de execução. Como o número de planos possíveis não é linear em relação ao número de datasets da junção, um método heurístico foi proposto em [Mamoulis and Papadias 2001b], o qual emprega um algoritmo de busca randômico e pode ser usado em consultas onde o tempo de resposta é comprometido pela busca exaustiva. Neste artigo implementamos o método exaustivo e deixamos a avaliação do método heurístico para trabalhos futuros.

3. Histograma Multidimensional e Otimizador Distribuído

Descrevemos nesta seção a combinação dos parâmetros dos datasets e da distribuição dos dados, identificados na revisão da literatura, na estrutura de dados e método de acesso distribuído que denominamos histograma multidimensional.

Um histograma multidimensional é uma estrutura de dados que divide a extensão espacial do dataset utilizando uma grade de células de tamanho fixo e mapeia para cada uma delas: *i*) a quantidade de objetos espaciais colocalizados (cardinalidade), *ii*) qual o tamanho destes objetos (quantidade de pontos) e *iii*) a localidade dos objetos, ou seja, em quais computadores estão ou serão armazenados. Essas informações permitem realizar uma estimativa da comunicação durante a execução de um algoritmo de junção espacial complexa distribuído, conforme será detalhado na Seção 3.2.

Uma representação visual dessas três dimensões para o dataset de limites políticos dos municípios brasileiros pode ser vista na Figura 3. No canto superior esquerdo das figuras 3a e 3b é possível observar que, apesar de existirem poucos objetos, a quantidade de pontos desses objetos é bem superior. Estes são os municípios da região amazônica, maiores em território, e menores em quantidade. A Figura 3c é um mapa da localidade dos objetos, ou seja, cada tom de cinza diferente representa um servidor. Os dados foram distribuídos em oito servidores, numerados de 1 a 8 conforme a legenda. O zero indica que a célula não tem objetos e não foi posicionada em um servidor.

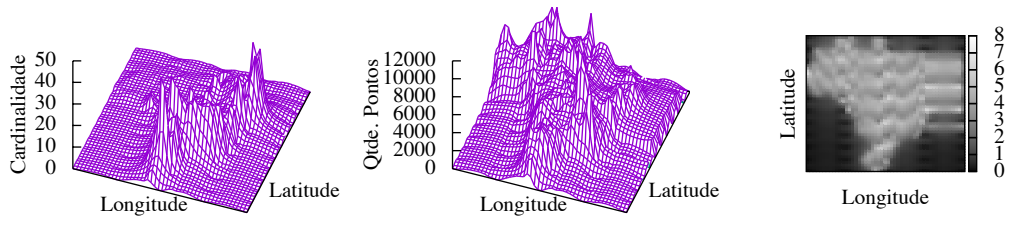


Figura 3. Três dimensões do histograma para o dataset de Municípios Brasileiros.

3.1. Sobreposição Proporcional na Construção do Histograma

A técnica de sobreposição proporcional incrementa o valor de cada célula do histograma, na dimensão cardinalidade, de acordo com a proporção entre a área da interseção do MBR do objeto espacial com a célula, dividida pela área total do MBR do objeto. Formalmente, seja H_r o histograma do dataset R, $C(x)$ a cardinalidade da célula $x \in H_r$ e $o \in R$ um objeto espacial do dataset, incrementa-se o valor de $C(x)$ usando a seguinte equação: $C(x) = C(x) + area(MBR(o) \cap MBR(x))/area(MBR(o)) \ \forall \ x \in H_r, \ o \in R$. Ao contrário

da técnica de centroide (Seção 2.1), que soma 1 em apenas uma célula do histograma para cada objeto, esta técnica incrementa a cardinalidade em todas as células que o objeto sobrepõe, usando a fração de sobreposição. Isto reflete melhor a cardinalidade nas células para objetos do tipo polígono e linha, quando estes objetos sobrepõem várias células.

Na construção da dimensão pontos do histograma, a quantidade de pontos de cada objeto espacial é somada em todas as células onde há sobreposição. Isto reflete o comportamento do algoritmo CJ, que clona os objetos em todas as células que o mesmo sobrepõe. Formalmente, seja $P(x)$ a quantidade de pontos de todos os objetos espaciais colocalizados na célula x, incrementa-se a quantidade de pontos da seguinte forma: $P(x) = P(x) + Pontos(o)\ \forall\ x \in H_r,\ o \in R \mid MBR(o) \cap x \neq \emptyset$. No método do centroide, o total de pontos é somado em apenas uma célula, o que provocaria um erro na estimativa do custo de comunicação deste algoritmo.

3.2. Estimativa de Comunicação

A comunicação de uma etapa do plano de execução pode ser estimada a partir das três dimensões do histograma da seguinte forma: sejam H_a e H_b dois histogramas para os datasets a e b, respectivamente, $F(c_x, c_y) = area(c_x \cap c_y)/area(c_x)$, $C(c_x)$ a cardinalidade da célula c_x, $P(c_x)$ a quantidade de pontos na célula c_x, e $L(c_x)$ a função que retorna o conjunto de servidores nos quais a célula c_x está armazenada, o custo de comunicação estimado é dado pelo Algoritmo 1.

O Algoritmo 1 estima a densidade e comunicação de uma junção entre os dois datasets representados pelos histogramas multidimensionais H_a e H_b, e gera um histograma intermediário H_r. Para construir o histograma H_r, cada célula de H_a e H_b, que possuem interseção entre si, são avaliadas. A cardinalidade e quantidade de pontos são definidos proporcionalmente à interseção das células, de acordo com a função F (Linhas 5 a 10).

De acordo com a localidade de cada célula (servidor que a armazena), estima-se a comunicação mais vantajosa (Linhas 15 a 23): copiar c_a para c_b ou vice-versa, de acordo com a quantidade de pontos a serem transmitidos. A estimativa de comunicação é calculada separadamente para cada servidor, de forma que seja possível mensurar o desvio padrão da carga (balanceamento do *cluster*). O histograma H_r resultante é usado na estimativa de E/S das etapas seguintes do plano de execução.

3.3. Seleção de Planos de Execução Distribuídos

A escolha do plano de execução deve observar o balanceamento de carga do *cluster*. Como o processamento do predicado espacial emprega algoritmos *cpu-bound* de Geometria Computacional, uma execução bem balanceada consegue aproveitar melhor o poder de processamento do *cluster* e isto provoca a redução do tempo de resposta da consulta.

Se um plano de execução faz com que um servidor do *cluster* concentre uma maior parte da cópia de partições, este se tornará tanto um gargalo de E/S de rede como um *hotspot* de processamento, haja visto que processará o predicado espacial de todos os dados copiados. O uso do algoritmo *round-robin* durante a distribuição das partições contribui para que a quantidade de cópias de partições em cada servidor seja uniforme. Caso seja utilizado um outro mecanismo de distribuição, que privilegie a colocalização de células de datasets diferentes e reduza a carga de E/S, seria necessário considerar a quantidade de células copiadas mais a quantidade de células já presentes em cada servidor, devido ao comportamento *cpu-bound* dos algoritmos de verificação do predicado da junção espacial.

```
CommunicationCost(H_a, H_b)
 1  H_r ← novo histograma, baseado nas dimensões de H_a e H_b
 2  ObjIO[servers] ← 0
 3  PntsIO[servers] ← 0
 4  for each ca ∈ H_a, cb ∈ H_b, such that ca ∩ cb ≠ ∅
 5       do D_a ← F(ca,cb) * D(ca)
 6          D_b ← F(cb,ca) * D(cb)
 7          if D_a < 1 or D_b < 1
 8            then continue
 9          P_a ← P(ca)/D(ca) * D_a
10          P_b ← P(cb)/D(ca) * D_b
11          cr ← célula de H_r, baseada nos limites de ca,cb
12          D(cr) ← D_a * D_b
13          P(cr) ← P_a ou P_b, baseado no predicado da próxima etapa
14          ▷ Define o servidor s para a célula do novo histograma H_r
15          if L(ca) ∩ L(cb) ≠ ∅
16            then L(cr) ← L(ca) ∩ L(cb)
17          elseif P_a < P_b
18            then s ← L(cb), L(cr) ← s
19                 PntsIO[s] ← PntsIO[s] + P_a
20                 ObjIO[s] ← ObjIO[s] + D_a * D_b
21            else s ← L(ca), L(cr) ← s
22                 PntsIO[s] ← PntsIO[s] + P_b
23                 ObjIO[s] ← ObjIO[s] + D_a * D_b
24  return ObjIO, PntsIO, H_r
```

Algoritmo 1. Estimativa do custo de uma junção espacial e construção de H_r.

Para determinar o melhor plano de execução de uma consulta, enumera-se todos os planos possíveis, e executa-se o Algoritmo 1 em cada etapa de cada um dos planos, acumulando os resultados de E/S de cada servidor em cada etapa. Para consultas com uma quantidade grande de datasets é mais viável utilizar a técnica citada na Seção 2.3 para reduzir a quantidade de planos candidatos avaliados.

Propomos, portanto, que o melhor plano de execução distribuído seja o plano que possui o menor valor máximo de comunicação para um servidor, ou seja, o minimax da transferência de objetos, que é definido formalmente como se segue: Seja P o conjunto de planos candidatos e $c(p) = \{s_1, s_2, ..., s_n\}$ uma função que retorna o conjunto de custos para cada servidor, $s_{1..n}$, em um plano $p \in P$, o melhor plano tem custo O:

$$O = Min(\{Max(c(p)) \; \forall \; p \in P\}) \quad (1)$$

4. Avaliação

Para avaliar nossa proposta escolhemos um conjunto de datasets reais, obtidos no site do IBGE (`www.ibge.org`), do Laboratório LAPIG do Instituto de Estudos Sócio-Ambientais da UFG (`www.lapig.iesa.ufg.br/lapig/`) e do *Digital Chart of the World* (`http://gis-lab.info/qa/vmap0-eng.html`). Os datasets empregados e suas características estão descritos na Tabela 1. Datasets com cardinalidade pequena foram usados para testar cenários de plano de execução onde um dos datasets restringe completamente a seletividade da consulta, retornando nenhum ou quase nenhum resultado. Os datasets com objetos mais complexos, do tipo linha, foram escolhidos devido a dificuldade e inexatidão de sua representação nos histogramas.

As consultas usadas nos experimentos são listadas na Tabela 2. Todas são consultas reais de junção espacial complexa, envolvendo todos os datasets. As consultas são de cadeia, e o predicado espacial de cada junção é a interseção com o primeiro dataset. As consultas com três datasets possuem dois planos de execução cada, que preservam a sua

semântica. As consultas com quatro datasets possuem seis planos de execução cada. Os planos de execução serão apresentados no formato $Q1..8\ P_{1..6}$, por exemplo, $Q1\ P_1$, $Q1\ P_2$.

Os métodos e algoritmos propostos foram implementados nas linguagens C e Go, utilizando a biblioteca GEOS para processamentos do predicado das consultas. Os resultados a seguir foram obtidos executando o sistema em 8 máquinas virtuais A2, com 2 vCPUs cada, e 3,5 GB de RAM, interligadas em uma rede virtual, na nuvem Azure. Para evitar distorções com máquinas vizinhas na plataforma, cada experimento foi executado cinco vezes. O maior e menor valor foram descartados e a média dos outros três foi utilizada.

Tabela 1. Datasets utilizados nos testes.

Nome	Sigla	Tipo	Cardinalidade	Tamanho SHP (MB)
Portos	*P*	Pontos	120	<1,0
Vegetação	*V*	Polígonos	2.140	4,7
Municípios	*M*	Polígonos	5.564	38,8
Alertas desmat. cerrado	*A*	Polígonos	32.578	11,2
Rodovias	*R*	Linhas	51.646	15,2
Hidrografia	*H*	Linhas	226.963	64,5
Culturas	*CU*	Polígonos	123.746	69,3
Ferrovias	*FM*	Linhas	194.261	28,7
Represas de água	*RA*	Polígonos	338.860	136,7
Contorno de relevo	*CR*	Linhas	703.574	572,5
Hidrografia Mundial	*HM*	Linhas	943.638	243,2

Tabela 2. Consultas espaciais utilizadas nos experimentos.

Sigla	Consulta	Característica Principal	Cardin. Junção
Q1	$FM \bowtie HM \bowtie CU$	Junção com polígonos e linhas	2313
Q2	$A \bowtie HM \bowtie FM \bowtie CU$	Conjunto resultante seletivo	168
Q3	$HM \bowtie FM \bowtie CR$	Maiores datasets de linhas	2659
Q4	$HM \bowtie CR \bowtie RA$	Os três maiores datasets	771
Q5	$V \bowtie A \bowtie M$	Datasets pequenos, colocalizados	36.469
Q6	$A \bowtie P \bowtie M \bowtie V$	Seletividade nula	0
Q7	$HM \bowtie CR \bowtie FM \bowtie R$	Todos datasets de linhas	3.139
Q8	$RA \bowtie CU \bowtie A \bowtie M$	Todos datasets de polígonos	26.128

4.1. Avaliação do Método de Construção de Histogramas

Para avaliar o método de construção das dimensões de cardinalidade e pontos do histograma multidimensional, definido na Seção 3.1, executamos todos os planos das consultas da Tabela 2 e comparamos a comunicação real mensurada no *cluster* com o valor estimado de comunicação, obtido através da execução do Algoritmo 1 utilizando histogramas gerados pelas duas técnicas (centroide e sobreposição proporcional).

O resultado da comparação pode ser visualizado no gráfico da Figura 4. O valor do eixo *y* foi obtido dividindo o valor estimado pelo valor real. O erro da técnica baseada no centroide dos objetos é de $\approx 45.7\%$, na média entre todos os planos, mas chega a 79.09% no pior caso ($Q8\ P_6$). O novo método proposto tem um erro médio de $\approx 10.97\%$ e de no máximo 44.15% ($Q5\ P_2$), mostrando ser bem mais efetivo para estimar a comunicação das consultas. Esta estimativa interfere na escolha do melhor plano de execução e, portanto, os resultados a seguir consideram o método de sobreposição proporcional.

4.2. Avaliação da Estimativa de Comunicação

Para avaliar o algoritmo de estimativa de comunicação, definido na Seção 3.2, que é base do otimizador de consultas de junção espacial complexas proposto, comparamos a estimativa do Algoritmo 1 com o resultado real capturado na execução. O resultado da comparação pode ser visualizado no gráfico da Figura 5. O gráfico possui duas escalas de eixo *y*. A escala da direita é usada para os três últimos planos ($Q3\ P_1$, $Q7\ P_2$ e $Q7\ P_3$). No gráfico,

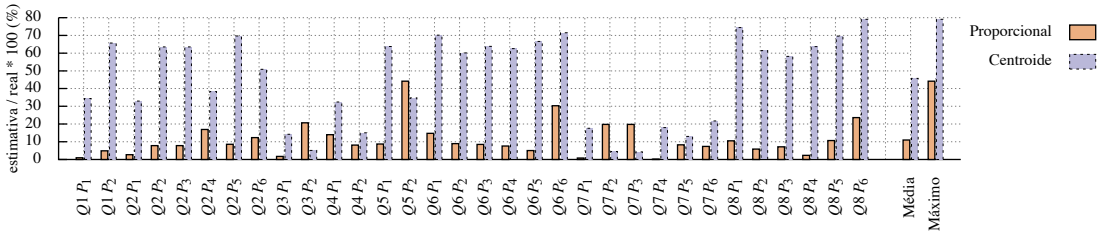


Figura 4. Erro de estimativa dos métodos de construção de histograma.

as barras representam o volume de comunicação médio estimado e real dos servidores, em cada consulta e plano de execução. O volume de comunicação total de cada plano pode ser obtido multiplicando o valor do eixo *y* por oito (quantidade de servidores). Em cada barra há também um limite inferior e superior, que representam, respectivamente, a comunicação do servidor com menor e maior comunicação. Uma variação muito grande nestes limites indica que o processamento do plano será desbalanceado, porque o servidor com a comunicação no limite superior se tornaria um *hotspot*.

Observando ainda o gráfico da Figura 5, nota-se que a estimativa média de comunicação para os servidores varia em relação à comunicação real, apesar de visualmente estar bem próxima. Conforme discutido na seção anterior (4.1), este erro é de $\approx 10.7\%$, em média. Observando as barras de erros, estimadas e reais, que representam o balanceamento da consulta, observa-se que, de forma geral, a barra de erro estimada está próxima da real, em todos os planos de execução. Dois casos particulares merecem atenção: $Q1\ P_2$, onde o servidor com maior comunicação estimada está muito superior ao resultado real, e $Q8\ P_6$ onde o inverso ocorre.

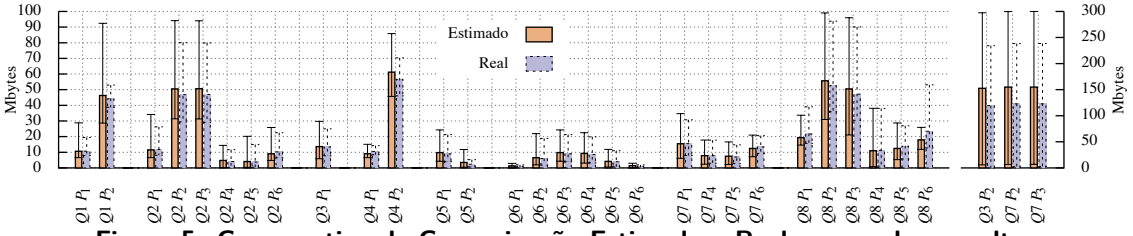


Figura 5. Comparativo de Comunicação Estimada e Real para cada consulta.

Estes números refletem a simplificação do dataset realizada pelo histograma, que não permite uma estimativa exata para todos os casos. No entanto, também demonstram a proximidade do resultado real, obtido pelo algoritmo e histogramas propostos. Os dois erros de estimativa citados podem ser investigados em trabalhos futuros, com o objetivo de aprimorar o método de estimativa. Conclui-se, portanto, que o Algoritmo 1 estima com sucesso a comunicação média dos servidores, de acordo com a distribuição dos dados, com uma margem de erro pequena. A próxima seção avalia o uso desta estimativa na escolha do melhor plano de execução das consultas.

4.3. Avaliação da Escolha de Planos de Execução

Nesta subseção, avaliamos se a Equação 1 (Subseção 3.3), associada à estimativa de comunicação de cada servidor fornecida pelo Algoritmo 1, escolhe um bom plano de execução para uma consulta espacial complexa.

Uma ilustração gráfica da execução do otimizador, detalhando a escolha do melhor plano estimado, é apresentada na Figura 6. As barras no gráfico indicam a quantidade (em MBytes) do servidor com maior comunicação do plano. A barra de menor altura para

cada consulta é o plano escolhido pelo otimizador, conforme indicamos usando setas no gráfico.

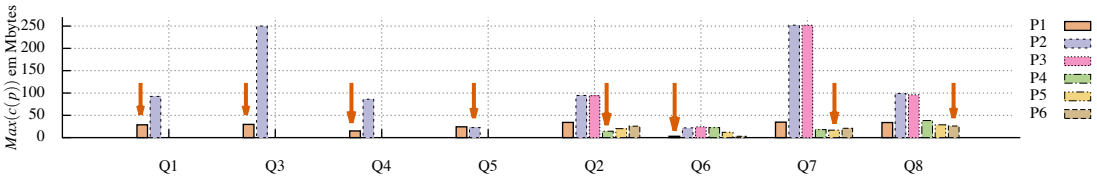


Figura 6. Estimativa de comunicação e escolha estimada do melhor plano.

Para validar se a escolha recai sobre o melhor plano, comparamos o tempo de execução dos 32 planos possíveis, com o melhor plano de execução estimado, escolhido pela Equação 1. O tempo de execução real de cada plano de execução é apresentado no gráfico da Figura 7. Usamos uma escala logarítmica para melhor visualização dos melhores planos, que demoram apenas alguns segundos, em contraste com os piores planos, cujos tempos de execução são da ordem de milhares de segundos. Pode-se observar que o melhor plano estimado foi também o melhor plano executado em 7 das 8 consultas: $Q1$, $Q3$, $Q4$, $Q5$, $Q6$, $Q7$ e $Q8$. Na consulta $Q2$, a única para a qual o plano estimado foi diferente do real, o otimizador escolheu um plano de execução que demora 14,4 seg, contra o melhor plano, cuja execução demora apenas 1,8 seg. Ainda assim, a estimativa desviou-se dos planos menos eficientes, que gastam em torno de 21 min cada (P_2 e P_3).

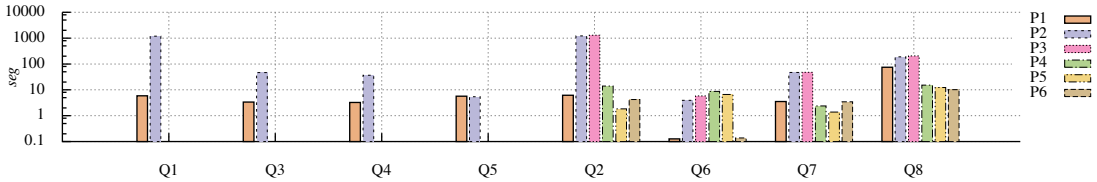


Figura 7. Tempo real de execução dos planos no *cluster*.

5. Trabalhos Relacionados

Propomos neste artigo um otimizador de consultas de junção espacial complexa e distribuída. Apesar da importância deste tipo de consulta e do aumento do tamanho dos datasets, que implicam na necessidade de formas mais eficientes de processamento e processamento distribuído de consultas, poucos são os trabalhos desta natureza encontrados.

Algumas propostas de otimizadores de consultas de junção espacial complexa, não distribuída, foram publicadas na literatura. O trabalho mais relevante encontrado é [Mamoulis and Papadias 2001b], o qual propõe um otimizador baseado em custos de E/S de disco e CPU. Tais custos são calculados a partir da cardinalidade dos datasets. [Fornari et al. 2006, Fornari et al. 2007] também propõem um otimizador de consultas, porém baseado em regras fixas, e para consultas de junção espacial simples.

Os trabalhos de [Aji et al. 2012, Zhang et al. 2009, Zhong et al. 2012, Gupta et al. 2013] empregam computadores distribuídos e *frameworks* MapReduce no processamento de consultas de junção espacial. O foco dos trabalhos é no particionamento dos dados para aderir ao *framework*. Algumas otimizações baseadas em regras fixas são propostas, mas os testes são realizados com o processamento de junções com dois datasets, não havendo definição de uma estratégia para consultas complexas.

Alguns artigos apresentam algoritmos para o processamento distribuído de junções espaciais simples, e que podem ser combinados em etapas sucessivas para o proces-

samento de junções complexas, como [Patel and DeWitt 2000, Chung et al. 2005, de Oliveira et al. 2013]. Outros trabalhos propõem soluções distribuídas para o processamento de junções espaciais complexas, combinando os datasets em uma só etapa, como [Gupta et al. 2013, Cunha et al. 2015].

Destas duas estratégias, o processamento em etapas necessita de um otimizador mais complexo, que define a quantidade de etapas e os algoritmos de junção simples para cada uma. Nosso trabalho implementou e avaliou um otimizador desta natureza, combinando os datasets dois a dois e empregando uma versão melhorada do algoritmo CJ. Os algoritmos que realizam a junção de todos os datasets em uma só etapa podem ser incluídos no otimizador. [Mamoulis and Papadias 2001b] realizou esta avaliação em sistemas centralizados e concluiu que o melhor plano, às vezes, inclui o processamento de mais de dois datasets em uma etapa. Devido a complexidade desta experimentação, deixamos esta implementação e avaliação para trabalhos futuros.

6. Conclusão

Neste artigo, identificamos as características dos datasets e da distribuição dos dados, relevantes para o processamento de junções espaciais complexas distribuídas, definimos um histograma espacial para organização dessas características, o qual também foi usado como método de acesso distribuído. Definimos um algoritmo que estima o custo de planos de execução através do histograma multidimensional e uma nova técnica de construção de histogramas espaciais baseada na sobreposição proporcional, que aproximou-se da estimativa de custo dos planos de execução (erro médio de $\approx 10.97\%$). O otimizador proposto acertou o melhor plano de execução em 7 das 8 consultas com datasets reais experimentadas, num total de 32 planos de execução possíveis.

Implementamos também uma versão melhorada do algoritmo Clone Join (CJ), que inclui um método de pré-eliminação de resultados duplicados. Com esta melhoria, este algoritmo se torna não bloqueante e forma um *pipeline* de processamento de etapas em consultas espaciais complexas, melhorando o tempo de execução das mesmas.

Em trabalhos futuros pretendemos incluir no histograma outros aspectos de um sistema distribuído, como a carga pré-existente, gerada durante o processamento de fluxo contínuo de consultas espaciais. Este é um cenário mais realista para um banco de dados espacial. Também pretendemos avaliar técnicas de histogramas mais avançadas, como o *Min-Skew*, para verificar se a melhora no tempo de processamento da consulta compensa o tempo de construção e manutenção dos histogramas.

Referências

Acharya, S., Poosala, V., and Ramaswamy, S. (1999). Selectivity estimation in spatial databases. In *SIGMOD*, volume 28, pages 13–24. ACM.

Aji, A., Wang, F., and Saltz, J. H. (2012). Towards building a high performance spatial query system for large scale medical imaging data. In *SIGSPATIAL*, page 309. ACM.

Bernhardsen, T. (2002). *Geographic information systems: an introduction*. Wiley.

Brinkhoff, T., Kriegel, H.-P., and Seeger, B. (1996). Parallel processing of spatial joins using R-trees. In *ICDE*, pages 258–265. IEEE.

Chung, W., Park, S., and Bae, H. (2005). Efficient parallel spatial join processing method in a shared-nothing database cluster system. *Embedded Software and Systems*, pages 81–87.

Cunha, A. R., de Oliveira, S. S. T., Aleixo, E. L., de C. Cardoso, M., de Oliveira, T. B., and do Sacramento Rodrigues, V. J. (2015). Processamento Distribuído da Junção Espacial de Múltiplas Bases de Dados - Multi-way Spatial Join. In *Proc. of XXXIII SBRC*, pages 165–178.

de Oliveira, S. S. T., do Sacramento Rodrigues, V. J., Cunha, A. R., Aleixo, E. L., de Oliveira, T. B., de C. Cardoso, M., and Junior, R. R. (2013). Processamento Distribuído de Operações de Junção Espacial com Bases de Dados Dinâmicas para Análise de Informações Geográficas. In *Proc. of XXXI SBRC*, pages 1009–1022.

Dittrich, J.-P. and Seeger, B. (2000). Data redundancy and duplicate detection in spatial join processing. *Proceedings of 16th ICDE*, pages 535–546.

Fornari, M., Comba, J., and Iochpe, C. (2007). A Rule-Based Optimizer for Spatial Join Algorithms. *Advances in Geoinformatics*, pages 73–90.

Fornari, M. R., Comba, J. L. D., and Iochpe, C. (2006). Query optimizer for spatial join operations. In *Proceedings of GIS '06*, pages 219–226, New York, USA. ACM.

Gupta, H., Chawda, B., Negi, S., Faruquie, T. A., Subramaniam, L. V., and Mohania, M. (2013). Processing multi-way spatial joins on map-reduce. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 113–124. ACM.

Mamoulis, N. and Papadias, D. (2001a). *Advances in Spatial and Temporal Databases*, volume 2121 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg.

Mamoulis, N. and Papadias, D. (2001b). Multiway spatial joins. *TODS*, 26(4):424–475.

Mutenda, L. and Kitsuregawa, M. (1999). Parallel R-tree spatial join for a shared-nothing architecture. In *DANTE*, pages 423–430. IEEE.

Naughton, J. and Ellmann, C. (2002). A non-blocking parallel spatial join algorithm. In *Proceedings 18th ICDE*, pages 697–705. IEEE, IEEE.

Papadias, D., Mamoulis, N., and Theodoridis, Y. (1999). Processing and optimization of multiway spatial joins using R-trees. *PODS*, pages 44–55.

Patel, J. M. and DeWitt, D. J. (2000). Clone join and shadow join. In *Proceedings of GIS '00*, pages 54–61. ACM, ACM.

Sivasubramaniam, A. (2001). Selectivity estimation for spatial joins. In *Proceedings 17th ICDE*, pages 368–375. IEEE, IEEE.

Zhang, S., Han, J., Liu, Z., Wang, K., and Xu, Z. (2009). Sjmr: Parallelizing spatial join with mapreduce on clusters. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–8. IEEE.

Zhong, Y., Han, J., Zhang, T., Li, Z., Fang, J., and Chen, G. (2012). Towards Parallel Spatial Query Processing for Big Spatial Data. *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pages 2085–2094.

Zhou, X., Abel, D. J., and Truffet, D. (1998). Data partitioning for parallel spatial join processing. *GeoInformatica*, 2(2):175–204.

sbbd:08

Employee Analytics through Sentiment Analysis

André Costa[1] **, Adriano Veloso**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

{andrecosta, adrianov}@dcc.ufmg.br

Abstract. *People discuss and talk about the most diverse topics in social media platforms, including about their jobs. This results in a stream of employee-related data, and organizations are increasingly interested in making sense of this data on an ongoing basis in order to assess key factors such as employee engagement, retention and satisfaction. In this paper we propose to estimate such factors from different sentiments that are implicit in employee communications in social media platforms. We introduce sentiment analysis approaches that are based on learning vector representations for employee communications in an unsupervised way, and then these representations are given as input to a state-of-the-art supervised regression algorithm which finally maps text/sentiment to employee factors. We collected a large set of employee communications in social platforms, survey data such as work/life balance, job culture and management, and also official data about retention and salary. Then, we performed a systematic set of experiments using the collected data, and our results show that learning representations leads to better sentiment analysis performance than engineering features based on the standard term-frequency and inverse-document-frequency numbers (i.e., TF-IDF weighting scheme).*

1. Introduction

Studies have found that 70% of American workers are not engaged or are actively disengaged in their jobs.[1] There is a direct correlation between employee engagement[2] and the quality of the customer experience, and thus disengaged employees may cost to businesses up to US$550 billion per year. In this paper we propose approaches that capture and analyze employee feedback, and then use them to understand factors that impact employee retention, satisfaction, and productivity. Specifically, our proposed approaches analyse positive and negative evidence and reviews appearing in social media platforms, such as `www.indeed.com` and `www.lovemondays.com.br`, to figure out what employees think about their jobs in order to build predictive models for diverse criteria of interest.

Our proposed approaches to sentiment analysis require: (i) representing text as a fixed-length vector and (ii) using machine learning techniques in order to score sentiments expressed in the text. Textual data (or more specifically job reviews) are represented as features vectors. For instance, a review may be represented as a vector of TF-IDF weights [Baeza-Yates and Ribeiro-Neto 2011]. Then, a state-of-the-art regression or classification algorithm, such as SVR [Drucker et al. 1996] and

[1]`http://www.gallup.com/services/176708/state-american-workplace.aspx`

[2]Employee engagement is the extent to which an employee is enthusiastic about his or her work and committed to furthering the organisation's goals.

SVM [Joachims 1998, Joachims 2006, Cortes and Vapnik 1995] are used in order to separate sentiments.

Our focus in this paper is in improving text representations for employee analytics (or people analytics) through sentiment analysis. While the TF-IDF scheme has been successfully adopted in many application scenarios, including sentiment analysis [Martineau and Finin 2009, Paltoglou and Thelwall 2010], its weighting scheme assumes an exact match of words and thus ignores the similarity between synonyms and cannot distinguish polysemy. The accuracy of sentiment analysis largely depends on measuring the semantic similarity of document pairs that are associated with the same sentiments, and thus we assume that an improved semantic representation for documents would lead to more accurate sentiment analysis. Thus, in addition to approaches based on the standard TF-IDF representation, we also evaluate approaches that learn a vector representation for words, such that related words appear next to each other in a common space [Mikolov et al. 2013b]. The learnt word vectors are then combined to represent a document [Le and Mikolov 2014].

Contributions and Findings. In practice, we claim the following benefits and contributions over existing solutions:

- Effective sentiment analysis approaches for employee analytics. Our approaches are based on learning vector representations for reviews posted by employees about their jobs. The representations are chosen in a way such that reviews that share words with similar meanings are placed next to each other in a common metric space.
- We collected a large number of job reviews posted in social platforms, as well as survey data such as work/life balance, management, culture, and also official data about retention and salary. We performed a systematic set of experiments in order to evaluated our proposed sentiment analysis approaches.

In the next section we discuss related work. In Section 3 we present our sentiment analysis approaches for employee analytics. In Section 4 we present our experiments and report results obtained by the proposed approaches. Finally, in Section 5 we conclude our paper.

2. Background and Related Work

Social media is full of opinative content [Liu 2012]. Analysing such opinative content is a task which is known as sentiment analysis. Approaches for sentiment analysis usually apply machine learning algorithms in order to derive predictive models based on the observed sentiments [Mukherjee and Liu 2012]. Such machine learning algorithms depends heavily on how the training data is represented. Paltoglou and Thelwall [Paltoglou and Thelwall 2010] systematically explore whether more sophisticated feature weighting schemes from Information Retrieval can enhance sentiment analysis accuracy. They showed that variants of the classic TF-IDF scheme provide significant increases in accuracy, especially when using a sublinear function for term frequency weights and document frequency smoothing.

Delta IDF weighting [Martineau and Finin 2009] is a supervised variant of TF-IDF weighting in which the IDF computation is done for each document class and then

one value is subtracted from the other. The authors present evidence that this weighting helps with sentiment classification.

It is also common to improve the TF-IDF scheme with latent information, using directed graphical models, notably probabilistic Latent Semantic Indexing [Hofmann 1999], as a variant of Latent Semantic Indexing [Deerwester et al. 1990] and Latent Dirichlet Allocation [Blei et al. 2003]. These approaches model each document as mixing proportions for latent components which are viewed as "topics", and a topic is represented as a probabilistic distribution over words to capture their correlations. In [Maas et al. 2011] the authors present a model to capture semantic similarities among words. The semantic component of their model learns word vectors via an unsupervised probabilistic model of documents.

Most of the existing works have focused on analysing the content of movie or general product reviews [Mesnil et al. 2015, Pang and Lee 2007]. There are also applications to other domains such as debates [Thomas et al. 2006], news [Devitt and Ahmad 2007], and blogs [Ounis et al. 2008]. To the best of our knowledge, this is the first study of sentiment analysis applied to employee analytics.

3. Extracting Employee Sentiment

The popular TF-IDF [Baeza-Yates and Ribeiro-Neto 2011] scheme represents each employee review using frequency count of words in a basic vocabulary, and normalize the values using inverse review frequency count. Then, TF-IDF vectors are given as input to machine learning algorithms in order to produce predictive models. This is known as the BoW (Bag of Words) approach.

3.1. Learning Review Representations

Despite their popularity, bag-of-words features have two major weaknesses: they lose the ordering of the words in the review and they also ignore semantics of the words. An alternative approach is to learn review representations by capturing semantic and syntactic information of words. Specifically, we represent words by building word projections in a latent space of n dimensions, such that words with similar meanings are placed next to each other in a common $n-$dimensional space metric. There are efficient algorithms to learn word vector representations, and a particularly efficient one follows the skip-gram approach. In this case, a neural network [Mikolov et al. 2013a, Mikolov et al. 2013b] receives as input a word w_i, and the output of the network could be w_{i-1}, w_{i-2}, w_{i+1}, and w_{i+2}. Informally, the task is to predict the context given a specific word w_i. During the process of predicting the context given a word w_i, the network essentially learns a set of weigths, which are then used to represent word w_i. Finally, a review can be represented as vector given as the centroid of the vector representations associated with the words in the review. This approach will referred as Avg-Vec. As with the TF-IDF representation, the Avg-Vec representation can also serve as input to a machine learning algorithm in order to build a predictive model.

A further alternative approach, called Par-Vec, is to learn paragraph vectors [Le and Mikolov 2014]. A paragraph vector can also be used to represent a review. The process of learning paragraph vectors is inspired by the process of learning word vector representations. Specifically, paragraph vectors are learnt by concatenating word

vectors, and again, using a neural network to predict the next word in the paragraph (or in the review).

3.2. Prediction Models

Once reviews are represented as vectors, either using BoW, Avg-Vec or Par-Vec approaches, they are given as input to a regression or classification algorithm. Specifically, we used the SVR algorithm [Drucker et al. 1996] for regression, and the SVM algorithm [Joachims 1998] for classification. These algorithms follows a supervised learning strategy, and associate patterns in the vector representation of the review and a variable or criterion of interest. Criteria can assume values as salary, retention, management, culture, work/life balance, and others.

4. Results

In this section we present the experimental results for the evaluation of the proposed approaches in terms of predicting employee related variables from textual reviews. In summary, we show that Avg-Vec and Par-Vec approaches achieve similar performance. However, they are significantly superior than the standard BoW approach. All experiments were run on an Intel Core i7 64GB main memory equiped with a Tesla K-40 GPU accelerator with 12GB memory and ultra-fast 288 GB/s throughput.[3] Our algorithms were implemented using Word2Vec[4] for computing word vector representations using skip-gram, and ParagraphVec[5] for computing vector representations of text with arbitrary length. These implementations are compatible with the CUDA interface. We implemented the BoW approach.

4.1. Dataset

We collected employee reviews from two sources, namely: `www.indeed.com.br` (refereed to as Indeed) and `www.lovemondays.com.br` (referred to as LM). We used a total of 11 machines with different IP addresses to efficiently gather large amounts of data. In total we crawled 161,708 reviews (151,349 labeled and 10,359 unlabeled) from Indeed and 20,013 reviews (13,007 labeled and 7,006 unlabeled) from LM. Labeled reviews from Indeed come with ratings (ranging from 0 to 5) based on management, culture, work/life balance, benefits, and career opportunities. An overall rating is also available. Labeled reviews from LM come with the salary of the employee, which are divided into 11 ranges according to the number of minimum wages (ranging from less than a minimum wage to more than 10 minimum wages). Further, we also obtained official data concerning employee retention.

Retention is divided into two groups: companies with low retention (60,158 reviews), and companies with high retention (85,737 reviews). Figure 1 shows the disctribution of the overall rating (Left), and the distribution of salaries (Right). Figure 2 shows the frequency distribution of words, segmented by overall rating on the Indeed dataset (Left), and segmented by salary on the LM dataset (Right).

[3]We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPU accelerator for this research.

[4]https://code.google.com/p/word2vec

[5]https://github.com/darshanhegde/ParagraphVec

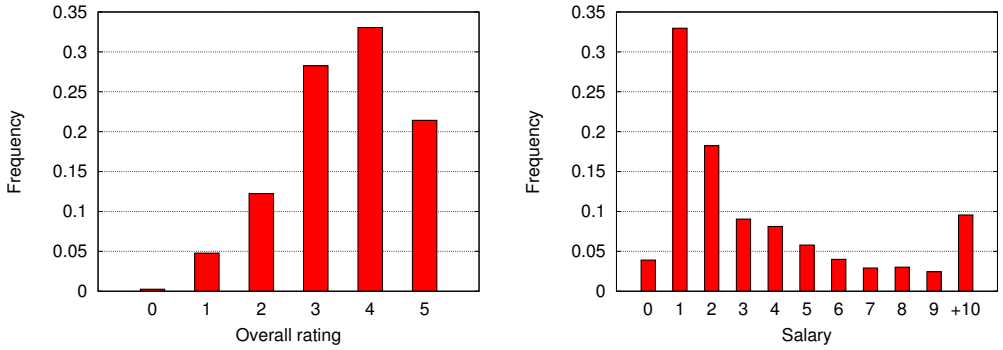


Figure 1. Distribution of ratings and salaries. Left – Indeed. Right – Love Mondays.

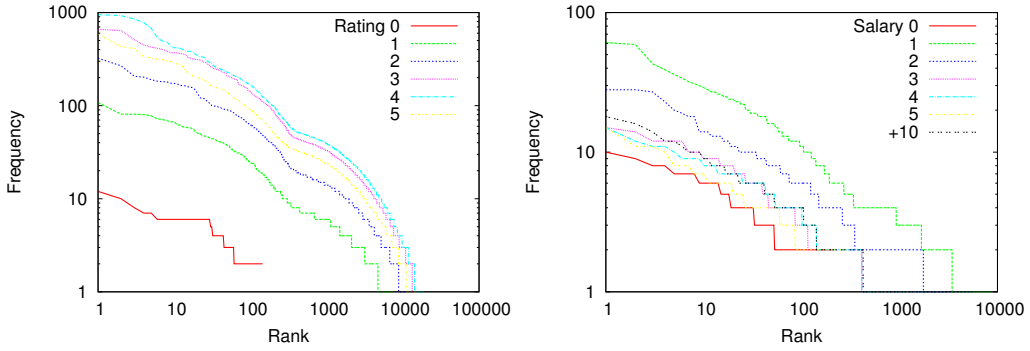


Figure 2. Frequency distribution of words. Left – Indeed. Right – Love Mondays.

Figure 3. Left – one minimum wage. Middle – 5-6 minimum wages. Right – +10 minimum wages.

Figure 3 show wordles that were built reviews of employees associated with different salaries. We can grasp clear differences between the vocabulary used by these employees. The vocabulary also emcompasses names of the companies and the employee position in the company.

4.2. Evaluation Procedure

To evaluate the prediction performance of our approaches, we have used the standard Root Mean Squared Error (RMSE) measure, which gives a summarized measure of the prediction error for regression tasks, and the standard accuracy and F1 measures for clas-

sification tasks. We conducted ten-fold cross validation using Indeed and LM datasets. Thus, the labeled dataset was arranged into ten folds, including training and test. At each run, nine folds are used as training set, and the remaining fold as test set. The results reported are the average of the ten runs. SVR and SVM parameters used are those that lead to the best results.

4.3. Prediction Performance

Our first experiment is concerned with predicting different objectives, from Indeed dataset, including: management, culture, work/life balance, benefits, and career opportunities. Both Avg-Vec and Par-Vec achieved similar performance numbers, and the BoW approach was the worst performer in the most prediction objectives and for the most word vector representations, as shown in Figure 4.

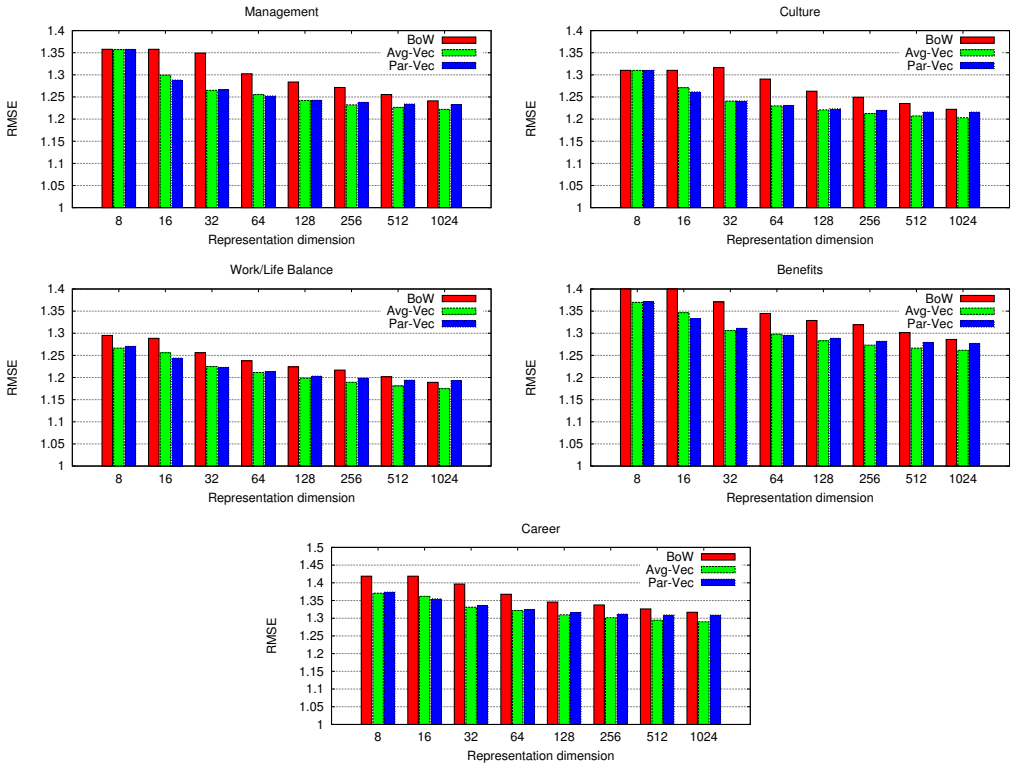


Figure 4. RMSE numbers for different objectives.

The next experiment is concerned with predicting the salary of the employee, from the LM dataset, based on the review he has written. As shown in Figure 5, Avg-Vec provides the best RMSE numbers, followed by Par-Vec, until the representation with 512 features. However, from 1024 features, the performance of Avg-Vec and Par-Vec begins to deteriorate. This happens due to the amount of data available from LM (20,013 reviews). In other words, LM dataset does not have data enough for Avg-Vec and Par-Vec converge in higher dimensions.

According to Mikolov et al., several factors can influence the quality of the word vectors, like amount and quality of the datasets and size of the word vector representations [Mikolov et al. 2013a, Mikolov et al. 2013b, Mikolov et al. 2013c]. Therefore, in order to try to improve the performance of Avg-Vec and Par-Vec against BoW, we joined data

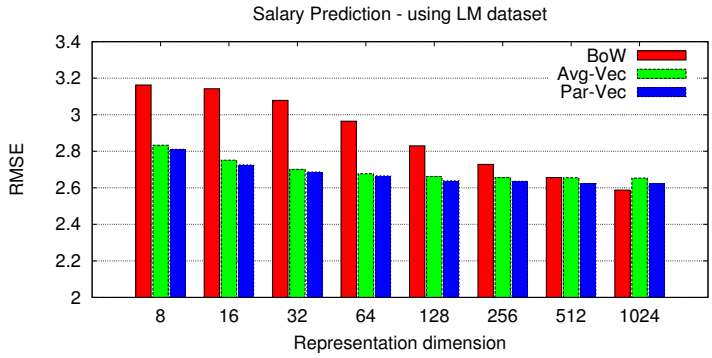


Figure 5. RMSE numbers according to the dimension of the representation (LM dataset)

from Indeed and LM datasets, totaling 181,721 reviews. Figure 6 shows RMSE numbers, confirming that more data improve the results.

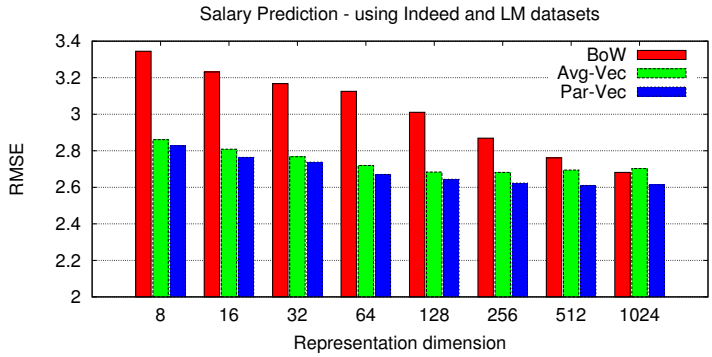


Figure 6. RMSE numbers according to the dimension of the representation (Indeed and LM datasets).

Another experiment performed is also concerned with predicting the salary of the employee based on the review he has written. However, for this experiment we varied the amount of unlabeled data available to learn word vector representations. Figure 7 (Left) shows RMSE numbers by varying the amount of unlabeled data with context (i.e., reviews without the rating). Figure 7 (Right), on the other hand, shows RMSE numbers by varying the amount of unlabeled data without context. In this case, text was randomly collected from Wikipedia. Clearly, RMSE numbers decrease as more unlabeled data with context is used to learn word vector representations. Unlabeled data without context, however, may be detrimental to learn word representations.

The last experiment is concerned with predicting the employee retention based on the review. In this case, we used the standard accuracy and F1 measures, since the retention can assume only two discrete values: low and high. Again, the best performers were Avg-Vec and Par-Vec, as shown in Figure 8. Specifically, Avg-Vec performed slightly better in terms of F1, while Par-Vec performed better in terms of accuracy. The BoW approach showed the worst results in terms of accuracy, but achieved competitive F1 numbers.

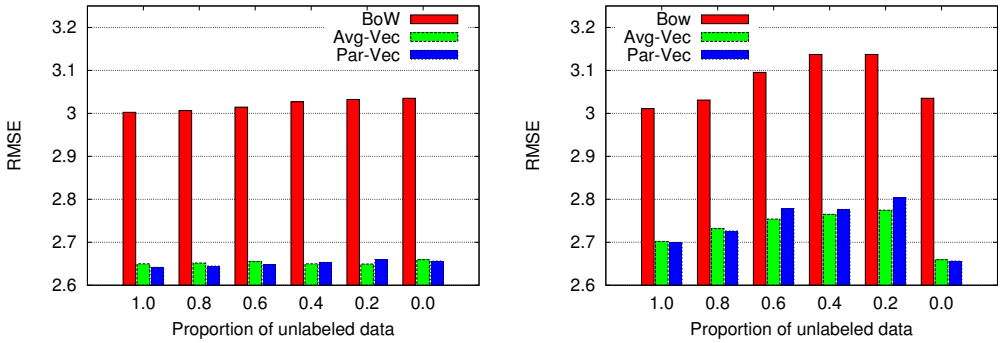


Figure 7. Salary prediction: RMSE numbers according to the proportion of unalabeled data available. Left – Unlabeled data with context. Right – Unlabeled data without context.

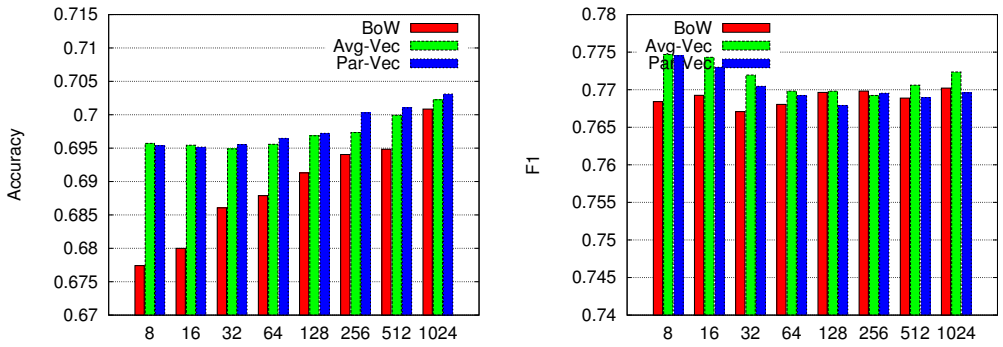


Figure 8. Retention prediction: accuracy and F1 numbers.

5. Significance Tests

We also applied Friedman (non-parametric statistical) and Nemenyi (post-hoc) tests to detect differences between the approaches BoW, Avg-Vec and Par-Vec across results achieved (RMSE, accuracy and F1 values) by experiments performed from all dimensions of vector representations considered [Pohlert 2014].

Tables 1, 2, and 3 shows the tests results for different prediction objectives and dimensions. As most Friedman tests indicates significance (p-value < 0.01), it is valid to make multiple comparisons to identify differences between the approaches through the Nemenyi test. Tests with p-value ≥ 0.01 – Retention (F1) [medium-sized vector] and Salary (LM only) [high-dimensional vector] – indicate that the approaches BoW, Avg-Vec and Par-Vec are statistically similar, not being necessary to apply the Nemenyi test in these cases.

In the following we apply Nemenyi post-hoc tests. As shown in Table 4, we have the post-hoc tests for *low-dimensional* vector representations. The BoW approach differs significantly (p-value < 0.01) to Avg-Vec and Par-Vec approaches in all prediction objectives. Avg-Vec and Par-Vec approaches do not differ (p-value ≥ 0.01) in most cases, i.e., they are statistically similar in these cases. Thus, considering low-dimensional vectors, we can conclude (based on subsection 4.3) that:

- BoW is statistically worse than Avg-Vec and Par-Vec;
- Avg-Vec and Par-Vec approaches diverge (p-value < 0.01) in two prediction cri-

Table 1. Friedman Tests – Low-dimensional vector representations (2^3 to 2^5 features)

Prediction Objectives	p-value
Management	1.87×10^{-9}
Culture	5.93×10^{-6}
Work/Life	1.25×10^{-10}
Benefits	1.48×10^{-10}
Career	1.48×10^{-10}
Salary (LM only)	6.03×10^{-12}
Salary (Indeed + LM)	2.46×10^{-13}
Retention (Accuracy)	5.06×10^{-11}
Retention (F1)	1.39×10^{-12}

Table 2. Friedman Tests – Medium-sized vector representations (2^6 to 2^8 features)

Prediction Objectives	p-value
Management	1.64×10^{-10}
Culture	3.30×10^{-11}
Work/Life	2.46×10^{-13}
Benefits	9.93×10^{-11}
Career	2.46×10^{-13}
Salary (LM only)	1.14×10^{-11}
Salary (Indeed + LM)	2.46×10^{-13}
Retention (Accuracy)	6.65×10^{-11}
Retention (F1)	$\mathbf{8.21 \times 10^{-2}}$

Table 3. Friedman Tests – High-dimensional vector representations (2^9 to 2^{10} features)

Prediction Objectives	p-value
Management	2.06×10^{-9}
Culture	5.33×10^{-9}
Work/Life	2.50×10^{-7}
Benefits	2.06×10^{-9}
Career	2.06×10^{-9}
Salary (LM only)	$\mathbf{1.06 \times 10^{-2}}$
Salary (Indeed + LM)	1.95×10^{-7}
Retention (Accuracy)	2.89×10^{-4}
Retention (F1)	1.76×10^{-5}

Table 4. Nemenyi post-hoc tests – Low-dimensional vector representations (2^3 to 2^5)

	BoW/Avg-Vec	BoW/Par-Vec	Avg-Vec/Par-Vec
Management	2.8×10^{-8}	2.8×10^{-6}	$\mathbf{6.8 \times 10^{-1}}$
Culture	5.3×10^{-4}	3.4×10^{-5}	$\mathbf{7.9 \times 10^{-1}}$
Work/Life	1.8×10^{-7}	1.7×10^{-9}	$\mathbf{7.2 \times 10^{-1}}$
Benefits	3.9×10^{-9}	8.5×10^{-8}	$\mathbf{8.6 \times 10^{-1}}$
Career	3.9×10^{-9}	8.5×10^{-8}	$\mathbf{8.6 \times 10^{-1}}$
Salary (LM only)	1.9×10^{-5}	3.8×10^{-12}	$\mathbf{2.7 \times 10^{-2}}$
Salary (Indeed + LM)	1.9×10^{-4}	9.7×10^{-14}	8.8×10^{-4}
Retention (Accuracy)	2.1×10^{-10}	1.0×10^{-6}	$\mathbf{3.3 \times 10^{-1}}$
Retention (F1)	5.8×10^{-13}	6.1×10^{-5}	5.5×10^{-3}

terias – Salary (Indeed+LM) and Retention (F1). In both criterias, Par-Vec is statistically better than Avg-Vec.

In Table 5, we have the post-hoc tests for *medium-sized* vector representations. The BoW approach differs significantly (p-value < 0.01) to Avg-Vec and Par-Vec ap-

Table 5. Nemenyi post-hoc tests – Medium-sized vector representations (2^6 to 2^8 features)

	BoW/Avg-Vec	BoW/Par-Vec	Avg-Vec/Par-Vec
Management	8.6×10^{-9}	4.0×10^{-8}	$\mathbf{9.6 \times 10^{-1}}$
Culture	5.7×10^{-11}	2.8×10^{-6}	$\mathbf{1.7 \times 10^{-1}}$
Work/Life	9.7×10^{-14}	1.9×10^{-4}	8.8×10^{-4}
Benefits	7.6×10^{-10}	3.6×10^{-7}	$\mathbf{5.6 \times 10^{-1}}$
Career	9.7×10^{-14}	1.9×10^{-4}	8.8×10^{-4}
Salary (LM only)	1.0×10^{-5}	9.4×10^{-12}	$\mathbf{5.3 \times 10^{-2}}$
Salary (Indeed + LM)	1.9×10^{-4}	9.7×10^{-14}	8.8×10^{-4}
Retention (Accuracy)	1.9×10^{-5}	5.7×10^{-11}	$\mathbf{7.2 \times 10^{-2}}$
Retention (F1)	–	–	–

proaches in all prediction objectives. Avg-Vec and Par-Vec approaches do not differ (p-value ≥ 0.01) in six criterias, i.e., they are statistically similar in these cases. Thus, considering *medium-sized* vectors, we can conclude (based on subsection 4.3) that:

- BoW is statistically worse than Avg-Vec and Par-Vec;
- Avg-Vec and Par-Vec approaches differ (p-value < 0.01) in three criterias: Work/Life, Career and Salary (Indeed+LM). So, Avg-Vec is significantly superior that Par-Vec in Career and Work/Life criterias; Par-Vec is significantly superior than Avg-Vec considering Salary (Indeed+LM) criteria.

Table 6. Nemenyi post-hoc tests – High-dimensional vector representations (2^9 to 2^{10} features)

	BoW/Avg-Vec	BoW/Par-Vec	Avg-Vec/Par-Vec
Management	7.6×10^{-10}	4.5×10^{-3}	4.5×10^{-3}
Culture	2.1×10^{-9}	$\mathbf{1.2 \times 10^{-2}}$	2.6×10^{-3}
Work/Life	1.3×10^{-6}	$\mathbf{8.0 \times 10^{-1}}$	2.8×10^{-5}
Benefits	7.6×10^{-10}	4.5×10^{-3}	4.5×10^{-3}
Career	7.6×10^{-10}	4.5×10^{-3}	4.5×10^{-3}
Salary (LM only)	–	–	–
Salary (Indeed + LM)	$\mathbf{6.1 \times 10^{-1}}$	5.4×10^{-7}	5.8×10^{-5}
Retention (Accuracy)	$\mathbf{2.0 \times 10^{-2}}$	2.3×10^{-4}	$\mathbf{4.1 \times 10^{-1}}$
Retention (F1)	4.3×10^{-4}	$\mathbf{8.8 \times 10^{-1}}$	5.8×10^{-5}

In Table 6, we have the post-hoc tests for *high-dimension* vector representations. Thus, considering *medium-sized* vectors, we can conclude (based on subsection 4.3) that:

- BoW approach differs (p-value < 0.01) in most criterias, to Avg-Vec and Par-Vec. So, BoW is statistically worse than Avg-Vec and Par-Vec in these cases;
- in Salary (Indeed+LM) and Retention (Accuracy) criterias, BoW and Avg-Vec are statistically similar;
- in Culture, Work/Life and Retention (F1), BoW and Par-Vec are statistically similar.

- Avg-Vec differs significantly (p-value < 0.01) to Par-Vec in most cases, except Retention (Accuracy). So, Par-Vec is significantly superior that Avg-Vec only in Salary (Indeed+LM); Par-Vec is significantly superior than Avg-Vec in the other criterias.
- in Retention (Accuracy), Avg-Vec and Par-Vec approaches are statistically similar

6. Conclusions

The focus of this paper is on the analysis of employee opinions and sentiments. Such analysis enables learning effective models that can predict variables such as salary, retention, work/life balance, and career opportunities, based solely on employee reviews. We evaluate the standard TF-IDF bag-of-words approach to represent the reviews, and also more sophisticate approaches based on the skip-gram and paragraph vector represention. We collected employee reviews from social media platforms, as well as official data about salary and retention. We performed an extensive set of experiments and conclude that review representations obtained using skip-gram and paragraph vector approaches offer similar performance. Also, our experiments showed that both approaches are significantly superior than the standard BoW approach.

References

Baeza-Yates, R. and Ribeiro-Neto, B. (2011). *Modern Information Retrieval - the concepts and technology behind search, Second edition*. Pearson Education Ltd., Harlow, England.

Blei, D., Ng, A., and Jordan, M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Deerwester, S., Dumais, S., Landauer, T., Furnas, G., and Harshman, R. (1990). Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Devitt, A. and Ahmad, K. (2007). Sentiment polarity identification in financial news: A cohesion-based approach. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Drucker, H., Burges, C., Kaufman, L., Smola, A., and Vapnik, V. (1996). Support vector regression machines. In *Advances in Neural Information Processing Systems 9*, pages 155–161.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57.

Joachims, T. (1998). Text categorization with suport vector machines: Learning with many relevant features. In *10th European Conference on Machine Learning*, pages 137–142.

Joachims, T. (2006). Training linear svms in linear time. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning*, pages 1188–1196.

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Maas, A., Daly, R., Pham, P., Huang, D., Ng, A., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics*, pages 142–150.

Martineau, J. and Finin, T. (2009). Delta TFIDF: an improved feature space for sentiment analysis. In *Proceedings of the 3rd International Conference on Weblogs and Social Media*.

Mesnil, G., Mikolov, T., Ranzato, M., and Bengio, Y. (2015). Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. In *Proceedings of Workshop at International Conference on Learning Representations*.

Mikolov, T., Chen, K., Conrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Mikolov, T., Yi, W., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of North American Chapter of the Association for Computational Linguistics - Human Language Technologies*.

Mukherjee, A. and Liu, B. (2012). Modeling review comments. In *The 50th Annual Meeting of the Association for Computational Linguistics*, pages 320–329.

Ounis, I., Macdonald, C., and Soboroff, I. (2008). On the TREC blog track. In *Proceedings of the 2nd International Conference on Weblogs and Social Media*.

Paltoglou, G. and Thelwall, M. (2010). A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1386–1395.

Pang, B. and Lee, L. (2007). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Pohlert, T. (2014). The pairwise multiple comparison of mean ranks package (pmcmr).

Thomas, M., Pang, B., and Lee, L. (2006). Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335.

sbbd:09

Heurísticas para Aprimorar o Método BMW e suas Variantes

Lídia Lizziane Serejo Carvalho[1], **Edleno Silva de Moura**[1],
Caio Moura Daoud [1], **Altigran Soares da Silva** [1]

[1]Instituto de Computação (IComp)
Universidade Federal do Amazonas (UFAM)
Manaus – Am – Brasil

lidializz@gmail.com, {edleno, caiodaoud, alti}@icomp.ufam.edu.br

Abstract. *In this paper, we propose and evaluate heuristics to improve the performance of BMW and its variants. The proposed changes retain ownership to preserve the order of the first results at the end of processing, offering benefits in an attempt to both further reduces query processing times and the amount of memory required for processing queries.*

Resumo. *Neste artigo são propostas e experimentadas modificações nas heurísticas de descarte de documentos utilizadas para processar consultas no algoritmo BMW e suas variantes. As modificações propostas mantém a propriedade de preservar a ordem dos primeiros resultados obtidos ao final do processamento, oferecendo como benefícios a redução no tempo de processamento de consultas e na quantidade de memória utilizada durante o processamento.*

1. Introdução

Processadores de consulta em sistemas de busca são implementados visando minimizar custos em relação ao tempo de processamento e uso de memória. Seu principal objetivo é tomar uma consulta especificada pelo usuário e fornecer um ranking com os documentos considerados relevantes para a consulta. Para isso, utilizam modelos de Recuperação da Informação que produzem funções de *ranking* como o modelo vetorial [Salton et al. 1975] e o modelo BM25 [Robertson and Walker 1994], que computam um peso para cada documento dada uma consulta. Em geral os sistemas retornam um *ranking* com os top-k documentos com maior peso, onde *k* é um número predefinido que corresponde a quantidade de documentos que serão retornados.

Ding e Suel (2011) propuseram o algoritmo Block-Max WAND (BMW), um algoritmo para processamento de consultas que mostrou-se mais veloz que os propostos anteriormente na literatura. Variantes do método BMW têm sido propostas e estudadas na literatura, incluindo o BMW-CS [Rossi et al. 2013], o qual reduz o tempo de processamento de consultas quando comparado ao BMW, mas tem como desvantagem um aumento significativo na quantidade de memória necessária para processar consultas e a possibilidade de perder resultados ao final do processamento, alterando assim a lista de documentos que aparecem no topo da resposta dada a cada consulta.

Algoritmos como o BMW utilizam técnicas de descarte de documentos (técnicas de poda) que estão diretamente relacionadas às atuais heurísticas que visam a redução de custos de processamento. O algoritmo mantém uma estrutura de *heap* de mínimo que armazena os top-k documentos durante o processamento. O menor peso entre todos os

documentos presentes no *heap* é usado como um limiar de descarte. Assim durante o processamento só são computados os escores dos documentos que têm chance de superar o limiar de descarte.

Este trabalho tem como principal objetivo estudar e propor melhorias ao algoritmo BMW e seu variante BMW-CS com a finalidade de reduzir o tempo de processamento e quantidade de memória utilizada ao processar consultas. Como principais resultados, obteve-se as seguintes contribuições: (i) Redução no tempo de processamento, com uma heurística de adoção de limiar de poda inicial que garante a integridade dos resultados. (ii) Redução no uso de memória decorrente de redução significativa no tamanho de uma estrutura auxiliar empregada pelos algoritmos no processamento de consultas, as *skiplists*, que são utilizadas para acelerar o processamento de consultas. Nos dois casos, as mudanças realizadas apresentam a vantagem de não alterar o resultado inicial provido pelos métodos (BMW ou BMW-CS). Como o BMW é um algoritmo que preserva resultados do topo da resposta para cada consulta, nossas alterações sobre o BMW também possuem tal propriedade.

O presente trabalho está estruturado como segue. A Seção 2 expõe um resumo dos principais trabalhos relacionados presentes na literatura. Nas Seções 3 e 4, explicamos as estratégias propostas para a solução do problema apresentado. A Seção 5 descreve os experimentos realizados e os resultados obtidos. E por fim, na Seção 6, discutimos as conclusões e direcionamentos para trabalhos futuros.

2. Trabalhos Relacionados

Diversos pesquisadores têm proposto estratégias para melhorar o desempenho do processamento de consultas em sistemas de busca. Os métodos de processamento de consultas encontrados na literatura podem ser divididos em duas classes principais : os que utilizam estratégia TAAT (*Term-At-A-Time*) e os que utilizam estratégia DAAT (*Document-At-A-Time*) [Baeza-Yates et al. 2011]. Nos métodos que utilizam a estratégia TAAT, as listas invertidas são ordenadas pelo peso dos documentos. Com isso, a consulta é processada um termo por vez. Sua maior desvantagem é que esta estratégia requer muita memória para armazenar os escores alcançados por cada documento em cada termo da consulta. Nos métodos que utilizam estratégia DAAT, as listas invertidas são ordenadas pelo *id* do documento permitindo que as listas sejam percorridas em paralelo e que o escore completo de um documento seja computado em cada iteração. Outra característica desta estratégia é o uso de *skiplists* associadas a cada lista invertida. Estas estruturas armazenam ponteiros para entradas da lista invertida dividindo-a em blocos para facilitar o acesso aos documentos contendo tipicamente entre 128 e 256 entradas [Broder et al. 2003].

Um dos trabalhos mais importantes apresentados nos últimos anos propõe um método de processamento de consultas DAAT conhecido como WAND (*Weak* AND ou *Weighted* AND) [Broder et al. 2003]. Neste método, durante a indexação da coleção de documentos é armazenado o maior escore de cada lista invertida. Durante o processamento, saber o escore máximo que um documento pode atingir é importante para evitar o custo de computar escores de documentos que não têm chance de alterar o topo do *ranking* de respostas. Durante o processamento de uma consulta, documentos que estão sendo avaliados, chamados de pivôs, são analisados em duas etapas. Primeiro, obtém-se o *Max Score*, escore máximo de cada lista onde o pivô tem chance de ocorrer. Quando

o *Max Score* for maior que o limiar de poda atual, a lista invertida de cada termo será acessada para que o documento pivô tenha seu escore real computado. Quando o *Max Score* não supera o limiar de poda, o documento pivô é descartado e um novo pivô é selecionado.

O algoritmo BMW [Ding and Suel 2011] foi criado tendo como base o WAND [Broder et al. 2003], porém propõe uma solução otimizada que abrange principalmente uma modificação nas *skiplists*. No BMW, para cada entrada das *skiplists*, é armazenado o maior escore do bloco. A estratégia de poda é similar à adotada no método WAND, com a vantagem de possuir um escore máximo mais próximo do escore real de cada documento. Com essa vantagem, o método BMW descarta ainda mais documentos que não apresentam peso suficiente para serem inseridos no topo do *ranking* de respostas. A poda no BMW acontece em dois momentos, (i) quando o *Max Score* não supera o limiar de descarte, como no método WAND, e (ii) quando o *Block Max Score* não supera o limiar de descarte. O *Block Max Score* é computado com base no escore máximo dos blocos em que o documento pode ocorrer e só é verificado quando o *Max Score* supera o limiar de descarte.

O limiar de descarte é dinamicamente atualizado sempre que um documento tem escore maior que o do atual limiar. Este documento atualiza o *heap* de respostas e o limiar de poda é atualizado com o peso do menor documento do *heap*.

Para exemplificar, supondo um cenário como mostra a Figura 1, o limiar de poda indica que é necessário que um documento tenha um escore maior que $3,2$ para atualizar o atual *ranking* de respostas. A figura retrata um momento do processamento onde os documentos 20, 40 e 90 associados a cada termo da consulta "amarelo verde azul" serão avaliados, nesta ordem. A lista invertida do primeiro documento a ser processado tem um escore máximo de $2,5$, ou seja, o documento 20, que só ocorre na lista do termo amarelo, não tem peso suficiente para superar o limiar de poda. No entanto, o documento 40, que pode ocorrer na lista invertida do termo "amarelo", poderá ter um escore de até $4,0$ $(2,5+1,5 > 3,2)$, maior que o limiar de poda atual. Já que existe tal possibilidade, o algoritmo verifica o *Block Max Score* do pivô (documento 40) . Caso o pivô ocorra na lista invertida do termo "amarelo" e ele tenha o maior escore dentre os de seu bloco, nas duas listas, seu escore máximo será então $3,3$ $(2,3+1,0 > 3,2)$, portanto superior ao limiar de poda. Após tais verificações, o documento 40 tem seu escore completo avaliado. Neste exemplo, o escore completo do documento é 3.3 $(2,3+1,0)$, que ultrapassa novamente o limiar de poda, portanto o documento 40 entra no *heap* de respostas atualizando o limiar de poda. Nesta etapa, os ponteiros das listas invertidas são movidos para que apontem para o próximo documento maior que o pivô avaliado, e o algoritmo então prossegue repetindo o processo.

Alguns autores exploram a abordagem do particionamento da lista invertida em mais de uma camada para acelerar o processamento de consultas. Uma das mais recentes, e também mais bem sucedidas estratégias de particionamento é adotada pelo algoritmo BMW-CS [Rossi et al. 2013], o qual divide cada lista invertida em duas camadas, a primeira camada tem tamanho menor e é criada usando entradas que apresentam os maiores escores das listas invertidas, enquanto a segunda camada tem um índice muito maior e contém as entradas restantes. O processamento é dividido em duas fases. A primeira, chamada de seleção de candidatos, processa a primeira camada das listas dos termos da

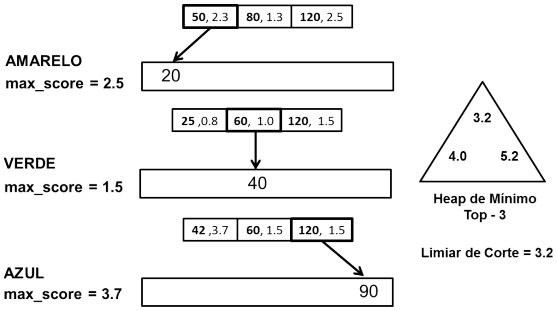


Figura 1. Listas invertidas para os termos da consulta "amarelo verde azul".

consulta. Nesta fase, o processo é similar ao utilizado no método BMW mas considera também o escore máximo do documento na segunda camada. O valor para a segunda camada é importante, pois podem existir documentos que estão na primeira camada mas também ocorrem na segunda camada para outros termos da consulta. Os documentos selecionados na primeira fase são chamados de candidatos à resposta. Na segunda fase do processo, a segunda camada é percorrida apenas para buscar ocorrências dos documentos candidatos na segunda camada e computar o escore real desses documentos.

O método BMW-CS apresenta a desvantagem de não preservar todas as top-k respostas de uma consulta, pois documentos que não ocorrem na primeira camada de pelo menos um dos termos da consulta não são avaliados. Nestes casos, mesmo que os documentos tenham escores suficientes para entrar na resposta, eles serão descartados pelo algoritmo BMW-CS.

3. Estudo de Limiares Iniciais para Poda

O limiar de poda é determinante para que as estratégias de descarte de documentos funcionem nos métodos apresentados. Em um primeiro momento do processamento, enquanto o *heap* de respostas não está cheio, o limiar de poda é igual a zero. Isso significa que a estratégias de poda não são aplicadas pelo BMW inicialmente. Assim que os primeiros k documentos são computados, também é obtido o primeiro limiar de poda diferente de zero, dando inicio ao processo de descarte de documentos que reduz o custo de processamento de consultas no algoritmo BMW. Na medida em que mais documentos são processados, o limiar de poda aumenta até atingir o valor mínimo do *ranking* final.

A presença de um limiar inicial baixo faz com que menos documentos sejam descartados no inicio do processamento de consultas do algoritmo BMW e seus variantes. Nossa primeira tentativa de melhorar os algoritmos foi estudar o desempenho de tais algoritmos ao utilizar como limiar de poda inicial uma estimativa de valor que seja ao mesmo tempo segura, sendo garantidamente menor que o limiar de poda obtido ao final do processamento, e grande o suficiente para ter impacto nos custos de processamento, tendo valor o mais próximo possível do limiar de poda final.

A estratégia para adoção de limiar inicial foi aplicada em dois cenários: quando estamos interessados em computar os top-10 resultados e os top-1000 para uma dada consulta. Os dois cenários são considerados típicos em sistemas de busca e são normalmente estudados em artigos que tratam do processamento eficiente de consultas [Rossi et al. 2013]. Para estimar o limiar inicial de poda, guardamos durante a

indexação o k-ésimo maior escore de cada palavra da coleção em sua entrada correspondente presente na estrutura de índice invertido. Dada uma consulta, estima-se um valor inicial de escore mínimo tomando-se o maior k-ésimo escore dentre os termos da consulta. Tal escolha garante que o limiar inicial não irá descartar respostas, pois sabe-se que há pelo menos k entradas com escore pelo menos igual ao limiar inicial escolhido. Por outro lado, a adoção de tal limiar permite que se descarte a priori entradas que seriam levadas em consideração quando adotado um limiar inicial igual a zero.

Além de propor e experimentar estratégias para determinar um limiar inicial, fez-se também um estudo para verificar quanto se pode ganhar em desempenho com esse tipo de estratégia. Para isso, foi armazenado o mínimo escore entre as k respostas obtidas no processamento de consultas, e depois as mesmas consultas foram novamente processadas, porém iniciando o limiar de poda com seu valor máximo. Denominamos este valor como limiar ótimo, por ser um indicativo do maior ganho que poderíamos obter com a utilização da estratégia de limiar de poda inicial. O limiar ótimo é apenas um valor de referência, pois para obtê-lo é necessário que a consulta seja primeiro processada integralmente.

4. Blocos de Tamanho Variáveis

Nos experimentos apresentados por [Ding and Suel 2011] e [Rossi et al. 2013] foram utilizados blocos fixos de 128 documentos na criação das *skiplists*. Isso quer dizer que a cada 128 documentos em uma lista invertida é criada uma entrada na *skiplist* para acelerar o acesso a documentos e guardar informações de descarte, como o maior escore de cada bloco (*Block Max Score*). Como vimos nas Figuras 3(a) e 3(b) e na Tabela 2, quanto menor o tamanho dos blocos, menor o tempo de processamento. Esse custo está diretamente relacionado à busca do documento nos blocos para computar o escore real.

Sempre que o *Max Score* e *Block Max Score* de um pivô superar o limiar de poda, os blocos correspondentes serão acessados em busca do documento pivô para que seu escore real seja computado. Blocos são totalmente descartados quando o *Block Max Score* do pivô não supera o limiar de poda. Com isso, percebe-se que blocos com *Block Max Score* baixo apresentam maior probabilidade de descarte. Neste caso, inferimos que blocos adjacentes que apresentam valor baixo de *Block Max Score* poderiam ser integrados para que mais documentos possam ser descartados durante uma iteração do processador de consultas.

Tabela 1. Desempenho do algoritmo BMW para blocos de tamanho fixo de 64, 128 e 256 documentos. A quantidade de blocos é apresentada com *.

		Pivôs	Tempo(ms)
Blocos de 64	Top-10	369.781.344	20
23.595.719*	Top-1000	1.002.528.576	51
Blocos de 128	Top-10	426.335.552	22
11.798.404*	Top-1000	1.116.841.472	55
Blocos de 256	Top-10	495.183.424	27
5.899.760*	Top-1000	1.223.783.040	59

Para determinar se o *Block Max Score* é baixo ou não, utilizou-se como referência o milésimo maior escore de cada termo da coleção, fazendo-se uma associação com a quantidade típica de entradas normalmente considerada nos resultados de busca. Logo,

blocos com *Block Max Score* menor que o valor de referência são considerados blocos de baixo impacto, e portanto podem se unir a outro bloco.

A Figura 2 ilustra uma lista invertida para um termo *A* que está dividida em n blocos, cada um composto por 128 documentos. No exemplo, o milésimo maior escore desta lista já foi computado e tem valor igual a 12. Neste cenário, os blocos A_2 e A_3 são combinados por apresentarem escore máximo menor que o valor de referência 12. Os blocos A_1, A_4 e An permanecem como estavam. O novo bloco $A_2 + A_3$ formado terá 256 documentos e *Block Max Score* igual a 8.

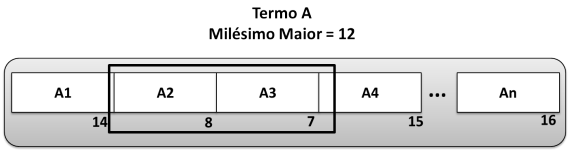


Figura 2. Exemplo de lista invertida com blocos variáveis.

Nossa hipótese é que a heurística de união de blocos descrita acima pode ser usada para aprimorar o processo de poda em sistemas de busca como o BMW e o BMW-CS. Na seção de experimentos, mostramos que nossa hipótese se comprova na prática e a heurística proposta, quando combinada com os benefícios de nossa proposta de estimar um limiar inicial de poda, resulta em melhora significativa nos indicadores de desempenho dos dois algoritmos.

5. Experimentos

Para os experimentos, utilizamos e modificamos o sistema de busca implementado por [Rossi et al. 2013]. O sistema adotado processa consultas utilizando o algoritmo BMW ou o variante BMW-CS. Utilizou-se nos experimentos a coleção de referência TREC GOV2. Esta é composta por $25.205.179$ milhões de páginas coletadas do domínio .gov no início de 2004. A TREC GOV2 possui 426 GB de texto, compostos por páginas HTML e texto extraído de páginas no formato PDF e postscript (PS). O índice inteiro possui aproximadamente 7 GB de listas invertidas e o vocabulário é composto por cerca de 4 milhões de termos distintos. A coleção TREC GOV2 foi adotada por ter se tornado um padrão para estudos que propõem melhorias de eficiência em sistemas de busca, tendo sido usada nos experimentos dos principais trabalhos encontrados na literatura.

Selecionou-se aleatoriamente um conjunto de 10.000 consultas extraídas da coleção TREC 2006 *efficiency queries*. Todos as *stopwords* dessas consultas foram retiradas. Durante o processamento de consultas, o índice inteiro é carregado para a memória. Todas as configurações foram escolhidas por serem similares às adotadas em trabalhos similares [Strohman and Croft 2007, Ding and Suel 2011], o que torna mais fácil a comparação entre os métodos estudados. Os experimentos foram executados em uma máquina Intel(R) Xeon(R), com processador X5680, 3.33GHz e 64GB de memória. Todos os experimentos foram executados em cenários que retornavam top-10 ou top-1000 respostas. A recuperação das top-1000 respostas foi incluída para simular um ambiente onde o conjunto top-1000 é utilizado como entrada para um método de *ranking* mais sofisticado, como por exemplo o uso de uma técnica de aprendizado de máquina. O cenário de top-10 respostas foi incluído para simularmos um cenário mais comum, onde o usuário

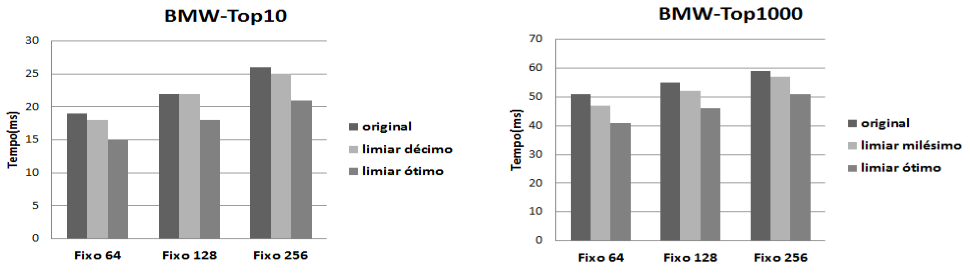
está interessado apenas em uma pequena lista de resultados para a sua consulta. Os algoritmos foram avaliados em termos de tempo de processamento e memória utilizada.

As estratégias foram implementadas no algoritmo BMW, proposto por [Ding and Suel 2011], e o algoritmo BMW-CS, proposto por [Rossi et al. 2013]. Estas foram desenvolvidas em C++ tendo como base trabalhos anteriores. Assim como os algoritmos originais, mantemos a utilização do modelo probabilístico Okapi BM25 como função de similaridade. Para o algoritmo BMW-CS, variamos o tamanho da primeira camada de 2 em 2% até 50% do índice completo nos experimentos.

5.1. Resultados Com Limiar de Poda Inicial

As Figuras 3(a) e 3(b) mostram ganhos nas estratégias propostas em relação ao algoritmo BMW sem limiar de poda inicial (BMW Original) para as top-10 e top-1000 consultas, respectivamente. As figuras apresentam os experimentos realizados também ao variarmos o tamanho dos blocos gerados nas listas invertidas. Foram experimentados tamanho fixo de 64, 128 e 256 documentos. Blocos fixos com 64 documentos resultam em tempos de processamento de consultas menores, porém como indicado na Tabela 2, formam mais blocos no índice invertido, requerendo mais memória para armazenar as *skiplists*. De maneira oposta, blocos fixos com 256 documentos formam menos blocos, no entanto resultam em tempo de processamento maior. Quanto maior a quantidade de blocos formados (número de entradas da estrutura *skiplist*), maior é o custo de memória usado no processamento de consultas, pois será necessário mais espaço para o armazenamento destes dados.

O tamanho de bloco utilizado tipicamente em trabalhos apresentados na literatura é de 128 entradas. Nesta configuração, observa-se que a escolha de um limiar inicial ótimo resultaria em ganho de cerca de 18% do tempo de processamento para as top-10 respostas e 16% para as top-1000 respostas, o que seria o limite para qualquer estratégia de estimativa de limiar inicial. Como podemos observar também na Figura 3, nossa heurística de estimativa de limiar teve impacto para top-1000 respostas, com uma redução aproximada de 5,5% em relação ao algoritmo original. Os maiores ganhos ocorreram com blocos de tamanho 64, onde temos reduções aproximadas de 5,2% e 7,8% para top-10 e top-1000 respostas, respectivamente.



(a) (b)

Figura 3. Desempenho do algoritmo BMW em diferentes abordagens para limiares de poda iniciais: BMW original, BMW com décimo/milésimo maior escore e BMW com o mínimo escore das top-k respostas para blocos fixos de 64, 128 e 256 documentos.

Tabela 2. Número total de entradas das *skiplists* para blocos fixos de 64, 128 e 256 documentos.

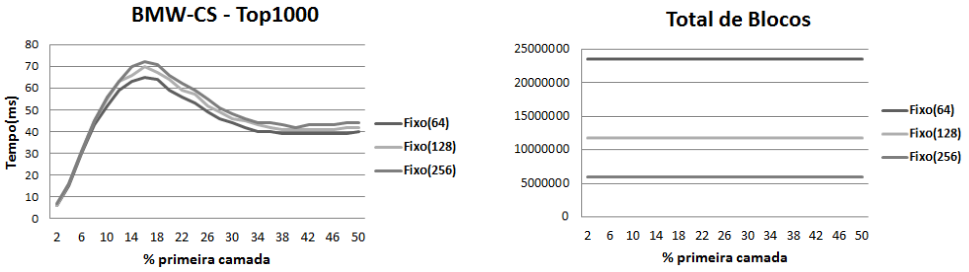
Fixo 64	Fixo 128	Fixo 256
23.595.719	11.798.404	5.899.760

No algoritmo BMW-CS, para blocos fixos de 64, 128 e 256 documentos, nas duas camadas, os maiores ganhos do método com estratégia de limiar inicial são obtidos para top-1000 respostas. Dentro deste cenário, para a análise de tempo de processamento, calculamos a média de tempo entre as variações de 18% e 50% da primeira camada para todas as opções de implementação, como mostra a Tabela 3. A tabela mostra que a introdução do milésimo como estratégia de limiar inicial reduz o tempo de processamento de consultas em até aproximadamente 17% .

Tabela 3. Média do tempo de processamento em milissegundos calculada a partir da variação de 18% da primeira camada para top-1000 respostas.

	Fixo 64	Fixo 128	Fixo 256
BMW-CS original	54,65	57,94	59,94
BMW-CS c/ limiar	45,12	47,82	49,88
Redução (%)	17,44	17,46	16,78

Temos nas Figuras 4(a) e 4(b) respectivamente, o comportamento da implementação da estratégia de limiar inicial e o total de blocos formados para os três cenários, quando variarmos o tamanho da primeira camada. Percebe-se que não há muita oscilação no total de blocos na medida em que aumentamos a porcentagem da primeira camada. Nota-se ainda que o uso de blocos fixos de 64 entradas gera custos de memória expressivos e dada sua semelhança em relação ao tempo de processamento com blocos fixos de 128 e 256 (Figura 4(a)), ele não se mostra como uma boa opção de implementação. Tal cenário, apresenta a média aproximada de 23,6 mil blocos, enquanto blocos de 128 e 256 documentos formam aproximadamente 12 mil e 6 mil blocos, respectivamente. Portanto, como verifica-se na Figura 4(a), visto que o tempo de processamento entre os cenários são bem similares, podemos inferir que a melhor implementação é aquela que apresenta o menor número de blocos criados, ou seja, o cenário de blocos fixos com 256 entradas.



(a) Processamento de consultas com limiar inicial

(b) Total de entradas nas skiplists

Figura 4. Desempenho e total de blocos fixos do algoritmo BMW-CS.

5.2. Variando o Tamanho Dos Blocos

Começamos com a avaliação do impacto da variação no tamanho dos blocos no BMW. Como primeira estratégia, estabelecemos um tamanho mínimo e máximo para variação do tamanho dos blocos. Variamos blocos com no mínimo 16, 32, 64, 128, 256 e 512 documentos, compondo ao máximo o total de 1024 documentos. A Figura 5 mostra um comparativo desta estratégia com o algoritmo BMW Original. Percebe-se o aumento no tempo de processamento para top-10 e top-1000 respostas. Para top-1000 respostas retornadas, ao aplicarmos a variação com blocos de 512 até 1024 documentos, o tempo aumenta de 52 milissegundos para um pouco mais de 60 milissegundos. Nota-se ainda na figura que, apesar do tempo de processamento aumentar, a utilização da estratégia diminui em até aproximadamente 80% o número total de blocos formados (Figura 5(c)) quando comparamos com o algoritmo BMW Original.

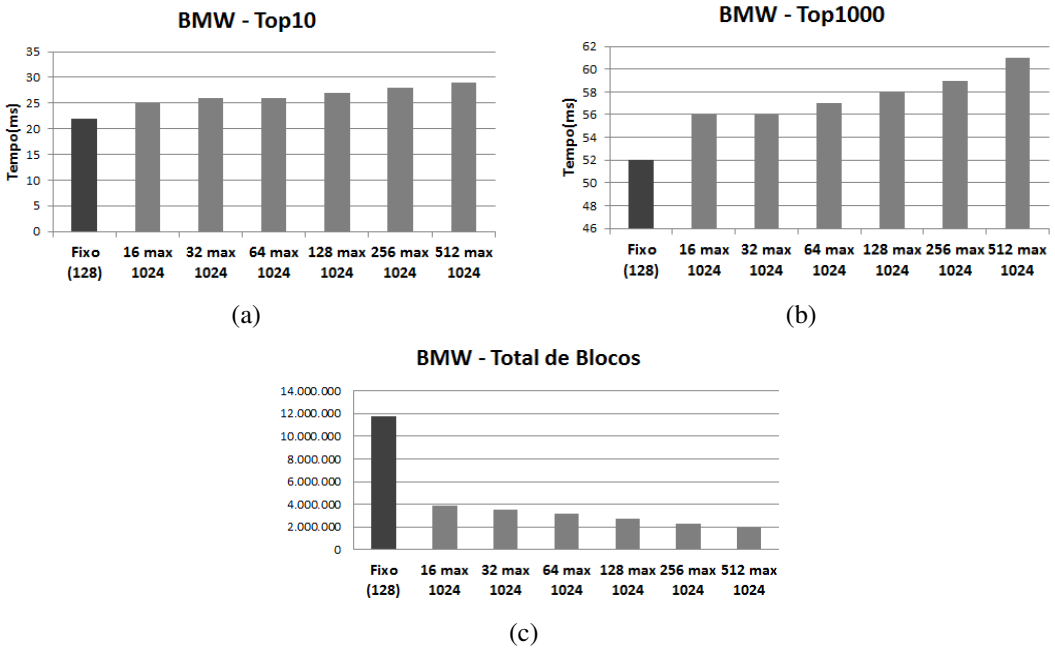


Figura 5. Desempenho do algoritmo BMW com a utilização das abordagem de variação do tamanho de blocos (máximo 1024 documentos).

Outra estratégia adotada com intuito de reduzir o tempo de processamento com blocos variados foi reduzir o tamanho máximo de 1024 para 128 ou 256 documentos. O gráfico apresentado na Figura 6 mostra os tempos alcançados com essas novas variações. Há redução no tempo de processamento para índices formados por blocos com no máximo 128 documentos para top-10 e top-1000 respostas. Ainda na figura, percebe-se que podemos manter o tempo de processamento similar ao BMW Original quando se utiliza o tamanho máximo de 256 documentos por bloco. Nota-se na Figura 6(c) que com esta opção de implementação reduzimos em até aproximadamente 44% o total de blocos formados. Por exemplo, gerar blocos de tamanho mínimo 32 e máximo 256 documentos, resultou em tempo de processamento igual ao obtido pelo BMW original (52 milisegundos). Contudo, o número de blocos gerados por tal opção foi 35% menor, gerando 7.676.292 blocos contra 11.798.404 do BMW original, o que significa um ganho significativo em termos de redução da memória extra gerada para armazenar informação sobre cada bloco.

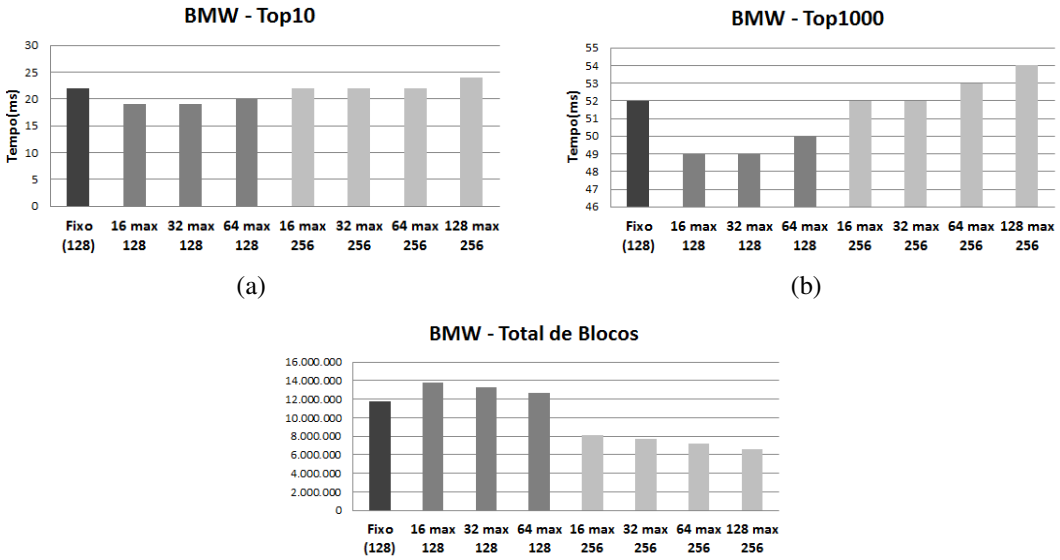


Figura 6. Desempenho do algoritmo BMW com a utilização das abordagem da variação do tamanho de blocos (máximo 128 e 256 documentos).

Alterações no tamanho dos blocos podem gerar efeitos colaterais no desempenho dos algoritmos, dado que elas também alteram os valores de escore máximo dos blocos e, consequentemente, afetam também as estratégias de poda de documentos, tanto do BMW, quanto do BMW-CS. No entanto, como foi visto nesta seção, podemos ter uma redução do número total de blocos formados sem ter impacto negativo no tempo de processamento. Unindo-se o uso de blocos de tamanho variável à estratégia de inserção de limiar inicial, conseguimos obter ganhos no tempo de processamento e, ao mesmo tempo, redução no tamanho das *skiplists* usadas para representar os blocos de documentos durante o processamento de consultas.

Também realizamos experimentos para estudar a aplicação de blocos de tamanho variável ao algoritmo BMW-CS. Mudamos o tamanho dos blocos de 64 até 1024 documentos nas duas camadas e analisamos o desempenho do algoritmo para cada porcentagem de variação da primeira camada, para top-10 e top-1000 respostas. Todas as opções de implementações (com a primeira, segunda ou ambas camadas variáveis) foram comparadas ao algoritmo BMW-CS usando blocos fixos com 128 documentos e a estratégia de limiar inicial proposta por nós. Os experimentos mostram que não obtivemos ganhos no tempo de processamento tanto para o cenário de top-10 como top-1000 respostas. A Tabela 4 mostra a média de tempo entre as variações de 18% e 50% da primeira camada para todas as opções de implementação no cenário de top-1000 respostas. Constata-se que os tempos de processamento obtidos pelas abordagens são bem similares.

Tendo em vista que o tempo de processamento entre as estratégias de blocos variáveis não apresenta grandes variações, podemos levar em consideração, como nas abordagens anteriores, a possibilidade de reduzir custos de memória. A Figura 7 faz um comparativo do total de blocos entre todas as opções de implementação, inclusive a utilizada no algoritmo BMW-CS original. Ainda na figura, notamos que o algoritmo BMW-CS fixo com 128 documentos configura a pior hipótese de implementação, visto

Tabela 4. Blocos Variáveis - Média do tempo de processamento em milisegundos calculada a partir da variação de 18% da primeira camada.

	1ª camada variável	2ª camada variável	1ª e 2ª camadas variáveis
Mín 64 Máx 256	48,47	48,88	49,18
Mín 64 Máx 512	48,76	49,65	50,41
Mín 64 Máx 1024	49,00	50,12	51,35
BMW-CS Original: 57,94			
BMW-CS Original c/ limiar: 47,82			

que o mesmo totaliza a média de aproximadamente $11,8$ milhões de blocos formados.

A opção de implementação de variar as duas camadas com o mínimo de 64 e máximo 256 documentos gera uma redução aproximada de 43% em média em relação ao BMW-CS original. Tais resultados são obtidos com uma mudança pequena no tempo de processamento. Por exemplo, em nossos experimentos, o tempo médio de processamento obtido ao considerarmos todos os limiares de tamanho estudados para a primeira camada (Tabela 4) no algoritmo BMW-CS original, que utiliza a estratégia de limiar inicial, é de $47,82$(ms), enquanto que temos um tempo médio de $49,18$(ms) ao variamos o tamanho dos blocos nas duas camadas com o mínimo de 64 e máximo 256 documentos.

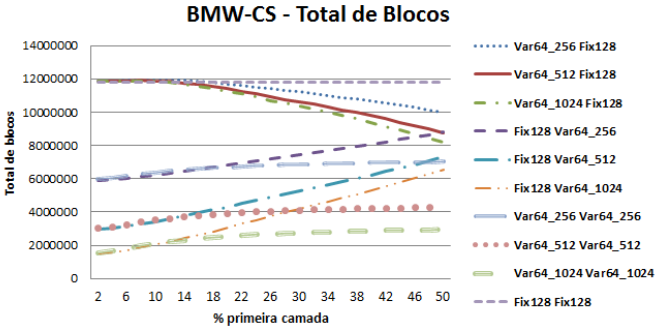


Figura 7. Número total de entradas das skiplists para cada porcentagem da primeira camada - blocos variáveis.

6. Conclusões e Trabalhos Futuros

Os resultados obtidos mostram que a estimativa de limiares iniciais para descartes de documentos pode melhorar o desempenho do algoritmo BMW e sua variante BMW-CS tanto para o cenário de top-10 como top-1000 respostas. Os maiores ganhos ao modificarmos o BMW para usar nossa estratégia de limiar inicial de poda ocorreram com blocos fixos de tamanho 64, havendo redução de cerca de 5,2% e 7,8% no tempo de processamento para top-10 e top-1000 respostas, respectivamente. Obteve-se ganhos ainda maiores ao modificarmos o algoritmo BMW-CS, com redução de aproximadamente 17% no tempo de processamento.

A estratégia de variação do tamanho dos blocos da estrutura de índices não resultou na redução de tempo de processamento, mas proporcionou reduções significativas no número de entradas das *skiplists*, tendo impacto portanto na quantidade de memória extra requerida por cada método para processar consultas. Para o algoritmo BMW, a melhor configuração encontrada nos experimentos empregou blocos com o máximo de 256

entradas. Nela mantivemos o mesmo tempo de processamento do algoritmo original e reduzimos em cerca de 35% o número total de blocos criados. No algoritmo BMW-CS, a opção de variar o tamanho dos blocos nas duas camadas com o mínimo de 64 e máximo 256 documentos gerou em nossos experimentos uma redução média de 43% em relação ao BMW-CS original, sem mudanças significativas no tempo de processamento.

Como trabalho futuro, seria interessante estudar o impacto das técnicas aqui implementadas em cenários onde o índice do sistema de busca não cabe em memória. Em tais situações, ganhos de espaço nas *skiplists* podem tornar-se mais relevantes e ter impacto positivo no tempo de processamento, uma vez que *skiplists* menores têm maior chance de serem colocadas completamente em memória. Também seria interessante o estudo do impacto das heurísticas de detecção de limiares quando aplicados a outros problemas práticos, como o da classificação de documentos [Cristo et al. 2003].

Referências

Baeza-Yates, R., Ribeiro-Neto, B., et al. (2011). *Modern information retrieval*. ACM press New York.

Broder, A. Z., Carmel, D., Herscovici, M., Soffer, A., and Zien, J. (2003). Efficient query evaluation using a two-level retrieval process. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 426–434. ACM.

Cristo, M., Calado, P., de Moura, E. S., Ziviani, N., and Ribeiro-Neto, B. A. (2003). Link information as a similarity measure in web classification. In *String Processing and Information Retrieval, 10th International Symposium, SPIRE 2003, Manaus, Brazil, October 8-10, 2003, Proceedings*, pages 43–55.

Ding, S. and Suel, T. (2011). Faster top-k document retrieval using block-max indexes. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 993–1002. ACM.

Robertson, S. E. and Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241. Springer-Verlag New York, Inc.

Rossi, C., de Moura, E. S., Carvalho, A. L., and da Silva, A. S. (2013). Fast document-at-a-time query processing using two-tier indexes. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 183–192. ACM.

Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Strohman, T. and Croft, W. B. (2007). Efficient document retrieval in main memory. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 175–182. ACM.

sbbd:10

SimDataMapper: An Architectural Pattern to Integrate Declarative Similarity Matching into Database Applications

Natália Cristina Schneider[1], Leonardo Andrade Ribeiro[2], Andrei de Souza Inácio[3], Harley Michel Wagner[3], Aldo von Wangenheim[3]

[1]Departamento de Ciência da Computação
Universidade Federal de Lavras (UFLA) – Lavras, MG – Brazil

[2]Instituto de Informática
Universidade Federal de Goiás (UFG) – Goiânia, GO – Brazil

[3]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brazil

natalia_schneider@comp.ufla.br, laribeiro@inf.ufg.br

{andrei, harley, awangenh}@inf.ufsc.br

Abstract. *Effective manipulation of string data is of fundamental importance to modern database applications. Very often, textual inconsistencies render equality comparisons meaningless and strings have to be matched in terms of their similarity. Previous work has proposed techniques to express similarity operations using declarative SQL statements. However, the non-trivial issue of embedding similarity support into object-oriented applications has received little attention. Particularly, a number of indexes have to be maintained for each similarity predicate, thereby severely complicating persistence of application objects. In this paper, we present SimDataMapper, an architectural pattern to provide easy, efficient, and flexible integration of declarative similarity matching with applications and programming environments. We describe implementation details and experimentally evaluate the performance of our approach.*

1. Introduction

Virtually all modern database applications deal with string data. Indeed, important information stored in databases such as people names, addresses, and product descriptions are represented as strings. Very often, string data have inconsistencies owing to a variety of reasons, such as misspellings and different naming conventions. Querying such inconsistent data is problematic because the traditional query paradigm based on exact matching in equality comparisons is meaningless. As a result, users are unable to find desired information in the database; even worse, the same information can be reinserted into the database with a different representation, thereby leading to even more inconsistency.

A better approach would be to apply the more general paradigm of *similarity matching*, where strings are matched in terms of their similarity. A *similarity function* is used to quantify the underlying notion of similarity and two strings are considered similar if the value returned by the similarity function for them is not less than a threshold. Considering that tuples are represented by a descriptive string attribute, a *similarity selection* returns tuples that are similar to a given query string [Koudas et al. 2007], whereas a

similarity join returns pairs of similar tuples from two input tables [Gravano et al. 2001]. For example, we can use similarity selections to find misspelled variants of patient names in a medical database and, likewise, similarity joins to identify tuples from two tables referring to the same patient.

Unfortunately, similarity matching is not directly supported by conventional RDBMSs. The direct approach of implementing similarity functions as user defined functions (UDFs) can be very expensive. In similarity selections, the UDF would be evaluated over all tuples of the input table, even those bearing no similarity to the query string. In similarity joins, the situation is much worse as the UDF would be applied over the cross-product of two input tables. Over the last years, several proposals addressed this problem by expressing similarity functions declaratively in SQL [Gravano et al. 2001, Gravano et al. 2003, Koudas et al. 2007]. Besides covering a wide range of notions of similarity, these works employ a number of filters and index tables to reduce the comparison space, thus achieving significant performance gains.

However, the above approaches are largely ad-hoc, which complicates their integration with application programs. For example, enabling similarity operations over a string field on the application side requires creation and maintenance of several index tables on the database side. As a direct consequence, the well-known "impedance mismatch" problem [Cook and Ibrahim], i.e., the problem caused by differences between programming language abstractions and persistent storage, is markedly exacerbated. The problem is much more severe when using similarity functions based on statistics, because these statistics need to be propagated to the indexes after updates on the original data [Koudas et al. 2007]. Moreover, similarity-aware database applications are likely to demand support of a variety of similarity functions — it is well-known that no similarity function performs consistently well across all domains [Cohen et al. 2003]. Such applications are thus faced with the daunting task of managing a huge number of indexes for each string attribute considered. We believe that these issues significantly hinder the widespread adoption of declarative similarity matching by applications beyond those for ad-hoc analysis over read-only data.

In this paper, we present SimDataMapper, an architectural pattern to facilitate integration of declarative similarity matching into object-oriented database applications. SimDataMapper allows loading in-memory objects based on similarity operations, while completely isolating the application from all details of the underlying similarity matching processing. In this context, a variety of similarity functions is uniformly supported by maintaining a common set of auxiliary tables. Furthermore, additional index tables can be flexibly created or dropped at any time based on the query/update ratio in the current workload. We present an efficient, flexible, and easily reusable implementation of the SimDataMapper, which is largely based on pure SQL, and experimentally validate our proposal on a dataset obtained from a production database.

The rest of the paper is organized as follows. In Section 2, we provide necessary background. In Section 3, we present our architectural pattern and describe its implementation details. Performance results are reported in Section 4. We discuss related work in Section 5, before we wrap up with the conclusions in Section 6.

2. Preliminaries

2.1. Mapping Strings to Sets

We map strings to *sets of tokens* using the popular concept of *q-grams*, i.e., substrings of length *q* obtained by "sliding" a window over the characters of an input string s. We (conceptually) extend s by prefixing and suffixing it with $q-1$ occurrences of a special character *"$"* not appearing in any string. Thus, all characters of s participate in exact *q* *q*-grams. For example, the string *"token"* can be mapped to the set of *2*-gram tokens {*$t*, *to*, *ok*, *ke*, *en*, *n$*}. As the result can be a multi-set, we append the symbol of a sequential ordinal number to each occurrence of a token [Chaudhuri et al. 2006] to convert multi-sets into sets, e.g, the multi-set {a,b,b} is converted to {*a∘1*, *b∘1*, *b∘2*}. Given a string s, we denote by $Q_q(s)$ its token set of substrings with length *q*.

We associate a weight with each token to obtain *weighted sets*. A widely adopted weighting scheme is the Inverse Document Frequency (*IDF*), which associates a weight $w(tk)$ to a token tk as follows: $w(tk)$=$ln(N/df(tk))$, where $df(tk)$ is the *document frequency*, i.e., the number of strings a token tk appears in a database of N strings. The intuition behind using IDF is that rare tokens are more discriminative and thus more important for similarity assessment. We obtain *unweighted sets* by associating the value of 1 to each token. The weight of a set x, denoted by $w(x)$, is given by weight summation of its tokens, i.e., $w(x) = \sum_{tk \in x} w(tk)$; note that we have $w(x) = |x|$ for unweighted sets. For ease of notation, given a string s, we use simply $w(s)$ instead of $w(Q_q(s))$ to denote its weighted (or unweighted) token set. Finally, $|s|$ denotes the length of s.

2.2. Similarity Functions

We focus on two classes of similarity functions, namely, token-based and edit-distance functions. The reasons for this choice are threefold: *token-based* and *edit-based* similarity functions 1) have been extensively used in many database applications, 2) lend themselves to efficient declarative realizations, and 3) share some common underlying principles that allow their support in a unified way.

The key idea behind token-based similarity functions is that most of the tokens derived from two similar strings should agree accordingly. Thus, the token sets of these strings would have a large overlap, which, in turn, would result in a high similarity value. Given two strings r and s, popular token-based similarity functions are defined as follows.

Definition 1. Let r and s be two strings. We have:

- *Jaccard similarity*: $J(r,s) = \frac{w(r \cap s)}{w(r \cup s)}$.
- *Dice similarity*: $D(r,s) = \frac{2 \cdot w(r \cap s)}{w(r)+w(s)}$.
- *Cosine similarity*: $C(r,s) = \frac{w(r \cap s)}{\sqrt{w(r) \cdot w(s)}}$

Example 1. Consider two strings r and s and their corresponding weighted token sets $w(r)$ ={⟨**c, 9**⟩,⟨**b, 7**⟩,⟨**d, 7**⟩,⟨**e, 7**⟩,⟨**h, 5**⟩,⟨`i, 5`⟩} and $w(s)$ ={⟨`a, 10`⟩,⟨**c, 9**⟩,⟨**b, 7**⟩,⟨**d, 7**⟩,⟨**e, 7**⟩,⟨`f, 5`⟩,⟨**h, 5**⟩} — note the token-weight association $\langle tk, w(tk) \rangle$, where tokens are represented by single characters. Then, we have $J(r,s) = \frac{35}{40+50-35} \approx 0.64$; $D(r,s) = 2 \cdot \frac{35}{40+50} \approx 0.78$; and $C(r,s) = \frac{35}{\sqrt{40 \cdot 50}} \approx 0.79$.

The dual notion of similarity is *distance* that quantifies the degree to which two entities are "different" or "far away". A popular distance measure is the *edit distance*,

Table 1. Overlap and size bounds.

Function	$O(r,s)$	$[min(r), max(r)]$
Jaccard	$\frac{\tau \cdot (w(r) + w(s))}{1+\tau}$	$\left[\tau \cdot w(r), \frac{w(r)}{\tau}\right]$
Dice	$\frac{\tau \cdot (w(r) + w(s))}{2}$	$\left[\frac{\tau \cdot w(r)}{2-\tau}, \frac{(2-\tau) \cdot w(r)}{\tau}\right]$
Cosine	$\tau \cdot \sqrt{w(r) \cdot w(s)}$	$\left[\tau^2 \cdot w(r), \frac{w(r)}{\tau^2}\right]$
ED	$max(\lvert r\rvert, \lvert s\rvert) - 1 - (\tau - 1) * q$	$[\lvert r\rvert - \tau, \lvert r\rvert + \tau]$
ES	$max(\lvert r\rvert, \lvert s\rvert) - 1 - ((1-\tau) \cdot max(\lvert r\rvert, \lvert s\rvert)) * q$	$[\lvert r\rvert \cdot \tau, \lvert r\rvert \cdot (2-\tau)]$

which is defined by the minimum number of character-editing operations to make two strings equal [Navarro 2001].

Definition 2. Let r and s be two strings. We have:

- *Edit distance*: $ED(r,s)$ is the least number of operations (i.e, character insertion, deletion, and substitution) to transform r into s.
- *Edit similarity*: $ES(r,s) = 1 - \frac{ED(r,s)}{max(|r|,|s|)}$.

Example 2. Consider the strings $r =$*"toen"* and $s =$*"token"*. Then, we have $ED(r,s) = 1$, obtained by inserting the character *"k"* into r, and $ES(r,s) = 1 - \frac{1}{5} = 0.8$.

2.3. Similarity Selection and Join

We are now ready to define similarity selection and join queries.

Definition 3 (Similarity Selection)**.** Given a table T with a string attribute $T.A$, a query string s, a similarity function Sim, and a threshold τ, retrieve all tuples $t \in T$ such that $Sim(t.A, s) \geq \tau$.

Definition 4 (Similarity Join)**.** Given two tables T_1 and T_2 with string attributes $T_1.A_i$ and $T_2.A_j$, respectively, a similarity function Sim, and a threshold τ, retrieve all pairs tuples $\langle t_1, t_2\rangle \in T_1 \times T_2$ such that $Sim(t_1.A_i, t_2.A_j) \geq \tau$.

The Similarity Selection and Join can be easily expressed in SQL using UDFs. For example, consider a UDF $Sim(r, s, \tau)$ implementing a similarity function and returning true if the similarity of the string arguments r and s according to the similarity function is not less than τ. Thus, given a query string s, a Similarity Selection can be declaratively realized as follows (the adaption of the query below to Similarity Join is obvious):

```
SELECT T.tid
FROM T
WHERE Sim(T.A, s, τ)
```

2.4. Filtering Techniques

A naïve approach to evaluating the similarity operations in the previous section would compute the similarity function for all input tuples. Obviously, the computational cost of this strategy is prohibitively high for large tables, particularly for joins where the time complexity would be quadratic in the size of the input tables. Next, we present filtering techniques that can substantially reduce the comparison space.

2.4.1. Overlap and Size Bounds

All similarity functions considered are closely related to the overlap measure between the token sets derived from the two input strings. For token-based similarity functions, such connection is straightforward: predicates involving these functions can often be equivalently represented in terms of an *overlap bound* [Chaudhuri et al. 2006]. Formally, the overlap bound between the token sets derived from strings r and s, denoted by $O(r, s)$, is a function that maps a threshold τ and the token set weights to a real value, s.t. $sim(r, s) \geq \tau$ iff $w(r \cap s) \geq O(r, s)$[1]. Now, a similarity join using token-based functions can be reduced to the problem of identifying all pairs r and s whose overlap of the corresponding token sets is not less than $O(r, s)$. For edit-based functions, we can employ overlap bound as a filter, since it can be shown that if $ED(r, s) \leq \tau$, then $Q_q(r) \cap Q_q(s) \geq O(r, s)$ [Gravano et al. 2001]. Table 1 shows the overlap bounds of the previous functions.

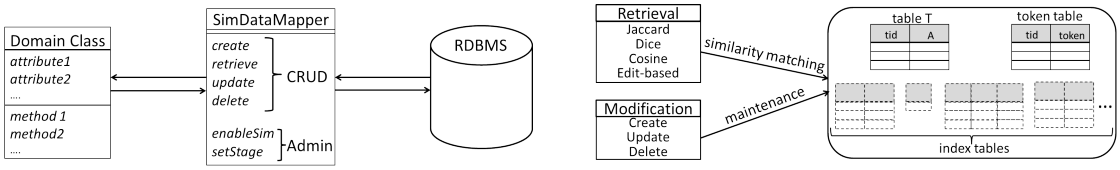
Further, similar strings have, in general, roughly similar sizes. (Here, we loosely use the term size to refer to both token set weights and string length; the former is used on token-based functions, whereas the latter on edit-based ones.) We can derive bounds for immediate pruning of candidate pairs whose sizes differ enough. Formally, the size bounds of r, denoted by $min(r)$ and $max(r)$, are functions that map τ and $w(r)$ or $|r|$ to a real value s.t. $\forall s$, if $sim(r, s) \geq \tau$, then $min(r) \leq w(s) \leq max(r)$ [Sarawagi and Kirpal 2004]. Thus, given a string r, we can safely ignore all strings whose sizes do not fall within the interval $[min(r), max(r)]$. Table 1 shows size bounds of all similarity functions considered in this paper.

2.4.2. Prefix Filtering

Focusing on the set overlap evaluation, we can prune a large share of the comparison space by exploiting the *prefix filtering principle* [Sarawagi and Kirpal 2004, Chaudhuri et al. 2006]. Prefixes allow selecting or discarding candidate pairs by examining only a fraction of the original sets. First, fix a global ordering $\mathcal{O}$; tokens of all sets are then sorted according to this ordering. A set $x' \subseteq x$ is a prefix of x if x' contains the first $|x'|$ tokens of x. Further, $pref_\beta(x)$ is the shortest prefix of x, the weights of whose tokens add up to more than β. It can be easily shown that for any two sets x and y, if $w(x \cap y) \geq \alpha$, then $pref_{\beta_x}(x) \cap pref_{\beta_y}(y) \neq \varnothing$, where $\beta_x = w(x) - \alpha$ and $\beta_y = w(y) - \alpha$, respectively [Chaudhuri et al. 2006].

By taking α equal to the overlap bound, we can use prefix filtering with the previous similarity functions. Note that prefix overlap is a condition necessary, but not sufficient to satisfy the original overlap constraint: an additional verification must be performed on the candidate pairs. Finally, the number of candidates can be reduced by using *document frequency ordering*, $\mathcal{O}_{df}$, as global token order to obtain sets ordered by increasing token frequency in the database. The idea is to minimize the number of sets agreeing on prefix elements and, in turn, candidate pairs by shifting lower frequency tokens to the prefix positions.

[1]For ease of notation, the parameter τ is omitted in the functions defined in this section.



(a) Pattern overview. (b) Similarity operations.

Figure 1. The SimDataMapper architectural pattern.

Example 3. Consider again the weighted sets $w(r)$ and $w(s)$ in Example 1 and assume that both sets are already sorted. For the Jaccard similarity and $\tau = 0.6$, we have $O(r,s) = 33.75$, $[min(r), max(r)] = [24, 66.7]$, $[min(s), max(s)] = [30, 83.3]$, $pref_{\beta_r}(r) = \langle$`c, 9`$\rangle$ and $pref_{\beta_s}(s)$=$\{\langle$`a, 10`$\rangle$,$\langle$`c, 9`$\rangle\}$.

3. The SimDataMapper Pattern

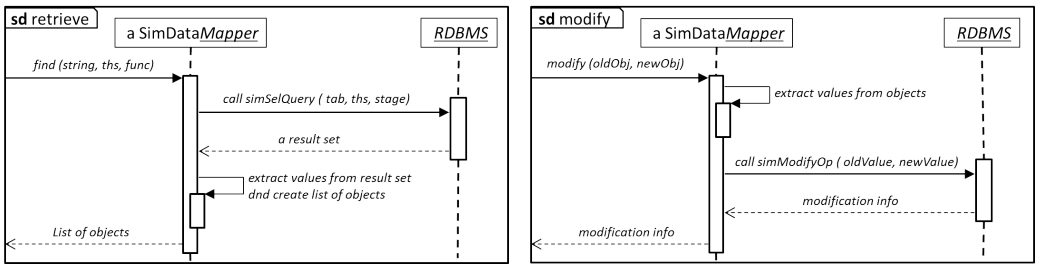
In this section, we present SimDataMapper, an architectural pattern to integrate the state-of-the-art similarity matching techniques previously described into database applications. We first present an overview of our approach before going into implementation details of the underlying solution scheme.

3.1. General Overview

The SimDataMapper pattern is essentially a specialization of the well-known *Data Mapper* pattern [Fowler 2003] and, as such, inherits its fundamental features. Basically, it is a layer of software that insulates in-memory domain objects and the database from each other and handles the data transfer between them. Figure 1(a) illustrates the SimDataMapper pattern and its interfaces for Create, Retrieve, Update, and Delete (CRUD) operations and administrative tasks, whereas Figure 1(b) zooms in on the similarity operations over original and auxiliary tables.

The method `enableSim` takes the specification of a string-valued attribute `T.A` of table `T` as input and create a *token table* based on the values in this attribute; also, a number of index tables are created depending on the selected *processing stage* (we explain the method `setStage` in the next section). Afterwards, one or more application objects whose corresponding class is mapped to `T` can be loaded by invoking a special `find` method, which takes a query string s, a threshold, and a similarity function identifier as input; such invocation results in a similarity selection query against the corresponding auxiliary tables (i.e., token and index tables). Figure 2(a) shows a sequence diagram capturing the behavior of Retrieve operations. Besides similarity selections, Retrieve also comprises similarity joins, in which case auxiliary tables for all attributes involved in the join condition must be available.

Further, Update operations affecting *T.A* as well as any Create and Delete operations translate into updates to the auxiliary tables. Figure 2(b) shows a general sequence diagram for those modification operations. In the `modify` method, either *oldObj* or *newObj* can be *null* for Create and Delete operations, respectively, or none of them for Update operations. Finally, SimDataMapper pattern acts as a regular Data Mapper for Retrieve operations based on exact matching or Update not involving any attribute considered for similarity matching.



(a) Sequence diagram for retrieving. (b) Sequence diagram for modification.

Figure 2. Sequence diagrams for CRUD operations based on similarity matching.

3.2. Implementation Details

The integration of similarity matching functionality into a database mapping layer can be realized in different ways. Stored procedures are a reasonable alternative in this context to avoid communication cost between the application and the RDBMS. In our implementation, procedural code is used to generate token sets from strings, populate auxiliary tables, and calculate the edit distance between two strings. All other operations are realized using standard, declarative SQL statements.

For simplicity, consider a simple table `T(tid, A)`, where `tid` is the key attribute of `T` and `A` is a string-valued attribute. Given `T.A` as input, the method `enableSim` creates the token table `TTokens(tid, token)`, where for each tuple t of `T`, the q-grams in $Q_q(t.A)$ are represented as a separated tuple associated with $t.tid$—the population of `TTokens` can be accelerated by using bulk loading functionality available in virtually all RDBMSs. The token table represents a "common ground" to the declarative realization of all similarity functions considered in Section 2.2. Further, the storage overhead of `TTokens`, denoted by $S(TTokens)$, is moderated: $S(TTokens) = n(q-1)(q+C) + (q+C)\sum_{j=1}^{n}|t_j.A|$, where n is the number of tuples in `T`, q the q-gram size, and C the size of `T.tid` [Gravano et al. 2001].

We can further identify a clear staged evaluation strategy for token-based similarity functions, which lends itself to the definition of index tables to improve query performance. Figure 3 illustrates several index tables organized into 4 stages—tables `T` and `TTokens` constitute the Stage 0. In this context, the tables at Stage 1 help to calculate token weights: `TDF(token, df)` (Figure 3(a)) associates each token to its *df* value, whereas `TSize(N)` (Figure 3(b)) is a dummy table storing the number of tuples of `T`—recall the IDF weight formula in Section 2.1. On the other extreme, we have table `TWeights(tid, token, tw, sw)` at Stage 4 (Figure 3(f)) , where `tw` and `sw` represent the token weight and the token set weight, respectively. While this index table greatly speeds up query processing, its maintenance can dramatically slow down update queries. In fact, `TWeights` would need to be recreated anew after any insertion or deletion on `T` because the weight of all tokens and token sets change with the number of tuples of `T`. To mitigate this problem, we follow the approach in [Koudas et al. 2007], where new terms independent of the number tuples of the original table were defined for the vector length—the similarity function under consideration in [Koudas et al. 2007] is the Cosine similarity in the vector space model. We adapt this strategy to our context by rewriting the token set weight formula as follows:

Stage 1

```
INSERT INTO TDF(token, df)
SELECT token, COUNT(*)
FROM TTokens
GROUP BY token
```

(a) Table storing *df* values for each token.

```
INSERT INTO TSize(N)
SELECT COUNT(*)
FROM T
```

(b) Table storing the number of tuples in T.

Stage 2

```
INSERT INTO TTokenDF(tid, token, df)
SELECT TT.tid, TT.token, TiDF.df
FROM TTokens TT, TDF
WHERE TT.token=TDF.token
```

(c) Table storing token and *df* values for each tuple.

```
INSERT INTO TRawSetWeight(tid, s1, s2)
SELECT tid, COUNT(*), SUM(LOG(df))
FROM TTokenDF
GROUP BY tid
```

(d) Table storing terms for set weight calculation.

Stage 3

```
INSERT INTO TSetWeight(tid, sw)
SELECT TR.tid, (LOG(TS.N)*TR.s1-TR.s2)
FROM TRawSetWeight TR, TSize TS
```

(e) Table storing set weight values.

Stage 4

```
INSERT INTO TWeights(tid, token, tw, sw)
SELECT TD.tid, TD.token, LOG(TS.N/TD.df), TW.sw
FROM TTokenDF TD, TSetWeight TW, TSize TS
WHERE TD.tid=TW.tid
```

(f) Table storing token and set weight values.

Figure 3. SQL statements to enable similarity matching on table `T`.

$$
\begin{aligned}
w(x) &= \sum_{tk \in x} w(tk) \\
&= \sum_{tk \in x} log(N) - log(df(tk)) \\
&= log(N) \cdot |x| - \sum_{tk \in x} log(df(tk)).
\end{aligned}
$$

We can now define Stage 2 containing two tables: `TTokenDF(tid, token, df)` (Figure 3(c)), which is simply the join result betwen `TTToken` and `TDF`, and `TRawSetWeight(tid, s1, s2)` (Figure 3(d)), where `s1` represents $|x|$ and `s2` represents $\sum_{tk \in x} log(df(tk))$ in the above formula. From Stage 2, we can easily define Stage 3 containing the table `TRawSetWeight(tid, sw)` (Figure 3(e)), which associates each `tid` with the corresponding token set weight. We end up with a variety of alternative processing stages: higher stages favor query processing, whereas lower stages favor updates. Applications can flexibly specify the best stage to the query-update ratio of an anticipated workload using the `setStage` method. Note, the storage overhead of the index tables in Stages 2–4 are within a constant factor of either $S(TTokens)$ (`TTokenDF` and `TWeights`) or the number of tuples in `T` (`TRawSetWeight` and `TSetWeight`).

Similarity selections are evaluated by first creating the table `SWeights(sid, token, tw, sw)` on-the-fly from the query string. `Sweights` is then matched against the auxiliary tables derived from table `T`. Figure 4(a) show the SQL query for Jaccard at Stage 2. Size bounds are checked in the `WHERE` clause and overlap bounds in the `HAVING` clause. Queries for Stages 3 and 4 are shown in Figures 4(b) and 4(c), respectively; the main difference from Stage 2 is the simplified calculation of token and set weights. Figure 4(d) shows the SQL query for edit distance. The only tables involved are the original tables — note that the query string is represented by table `S` — and the derived token tables. Similarly to Jaccard, size and overlap bounds are checked in the `WHERE` and `HAVING` clauses, respectively. The UDF ED is finally applied on the remaining string pairs that survived the filter steps. The adaption to the other similarity functions is straightforward: one basically needs to change the filters in the `WHERE` and `HAVING` clauses to the corresponding definitions shown in Table 1.

Similarity joins can be readily implemented using the same queries for selections. However, only the filters based on size and overlap bounds are insufficient to deal with the

```
SELECT   TD.tid, S.tid
FROM     TTokenDF TD, TRawSetWeight TR, TSize TS, SWeights S
WHERE    TD.token=S.token AND TD.tid=TR.tid AND
         (LOG(TS.N)*TR.s1-TR.s2)>=(S.sw*τ) AND
         (LOG(TS.N)*TR.s1-TR.s2)<=(S.sw/τ)
GROUP BY TD.tid, S.tid, TR.s1, TR.s2
HAVING   SUM(TS.N/TD.df)≥(τ/(1+τ))*((LOG(TS.N)*TR.s1-TR.s2)+S.sw)
```

(a) Query for Jaccard at Stage 2

```
SELECT   TD.tid, S.tid
FROM     TTokenDF TD, TSetWeight TW, TSize TS, SWeights S
WHERE    TD.token=S.token AND TD.tid=TW.tid AND
         TW.sw>=(S.sw*τ) AND TW.sw<=(S.sw /τ)
GROUP BY TD.tid, S.tid, TW.sw, S.sw
HAVING   SUM(TS.size/TD.df)>=(τ/(1+τ))*(TW.sw+S.sw)
```

(b) Query for Jaccard at Stage 3.

```
SELECT   T.tid, S.tid
FROM     TWeights T, SWeights S
WHERE    T.token = S.token AND T.sw>=(S.sw*τ) AND
         T.sw<=(S.sw/τ)
GROUP BY T.tid, S.tid, T.sw, S.sw
HAVING   SUM(T.tw)≥(τ/(1+τ))*(T.sw+S.sw)
```

(c) Query for Jaccard at Stage 4.

```
SELECT   T.tid, S.tid
FROM     T, TTokens TT, S, STokens ST
WHERE    TT.token=ST.token AND T.tid = TT.tid AND
         S.tid=ST.tid AND ABS(LEN(T.A)-LEN(S.A))<=τ
GROUP BY T.tid, T.A, S.tid, S.A
HAVING   COUNT(*)>=LEN(T.A)-1-(τ-1)*q AND
         COUNT(*)>=LEN(S.A)-1-(τ-1)*q AND ED(T.A, S.A, τ)
```

(d) Query for edit distance.

Figure 4. SQL statements for similarity selection.

Prefix generation

```
INSERT INTO TiPrefix (tid, token)
SELECT      tid, token
FROM        (SELECT tid, token, sw, (SUM(tw) OVER
             (PARTITION BY tid ORDER BY tw DESC, token)-tw) as p
             FROM TWeights) P
WHERE       p >= sw-(sw * τ)
```

(a) Table storing Jaccard prefixes.

```
INSERT INTO TiPrefix (tid, token)
SELECT      tid, token
FROM        (SELECT TT.tid as tid, TT.token token, (RANK() OVER
             (PARTITION BY TT.tid ORDER BY TD.df ASC, TT.token)) as p
             FROM TTokens TT, TDF WHERE TT.tid = TFD.tid) as P
WHERE       p <= τ*q + 1
```

(b) Table storing edit distance prefixes.

Candidate generation

```
CREATE VIEW     CAND AS
SELECT DISTINCT P1.tid, P2.tid
FROM            T1Prefix P1, T2Prefix P2
WHERE           P1.token=P2.token
```

(c) View for candidate generation.

Combination table generation

```
INSERT INTO T1T2DF(tid, token)
SELECT      D.tid, SUM(D.df)
FROM        ((SELECT * FROM T1DF)
            UNION ALL
            (SELECT * FROM T2DF)) as D
GROUP BY    D.token
```

(d) Table storing *df* values combined from two input tables.

Figure 5. SQL statements for similarity join.

high cost of joining two large tables. Thus, a more aggressive pruning approach based on prefix filtering is needed. To this end, we first create *prefix tables* from each input, which store only tokens in the prefix of each token set. The creation of prefix tables can be implemented in pure SQL using *window functions*. The SQL statements to create prefix tables based on weighted sets, for Jaccard, and unweighted sets, for the edit distance, are shown in Figures 5(a) and 5(b), respectively. Previous implementations of prefix filtering in the RDBMS were either procedural [Chaudhuri et al. 2006] or did not consider token weights [Xiao et al. 2011, Augsten et al. 2014].

After having created the prefix tables, pairs of tuples whose associated prefixes have at least a token in common can be identified by a simple equi-join; those pairs constitute the candidate set which is further processed by the queries discussed previously. This operation is illustrated by a simple view definition in Figure 5(c). Finally, except for self-joins, we create new tables storing the summation of the *df* value of tokens appearing in the two input tables as shown in Figure 5(d); also the value of N used in the IDF calculation should be given by the sum of the sizes of the tables (not shown in figure). These summation tables are deleted upon completion of the join operation, together with the prefix tables. Alternatively, if there is no intervening data update, prefix tables can be reused on subsequent similarity join evaluations with the same or lower threshold values.

4. Experiments

We now report some performance experiments. We used a real world dataset containing medicine names, which were obtained from a production medical database.[2] The input table contains 20K tuples; the values of the string attribute have an average length of 10 characters and a large number of tuples have similar content referring to a same medicine owing to naming variations and misspellings. We then doubled the table size to 40K tuples by producing a copy of each string, applying 1–3 character-level transformations (insertion, deletion, and substitution) to the copy and inserting it into the table. A large number of similar strings results in an increased output size as well as execution time.

All experiments were run on a workstation with Intel i7 1,6 GHz, 6MB CPU cache, HD Sata 5400 rpm, and 8 GB of main memory. We implemented SimDataMapper using Java JDK 7 (Oracle) 1.7.0 and JDBC Driver version 4. We used a vanilla installation of PostgreSQL 9.3 as RDBMS platform, without any configuration adjustments and physical design tuning such as clustered indexes. Therefore, the results reported here can be seen as a lower bound of the achievable database system performance. We report the RDBMS processing cost measured in average runtime over repeated executions; the time required to process application objects was not included in the measurements (of course, this time can significantly vary along different applications). We report results for Jaccard and *ED*; results for the other similarity functions followed similar trends. We fixed a threshold of 0.75 for Jaccard, a distance of 1 for *ED*, and generated q-grams of size 3.

Figure 6(a) shows the results for similarity selections; for Jaccard, Stages 2–4 are considered. All queries took less than 7 seconds to complete. As expected, performance increases with the processing stage. Owing to the higher cost of the edit distance calculation, *ED* is only faster than Jaccard when the latter runs at Stage 2. For updates (Figure 6(b)), the results are just the opposite and now performance decreases at higher stages. Timings for updates are slightly higher as compared to selections, which indicates that there is still room for improvement in our current implementation.

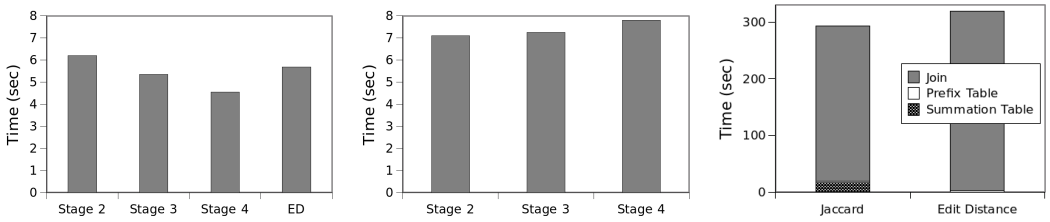
Finally, Figure 6(c) shows the results for similarity joins. For this experiment, we generated another dataset with 40K "dirty" copies. The time for the actual join largely dominates the overall execution time, thereby rendering the cost of construction of the temporary index and prefix tables negligible. Results for Jaccard are about 15% faster than those for ED. Besides the cost of the edit distance, prefixes based on unweighted sets normally exhibit worse filtering effectiveness as compared to weighted ones [Ribeiro and Härder 2011].

5. Related Work

Many different approaches have been proposed for string similarity matching, which fall into three broad, mostly complementary, categories: 1) *stand-alone algorithms*; 2) *integration with query engines*; and 3) *declarative realization on top of query engines*.

There is a large body of research on stand-alone, specialized algorithms for similarity selection and join [Sarawagi and Kirpal 2004, Xiao et al. 2011,

[2]We also evaluated our approach on other real datasets, e.g., actor names from the IMDB database (imdb.com) and obtained similar results.



(a) Selection query timings. (b) Update timings. (c) Join query timings.

Figure 6. Query and update timings for Jaccard and Edit Distance.

Ribeiro and Härder 2011]. Numerous techniques have been proposed, including prefix- and size-based filters, specialized data structures based on inverted lists, and many others.

Several proposals to integrate string similarity matching into query engines focus on the design of new physical operators. The SSJoin operator is based on composing regular and similarity-specific operators into query evaluation plans [Chaudhuri et al. 2006]; SSJoin supports the same class of similarity functions considered in this paper. The work in [Augsten et al. 2014] defines a tokenizer operator to allow evaluation of similarity queries without precomputed tokens and indexes– the idea of using a tokenizer operator to enable on-the-fly evaluation first appeared in the context of XML DBMSs [Ribeiro and Härder 2007]. Tight coupling of such approaches with full-fledged RDBMSs requires adaptions on all query optimizer components, including algebraic operators, cost models, cardinality estimates, and index infrastructure. Thus, integration of similarity processing into general-purpose RDBMSs remains a long-term research goal.

Declarative realization of similarity matching through SQL statements permits readily use of existing RDBMS machinery. Our work builds upon previous efforts in this area [Gravano et al. 2001, Gravano et al. 2003, Koudas et al. 2007]. But, instead of considering each similarity predicate in isolation, we focus on identifying key data structures and operations and assembling them into an architectural pattern to provide common support to a variety of similarity predicates.

In some sense, our work can be classified into a new category of approaches to string similarity matching: 4) *integration with object-oriented applications*. Many popular products offer a data mapping layer to bridge object-oriented applications and relational databases, such as Hibernate and ADO.Net. Declarative similarity matching is not directly supported by these solutions. Frameworks such as Hibernate Search provide similarity matching capability to object domain models via integration with full-text search engines. In such approaches, data resides in two different and independent systems, i.e., an RDBMS and a search engine, and the corresponding data models need to be "glued together" somehow at the application level. In contrast, our solution requires no support of specialized systems.

Finally, there is a long and fruitful line of research on similarity operations over image and other complex data types (e.g., see [Bedo et al. 2014] and references therein). Most techniques developed in this area are quite different from the ones considered here for string data. Integration of similarity matching techniques for image and string data into a single framework is an interesting direction for future research.

6. Conclusion

This paper addressed the problem of bridging database applications and string similarity operations on RDBMSs, which has received very little attention in the existing literature. Existing solutions require creation and maintenance of several auxiliary tables, severely complicating application persistence. We proposed the SimDataMapper, an architectural pattern that isolates in-memory domain objects from details of the underlying similarity processing in the RDBMS. An efficient, flexible, and reusable implementation of SimDataMapper was presented, which is largely based on standard SQL, incorporates state-of-the-art optimization techniques, and supports a rich variety of similarity functions.

Acknowledgments We thank the anonymous reviewers for their helpful comments. This research was partially supported by FAPEMIG, FAPESC, CNPq, and INCT INCoD.

References

Augsten, N., Miraglia, A., Neumann, T., and Kemper, A. (2014). On-the-fly token similarity joins in relational databases. In *SIGMOD*, pages 1495–1506.

Bedo, M. V. N., Traina, A. J. M., and Jr., C. T. (2014). Seamless integration of distance functions and feature vectors for similarity-queries processing. *JIDM*, 5(3):308–320.

Chaudhuri, S., Ganti, V., and Kaushik, R. (2006). A primitive operator for similarity joins in data cleaning. In *ICDE*, page 5.

Cohen, W. W., Ravikumar, P. D., and Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. In *IIWeb*, pages 73–78.

Cook, W. R. and Ibrahim, A. H. Integrating programming languages & databases: What's the problem? ODBMS.ORG, Expert Article.

Fowler, M. (2003). *Pattern of Enterprise Application Architecture*. Addison-Wesley.

Gravano, L., Ipeirotis, P. G., Jagadish, H. V., and et al. (2001). Approximate string joins in a database (almost) for free. In *VLDB*, pages 491–500.

Gravano, L., Ipeirotis, P. G., Koudas, N., and Srivastava, D. (2003). Text joins in an rdbms for web data integration. In *WWW*, pages 90–101.

Koudas, N., Marathe, A., and Srivastava, D. (2007). Propagating updates in spider. In *ICDE*, pages 1146–1153.

Navarro, G. (2001). A guided tour to approximate string matching. *CSUR*, 33(1):31–88.

Ribeiro, L. A. and Härder, T. (2007). Embedding similarity joins into native xml databases. In *SBBD*, pages 285–299.

Ribeiro, L. A. and Härder, T. (2011). Generalizing prefix filtering to improve set similarity joins. *Information Systems*, 36(1):62–78.

Sarawagi, S. and Kirpal, A. (2004). Efficient set joins on similarity predicates. In *SIGMOD*, pages 743–754.

Xiao, C., Wang, W., Lin, X., Yu, J. X., and Wang, G. (2011). Efficient similarity joins for near-duplicate detection. *TODS*, 36(3):15.

sbbd:11

Uma Estratégia não Supervisionada para Previsão de Eventos usando Redes Sociais

Derick M. de Oliveira, Roberto C.S.N.P. Souza, Denise E.F. de Brito, Wagner Meira Jr., Gisele L. Pappa

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte – MG – Brasil

{derickmath,nalon,denise.brit,meira,glpappa}@dcc.ufmg.br

Resumo. *Desde a popularização de mídias sociais baseadas em texto, vários estudos têm sido conduzidos utilizando dados destas plataformas para prever eventos do mundo real. A abordagem tradicional é considerar usuários postando em mídias sociais como sensores e as respectivas mensagens sobre um evento como um indicador da sua ocorrência ou intensidade. Em geral, uma fase de análise de sentimentos é realizada para assinalar mensagens a um conjunto pré-definido de categorias. Neste caso, rotulação manual de dados é necessária para treinar um classificador. Entretanto, tal rotulação pode ser muito custosa, demandar muito tempo e está sujeita a erros humanos. Mais ainda, algumas vezes é difícil distinguir qual categoria pré-definida é a mais apropriada para uma dada mensagem. Neste sentido, propomos uma metodologia não supervisionada para prever eventos baseados em dados de mídia social. A metodologia proposta foi aplicada a metodologia proposta a dados associados a dengue no Brasil e mostramos como tal metodologia pode melhorar o desempenho na maior parte dos casos.*

Abstract. *Since the popularization of text-based social media, many studies have been conducted using data from these platforms to predict real-world events. The common approach is to consider people posting to a social media as sensors and their respective messages about an event as an indicator of its occurrence or intensity. In general, a sentiment analysis step is performed in order to assign messages to a set of predefined categories. In this case, manually labeled data is required to train a classifier. However, manually labeling messages can be costly, time consuming and is subject to human error. In addition, sometimes it is difficult to distinguish which predefined category is the most appropriate for an specific message. In this sense, we propose an unsupervised methodology to handle social media data for event prediction. We apply our methodology to dengue-related data from Brazil and show how an unsupervised approach can significantly improve the event prediction performance in the majority of the cases.*

1. Introdução

Desde o surgimento das redes sociais *online*, um grande número de estudos vêm sendo conduzidos com a utilização de dados coletados a partir dessas plataformas para previsão de eventos reais [Sakaki et al. 2010, Chakrabarti and Punera 2011]. Uma abordagem comum nesses estudos é considerar os usuários como sensores, e as mensagens

postadas por eles como indicadores da ocorrência e intensidade desses eventos. Diversas aplicações atestam a capacidade das redes sociais *online* em fornecer informações úteis para a previsão de eventos reais. Por exemplo, este tipo de dado foi utilizado para antecipar a ocorrência de fenômenos naturais [Sakaki et al. 2010], monitorar epidemias de gripe [Cullota 2010] e lances importantes durante eventos esportivos [Chakrabarti and Punera 2011].

Ao instanciar um modelo para previsão do evento real, o peso atribuído a cada mensagem, independente de seu conteúdo ou localização geográfica, é normalmente o mesmo. Porém, dependendo do tipo de evento sendo monitorado e sua área de abrangência, a relevância de diferentes tipos de mensagem pode variar para o modelo de previsão. Por exemplo, no caso do monitoramento de epidemias de dengue, mensagens falando sobre experiências pessoais dos usuários com a doença podem ser muito mais indicativas de epidemias que mensagens de campanhas de conscientização, como mostrado em [Souza et al. 2014]. Da mesma forma, hábitos culturais de diferentes regiões podem fazer com que o vocabulário utilizado para discriminar mensagens relevantes varie de uma região para outra.

Em eventos com uma ampla abrangência geográfica, normalmente assume-se que locais próximos espacialmente são mais similares, entretanto, essa premissa pode se mostrar inadequada por uma série de características culturais, sócio-econômicas e tecnológicas. A consequência é que cidades demograficamente semelhantes podem se mostrar fundamentalmente diferentes sob a perspectiva da apropriação e uso das redes sociais, dificultando a criação de modelos globais que sejam efetivos.

Em geral, durante esse processo de previsão de eventos, uma etapa de análise de sentimentos é realizada [Chew and Eysenback 2010, Souza et al. 2014]. Nesta etapa, as mensagens coletadas são mapeadas para um conjunto pré-definido de categorias, de acordo com o sentimento expresso pelo conteúdo dos textos. Assim, dados rotulados manualmente são necessários para treinar um classificador. Contudo, rotular manualmente um conjunto de dados demanda tempo e está sujeito a erro humano. Além disso, as categorias pré-definidas podem ser semanticamente muito próximas, dificultando a distinção entre elas. Mesmo definindo um conjunto amplo de categorias, a dinamicidade do cenário de redes socias *online* leva a alguns problemas tais como o fato das categorias definidas não contemplarem plenamente o conteúdo das mensagens e a necessidade de atualização constante do conjunto de treinamento para lidar com essa dinamicidade e mudanças de conceito. Em particular, no caso de eventos que atingem uma grande área geográfica, como no caso de eventos relacionados a vigilância de saúde, essas categorias pré-definidas podem muitas vezes não refletir o comportamento dos usuários considerando as diferenças geográficas e culturais. Por exemplo, a população de determinada cidade pode ser mais comprometida a discutir assuntos relacionados a determinada enfermidade em uma perspectiva pessoal, ao passo que em outro local pode ser mais comum uma mobilização pela conscientização e informação da população.

Por fim, todos esses aspectos mencionados estão alinhados a uma característica marcante desse cenário de redes sociais *online* que é o ruído inerente. Esse ruído é fruto de diversos fatores associados ao contexto, tais como a ambiguidade proporcionada pela língua, o vocabulário que se altera de forma dinâmica e até outros fatores como ironia/sarcasmo.

Neste trabalho, é proposta uma metodologia não supervisionada para prever eventos do mundo real a partir de dados de redes sociais *online*. A metodologia constrói um modelo de previsão por localidade com base nas mensagens associadas à mesma. Essas mensagens são segmentadas em grupos coesos em relação ao seu tópico e a série temporal do número de mensagens por tópico é então determinada. As séries temporais relacionadas aos vários tópicos são então utilizadas para construir o modelo de previsão.

Essa metodologia possui algumas características que a tornam adequada para lidar com o problema de previsão de eventos a partir de dados de redes sociais. Em primeiro lugar, não é utilizado um conjunto de categorias pré-definidas para todas as cidades, mas tópicos semanticamente coesos. Os tópicos são determinados por cidade através de uma abordagem não supervisionada que descarta automaticamente dados ruidosos e considera as peculiaridades de cada cidade, como manifestado pelos tópicos. A dinamicidade inerente às redes sociais é naturalmente contemplada pela consideração tardia da dimensão temporal, o que ocorre apenas após a determinação dos tópicos. Como demonstrado nos nossos resultados, essa metodologia é efetiva e proveu bons resultados superando técnicas baseadas em análise de sentimentos.

2. Trabalhos relacionados

Diversos trabalhos recentes têm se utilizado de dados obtidos a partir de redes sociais *online* para prever eventos do mundo real. De fato, a quantidade de conteúdo gerado pelos usuários nesses canais possibilita obter informações valiosas e em tempo real sobre uma grande variedade de assuntos.

A maior parte dos trabalhos nesse contexto se diferencia em alguns aspectos, tais como: rede social utilizada, por exemplo, Twitter, Facebook, entre outras; a granularidade temporal adotada, em que semanas e meses são os mais comuns; e o modelo empregado para previsão. Contudo, algumas características permeiam diversos trabalhos na área de detecção de eventos. Uma estratégia comum em grande parte dos trabalhos é considerar que as mensagens coletadas possuem o mesmo peso na previsão de eventos. Em [Cullota 2010] dados coletados do Twitter são utilizados para monitorar incidência de gripe. Após coletar os dados, uma fase de análise de sentimento é empregada para separar as mensagens que estão relacionadas diretamente com surtos da doença. De posse disso, são aplicados modelos de regressão para estimar a incidência da doença a partir do volume de mensagens. Uma abordagem similar é proposta em [Lampos and Cristianini 2010] para monitorar epidemias, ou surtos do vírus H1N1 no Reino Unido, a partir de mensagens relatando sintomas, e em [Santos and Matos 2014] em que mensagens com conteúdo relacionado à gripe e sintomas da doença são utilizadas para prever a incidência da doença em Portugal.

No contexto de vigilância de saúde, estudos anteriores propuseram um conjunto de características para acomodar o comportamento dos usuários nas redes sociais [Chew and Eysenback 2010]. Baseado nessa estratégia, [Gomide et al. 2011] propôs uma abordagem para monitorar epidemias de dengue. A estratégia consiste em classificar as mensagens coletadas de acordo com o sentimento expresso pelo conteúdo. A partir dessa classificação, somente as mensagens expressando experiência pessoal com a doença são usadas como indicativo da ocorrência de surtos. Contudo, [Souza et al. 2014] mostrou que, dependendo do contexto analisado ponderar adequadamente mensagens que expres-

sam outros sentimentos, como sarcasmo ou opinião, pode ampliar a capacidade de análise dos modelos.

Outra característica comum na maior parte dos trabalhos é realizar a análise em um escala global, no sentido de que essa análise é feita em uma granularidade espacial grossa. Para isso, porções regionais são agregadas em regiões maiores de acordo com a referência geográfica. Essa estratégia é adotada em diversos trabalhos mencionados anteriormente. Por exemplo, em [Lampos and Cristianini 2010] os dados coletados são agregados por nível regional, ao passo que em [Cullota 2010] e [Santos and Matos 2014] esse dados são utilizados em uma perspectiva que considera todo o país.

Essa estratégia de agregar localidades geográficas em porções maiores favorece em alguns contextos, como no caso de eventos cuja quantidade de mensagens no grão fino inviabiliza uma previsão acurada. Contudo, não leva em consideração diferentes hábitos e costumes locais. De forma recíproca, considerar que todas as mensagens possuem o mesmo peso pode não ser a melhor estratégia. Cada unidade geográfica possui um vocabulário diferente e um nível de percepção dos eventos que varia. Assim, considerar essas diferenças ao propor uma estratégia de previsão de eventos a partir do conteúdo gerado pelos usuários em redes sociais pode fornecer um caminho mais adequado para a comprensão dos fatores associados.

3. Metodologia proposta

Esta seção descreve a metodologia proposta neste trabalho, que é dividida em 4 etapas, como a seguir: (i) atribuição da localização, em que os dados coletados a partir das redes sociais *online* são atribuídos a uma localização válida; (ii) agrupamento de mensagens de uma localização em conjuntos que possuem a mesma semântica sobre determinado assunto; (iii) determinação da base temporal, quantificando o número de mensagens por tópico e localidade ao longo do tempo; e (iv) previsão de eventos de interesse.

Atribuição da localização A fase de atribuição da localização é uma parte crucial da metodologia proposta. Uma das condições colocadas por [Sakaki et al. 2010] para que seja possível realizar a previsão de eventos do mundo real a partir de dados de plataformas sociais é que o número de pessoas em contato com o evento seja suficiente e que esse evento influencie a vida de uma grande quantidade de pessoas, para que elas falem sobre ele. Contudo, é fundamental considerar também as diferenças de comportamento dos usuários de redes sociais em face à experiência com determinados eventos. Essas diferenças podem estar associadas, por exemplo, a localidade geográfica e dimensões culturais dos usuários. Dependendo do tipo de evento de interesse, determinada população pode estar mais disposta a se manifestar sobre o evento em um âmbito pessoal do que outra. Essa variabilidade impede que quaisquer estratégias sejam aplicadas de forma uniforme aos dados de todas localidades. Levar essas peculiaridades em consideração é fundamental para obter previsões mais acuradas.

Agrupamento das mensagens O objetivo da segunda etapa da metodologia proposta é separar as mensagens coletadas automaticamente em grupos que, de certa forma, estão relacionados ao mesmo assunto. Para isso, pode-se adotar um estratégia baseada em algoritmos para detecção de tópicos latentes, tal como Alocação Latente de Dirichlet (LDA)

[Blei et al. 2003] ou ainda abordagens baseadas em classificação ou agrupamento a partir dos textos das mensagens. Uma desvantagem no caso da aplicação de algoritmos como LDA para detecção de tópicos em mensagens coletadas a partir de redes sociais é que o número de tópicos a ser detectado deve ser previamente definido, o que depende das características dos dados de cada localização. Em um universo de discussão ampla como essas plataformas, essa pode ser uma tarefa árdua. No caso da classificação dos textos, demanda-se um conjunto de dados previamente rotulado para treinar o classificador. Nesse caso, embora alguns estudos apontem algumas categorias como sendo comuns a vários cenários [Chew and Eysenback 2010], usar esse conjunto de categorias pré-definidas tende a não captar plenamente o comportamento de usuários nas redes sociais.

Assim, neste trabalho, a estratégia adotada consiste em agrupar as mensagens coletadas através de um algoritmo de agrupamento de texto. Esse agrupamento é realizado usando o algoritmo *Density Based Spatial Clustering of Applications with Noise* (DBSCAN) [Ester et al. 1996] e emprega como medida de similaridade a distância do Coseno. Um aspecto primordial desse algoritmo, é que ele se baseia na densidade dos pontos presentes no conjunto de dados para definir os grupos, ao invés de considerar simplesmente a distância entre esses pontos, como no caso do *K-means*. Dessa forma, o DBSCAN é capaz de encontrar, entre os dados, grupos de formato arbitrário, incluindo grupos cujo formato é não-convexo. Para tanto, dois parâmetros principais são requeridos: Um valor ε que representa o tamanho da vizinhança considerada, e o número mínimo de pontos nessa vizinhança *minpts*. Cada ponto da base de dados é então definido como: ponto central, se existem pelo menos *minpts* pontos em sua ε-vizinhança; ponto de fronteira se o número de pontos em sua ε-vizinhança não atinge *minpts*, mas ele se encontra na ε-vizinhança de um ponto central; ponto de ruído quando não está em nenhum dos casos anteriores. Um aspecto interessante do DBSCAN então é que ele descarta determinados pontos que são considerados ruidosos durante a fase de agrupamento. Em síntese, é adotado o DBSCAN pela sua robustez e adaptabilidade em termos dos formatos dos agrupamentos.

Determinação da base temporal Na terceira etapa da metodologia cada um dos grupos é considerado como uma variável preditora para o volume do evento de interesse. No contexto de vigilância de saúde, por exemplo, foi demonstrado que, para algumas doenças, há uma forte correlação entre a incidência da doença e o volume de mensagens relacionados ao evento. Assim, utiliza-se a quantidade de mensagens relacionadas ao evento para estimar a sua intensidade, mais especificamente a quantidade de mensagens por tópico ao longo do tempo. Cabe lembrar que os grupos são determinados sem considerar quando as mensagens são postadas, mas para determinar as séries temporais utiliza-se essa informação. Num primeiro momento, é determinada a granulação temporal a ser utilizada, que é semanal no contexto deste trabalho. Assim, cada grupo e semana do período avaliado tem um contador e cada mensagem incrementa o contador do grupo e semana correspondente. Ao final, tem-se uma série temporal de mensagens para cada um dos grupos.

Previsão de eventos de interesse A última etapa da metodologia proposta tem como objetivo avaliar a capacidade de previsão de eventos a partir das bases de dados temporais

construídas na etapa anterior.

De posse dessa série temporal de cada grupo e da série temporal verdadeira do evento, tal como relatórios de vigilância de saúde divulgados por secretarias de saúde, realiza-se a previsão do evento de interesse. Apesar do método de agrupamento selecionado realizar a remoção de pontos considerados ruídos, as séries geradas ainda assim estão sujeitas a pontos ruidosos. Além disso, essas informações estão correlacionadas no tempo. Dessa forma, é preciso um modelo de previsão que seja capaz de lidar com essas peculiaridades. Na metodologia proposta nesse trabalho, essa previsão é feita usando o filtro de Kalman (FK)[Kalman 1960].

O FK é um modelo de previsão que usa dados passados para corrigir seu atual estado. Ele é regido por um conjunto de equações lineares que soluciona o problema de forma recursiva através da minimização do erro quadrático. A cada etapa o valor verdadeiro é estimado e obtém-se a variação da medição. O FK pode ser dividido em duas etapas principais: *atualização temporal* e *atualização de medida*. Na etapa de atualização temporal, os dados passados são utilizados para prever o valor do estado atual e calcular o ruído dessa medição. Na etapa de atualização da medida, calcula-se o chamado ganho de Kalman e junto com o valor observado pelos sensores, nesse caso a contagem de mensagens, ajusta-se o valor obtido na etapa anterior.

Uma fase importante para o uso do FK é adaptar as matrizes do sistema de equações para o contexto da aplicação. No caso de eventos relacionados a vigilância de saúde, por exemplo, é comum que o número de mensagens seja muito menor que o número real de casos. Em outras palavras, a medição dos sensores se encontra em uma escala diferente dos valores reais a serem estimados. Esse problema é contornado introduzindo uma matriz de medição. Essa matriz é responsável pela forma como a escala dos sensores se adaptam à escala do fenômeno real.

A figura 1 sumariza a metodologia descrita anteriormente.

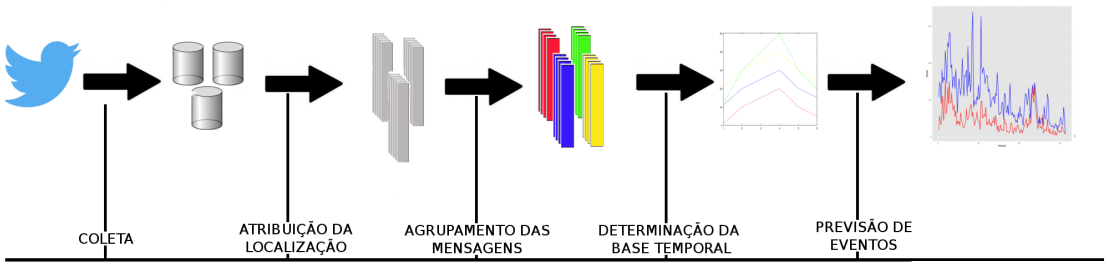


Figura 1. Metodologia proposta.

4. Resultados experimentais

Esta seção descreve a aplicação da metodologia proposta para detecção de epidemias de dengue em cidades brasileiras, a partir de mensagens postadas no Twitter.

4.1. Coleta e pré-processamento dos dados

Para coletar as mensagens postadas no Twitter, que estejam relacionadas a ocorrência de dengue, foi utilizada a API do Twitter, definindo-se como palavras-chave os termos

"dengue" e "Aedes". A coleta foi realizada no período de Janeiro de 2011 até Dezembro de 2013. Inicialmente, os dados coletados passaram por uma fase de pré-processamento. Nesta fase foram filtrados os acentos e as URL's presentes nos textos das mensagens. Em seguida foram geradas bi-gramas a partir da junção de palavras adjacentes no texto. Além disso, foram filtradas as *stopwords* bem como as bi-gramas compostas pela junção de duas *stopwords*.

4.2. Atribuição da localização

Como descrito na seção 3, a atribuição da localização é uma fase fundamental da metodologia proposta. Em particular, no caso da vigilância de epidemias de dengue no Brasil, essa atribuição da localização tem um papel ainda mais proeminente. A definição de políticas públicas para controle de epidemias de dengue é de responsabilidade de cada município. Assim, um requisito nesse caso é que a análise seja feita em nível municipal.

A API do Twitter fornece a localização geográfica do usuário, quando informada, de três formas possíveis: (i) uma localização sugerida ao usuário baseada no seu endereço IP; (ii) um campo de texto que é preenchido pelo usuário, podendo conter um conjunto qualquer de termos; (iii) coordenadas geográficas, no caso de usuários que postam mensagens atráves de dispositivos móveis. Para atribuir as mensagens a uma localização válida, ou seja, uma cidade brasileira, o primeiro filtro consiste em remover todas as mensagens que não apresentam "Brasil" nos campos de localização. Assim, grande parte das mensagens postadas de locais fora do Brasil são eliminadas. Verificou-se que essa estratégia não gera perda de mensagens postadas a partir de cidades brasileiras e sua adoção torna parte do processamento mais eficiente. A seguir toda informação de localização disponível é utilizada de forma exaustiva durante uma busca por nomes consistentes e não ambíguos de cidades brasileiras e seus respectivos estados. O nome da cidade é sempre associado ao estado para evitar inconsistências no caso de cidades com mesmo nome que estão em estados diferentes. Nesta etapa foram consideradas todas as cidades cuja população é superior a 100 mil habitantes, totalizando 298 cidades. Todas as mensagens cuja localização não pode ser atribuída ao final desse processo são então descartadas.

4.3. Agrupamento das mensagens

Após a etapa de atribuição de localização tem-se, para cada uma das cidades com mais de 100 mil habitantes, uma parcela significativa (e completa dentro do universo de mensagens) das mensagens que de alguma forma estão relacionadas a ocorrência de dengue naquela cidade. O próximo passo então é realizar o agrupamento dessas mensagens usando o DBSCAN. Para isso, utilizou-se a ferramenta Scikit-learn [Pedregosa et al. 2011]. Essa ferramenta de código aberto permite a criação de prótotipos de algoritmos para mineração e análise de dados de forma simples e eficiente.

Antes de realizar o agrupamento das mensagens, cada conjunto de mensagens de determinada cidade precisa ser representado de forma a refletir a importância de cada palavra dentro de cada mensagem. Para isso, os textos das mensagens são representados utilizando a métrica *term frequency-inverse document frequency* (Tf-idf). A matriz Tf-idf obtida é usada para realizar o agrupamento das mensagens. A métrica adotada para representar a distância entre dois pontos da base de dados foi a distância de cossenos. Além disso, para o DBSCAN foram definidos os parâmetros de vizinhança $\epsilon = 0.4$ e número

mínimo de pontos $minpts = 10$. Devido a restrições espaço, as 298 cidades consideradas na fase de localização das mensagens foram ordenadas de acordo com o número total de mensagens atribuído a cada uma delas. Assim, selecionou-se uma variedade de cidades de acordo com a quantidade de mensagens relacionadas a dengue para realizar o agrupamento.

Ao final da execução, tem-se, para cada cidade, um determinado número de grupos que é desconhecido inicialmente. Diferente de uma abordagem supervisionada, em que o número de classes é definido *a priori*, o agrupamento busca grupos que sejam mais coesos e que, portanto, podem levar a uma compreensão melhor dos fatores envolvidos com o evento. No caso da vigilância de dengue, uma enormidade de fatores podem levar as pessoas a postarem mensagens sobre o assunto. No caso da abordagem supervisionada, as categorias pré-estabelecidas são as únicas opções de atribuição de classe para determinada mensagem e portanto podem não ser suficientes para acomodar o comportamento dos usuários.

A figura 2 apresenta um exemplo que ilustra tipicamente essa situação. Na primeira semana de janeiro de 2012, o ex-jogador de futebol Ronaldo "fenômeno" Luís Nazário de Lima foi ao hospital com suspeita de dengue. O próprio Ronaldo usou sua conta no Twitter para se manifestar sobre o ocorrido. Imediatamente, diversas pessoas passaram a *retweetar* o jogador sob as mais diversas óticas: desejando melhoras, fazendo piadas ou simplesmente repassando a informação. Da mesma forma jornais esportivos reproduziram a informação sobre o caso de dengue do ex-jogador. A figura 2 apresenta uma nuvem de palavras construída com o conteúdos das mensagens de agrupamentos que mencionam o caso do ex-jogador. Esse exemplo mostra como o agrupamento pode ser mais eficiente na tarefa de detectar uma elevação no número de postagens que não estão diretamente relacionadas com aumento no número de pessoas com a doença. No caso da abordagem supervisionada, todas essas mensagens seriam classificadas em algumas das categorias disponíveis e não seria possível detectar esse fator.

Figura 2. Nuvem de palavras construída com textos de tweets sobre o caso de dengue do ex-jogador Ronaldo.

4.4. Determinação da base temporal

A partir do exemplo mencionado anteriormente, uma abordagem para solucionar esse problema, que surge imediatamente seria remover da base de dados as mensagens que são *retweets*. Contudo, para alguns eventos, como no caso das epidemias de dengue, informações de retweets também podem ser valiosas para o processo de previsão. Em muitos casos, pessoas repassam mensagens relacionadas a informação e conscientização sobre a doença quando o ambiente em que elas se encontram está de alguma forma afetado pela epidemia [Souza et al. 2014]. Dessa forma, simplesmente excluir todas essas mensagens leva a uma perda de informação. Dessa forma, a determinação da base temporal

das mensagens pode ser fundamental nesse processo. Define-se inicialmente a granulação temporal, que no caso deste trabalho é semanal. A seguir, basta estabelecer um contador utilizando a data de postagens das mensagens para construir a série temporal de cada grupo de mensagens obtido na etapa de agrupamento. Assim, para cada cidade, obtém-se uma série temporal por grupo de mensagens, como na figura 3

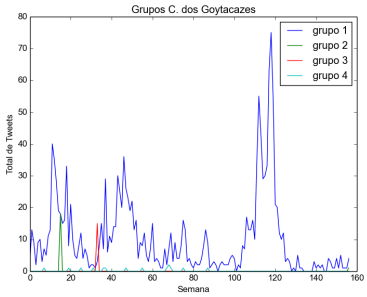


Figura 3. Série temporal por grupo de mensagens para a cidade Campos dos Goytacazes

A partir da característica de cada uma das séries é possível detectar grupos que são ou não propícios para a previsão de epidemias, independente do fato das mensagens serem retweets. A figura 4 ilustra esse comportamento. A figura da esquerda consiste na série temporal do agrupamento relacionado ao caso do ex-jogador Ronaldo, mencionado anteriomente. Observa-se claramente que todas as mensagens foram postados em um período específico de tempo, que coincide com a data da postagem do ex-jogador. Esse grupo de mensagens não é de fato efetivo para a previsão, principalmente quando comparado com a série real de casos de dengue apresentada na mesma figura. Por outro lado, a figura da esquerda consiste na série temporal construída a partir de um grupo de mensagens na cidade do Rio de Janeiro que estão relacionadas com a mesma piada. Observa-se que essa piada foi retweetada diversas vezes durante todo o período de análise. Em particular, a maior quantidade de retweets coincide com uma elevação no número de casos de dengue. Assim, embora relacionado ao mesmo assunto, a série temporal desse grupo pode ser bem mais indicativa e útil para previsão de casos de dengue do que o exemplo anterior.

4.5. Previsão de casos de dengue

O objetivo final da metodologia proposta é ser capaz de fazer a previsão de eventos de interesse. No caso deste trabalho, esse evento corresponde a estimar a incidência de casos de dengue em cidades brasileiras a partir das mensagens coletadas que estão relacionadas a dengue. Portanto, essa seção compara a previsão de casos de dengues usando a metodologia proposta com uma abordagem supervisionada. Para o caso da abordagem supervisionada seguiu-se uma estratégia similar ao que é feito em [Gomide et al. 2011, Souza et al. 2014], em que as mensagens coletadas são classificadas em uma das cinco categorias a seguir: experiência pessoal, informação, opinião, campanha e ironia/sarcasmo.

Para tratar a previsão de forma similar, em ambos os casos foi utilizado o filtro de Kalman para esta etapa de previsão. Cada série temporal obtida a partir do agrupamento é utilizada como um sensor para o filtro de Kalman. No caso supervisionado o número

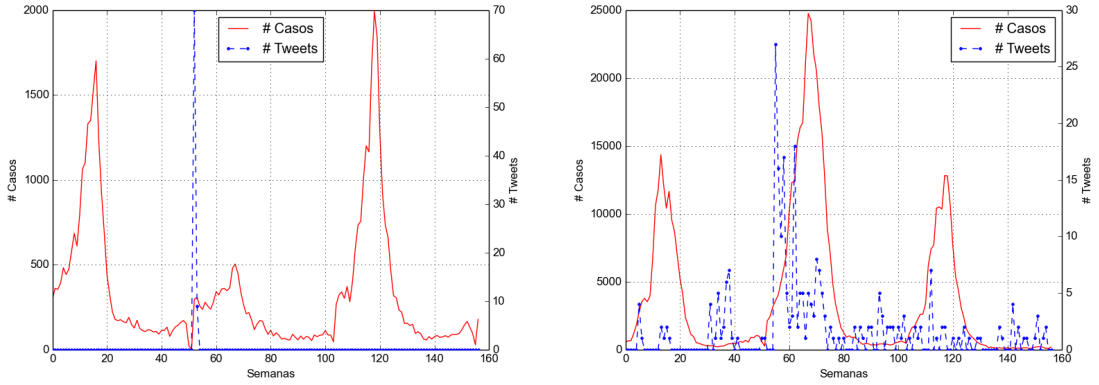


Figura 4. Série temporal do número de tweets e número de casos de dengue. Na *esquerda* estão representadas as séries de casos reais de dengue em São Paulo e a do número de tweets sobre o caso de dengue do ex-jogador Ronaldo. Na *direita* estão representadas as séries de casos reais da cidade do Rio de Janeiro e de um grupo de tweets sobre a mesma piada.

de sensores é fixado no número de categorias pré-definidas, ao passo que na estratégia não-supervisionada apresentada neste trabalho, o número de sensores depende da saída do agrupamento. Em todos os casos, o tamanho total da série é de 157 semanas, das quais as 120 primeiras semanas foram usadas para treinamento do modelo e as últimas 37 semanas são usadas para teste/previsão.

A tabela 1 apresenta os resultados obtidos pela aplicação da metodologia proposta e comparação com uma abordagem supervisionada para classificação das mensagens. Foram selecionadas aleatoriamente 37 cidades que mostram a variedade de cenários que surgem na aplicação da metodologia proposta. A tabela 1 apresenta o número total de mensagens atribuídas a determinada cidade (n); o número de mensagens que restaram após o agrupamento ($n - r$), já que o DBSCAN exclui pontos de ruído; o número de grupos obtidos (ng); a raiz do erro médio quadrado (RMSE) para a parte de previsão, que considera somente as 37 semanas finais, tanto para a proposta desse trabalho quando para a abordagem supervisionada; a acurácia em relação às faixas de incidência determinadas pelo Ministério da Saúde e que são obtidas a partir da previsão em comparação com a faixa verdadeira obtida a partir do número de casos reais.

A partir da tabela observa-se que, a metodologia proposta superou ou se igualou, em termos de acurácia, a abordagem supervisionada em 27 cidades, tendo havido empate em 18 casos. Nos casos de empate na acurácia, a metodologia proposta foi superior em 10 dos 18 casos, considerando o RMSE de previsão. Assim a estratégia não supervisionada proposta neste trabalho se mostrou efetiva em comparação com a abordagem supervisionada para a previsão de casos de dengue.

5. Conclusão

Neste trabalho é investigado o problema de prever eventos reais utilizando dados de redes sociais, os quais apresentam vários desafios como ruído, baixa correlação demográfica e sócio econômica, mudanças de conceito e dificuldade de rotular dados no contexto de uma abordagem supervisionada, entre outros.

Tabela 1. Resultados da aplicação da metodologia proposta e comparação com a abordagem supervisionada.

Cidade	n	$n-r$	ng	RMSE		Acurácia	
				Proposta	Baseline	Proposta	Baseline
Campinas	8577	5001	57	58.329	206.090	100%	100%
Santos	8487	5512	51	679.860	415.180	78%	86%
Ribeirão Preto	7899	4626	59	144.430	1002.500	97%	76%
Aracajú	6613	3674	53	11.377	28.891	100%	100%
Londrina	6030	3837	28	104.790	77.933	100%	97%
Maceió	5746	3579	44	24.775	184.980	100%	100%
Teresina	4962	3575	44	73.309	96.126	100%	100%
Macapá	4940	3098	53	53.678	66.480	100%	100%
Mossoró	4746	2724	41	65.666	25.771	100%	100%
Sorocaba	4648	3613	21	195.570	162.170	95%	97%
Florianópolis	4479	3316	28	19.190	4.899	100%	100%
Juiz de Fora	4347	3091	27	266.340	151.680	95%	100%
Uberlândia	4263	3071	23	293.510	120.71	76%	97%
Campina Grande	4240	2391	39	56.090	81.591	100%	100%
Montes Claros	4073	2251	23	84.177	64.115	100%	100%
S. J. dos Campos	4017	3022	27	107.790	73.885	100%	100%
Maringá	3997	3112	18	58.514	122.38	100%	89%
Vila Velha	3858	2818	17	71.305	64.245	100%	100%
Rio Branco	3711	2654	38	478.900	222.680	70%	81%
Cuiabá	3595	2798	16	95.043	65.295	100%	100%
Bauru	3523	2810	13	186.660	153.710	92%	86%
S. J. do Rio Preto	3220	2484	10	387.940	254.850	65%	86%
Taubaté	2999	2417	9	49.379	81.693	100%	100%
Divinópolis	2797	2411	11	123.480	198.910	95%	65%
Rio Grande	2706	2285	15	0.805	0.850	100%	100%
Guarulhos	2654	2264	8	231.440	119.340	100%	100%
Joinville	2642	2393	8	2.493	3.412	100%	100%
Santo André	2530	2147	5	11.830	23.329	100%	100%
Foz do Iguaçu	2459	2040	8	280.110	157.640	68%	84%
Praia Grande	2153	1892	6	162.830	1670.500	95%	68%
Uberaba	2025	1769	4	342.950	204.290	65%	86%
C. dos Goytacazes	1953	1771	4	126.730	402.490	97%	81%
Ipatinga	1875	1713	5	214.350	112.240	68%	95%
Betim	1106	978	1	454.830	372.830	81%	86%
Sumaré	735	625	1	64.951	111.83	97%	92%
Ap. de Goiânia	395	321	2	285.060	534.820	89%	89%
R. das Neves	132	75	2	499.330	1670.500	76%	68%

É proposta uma estratégia não supervisionada que minimiza os impactos de todos esses desafios e mostrou bons resultados experimentais. A abordagem se baseia em criar modelos personalizados para as cidades, os quais se baseiam nos tópicos que aparecem

na mesma e como a sua intensidade se correlaciona com a intensidade do evento.

Essa estratégia foi avaliada experimentalmente, tendo atingido bons resultados comparado a uma abordagem supervisionada, a qual foi superada considerando uma base de dados contendo dezenas de cidades.

Em termos de trabalhos futuros, pretende-se entender as características temporais das várias séries, e investigar como melhor explorar essas características. Mais ainda, visa-se aplicar a referida técnica a outros cenários.

Referências

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Chakrabarti, D. and Punera, K. (2011). Event summarization using tweets. In *Proc. 6th AAAI Int. Conf. on Weblogs and Social Media*.

Chew, C. and Eysenback, G. (2010). Pandemics in the age of twitter: Content analysis of tweets during the 2009 h1n1 outbreak. *Plos One*, 5.

Cullota, A. (2010). Towards detecting influenza epidemics by analyzing twitter messages. In *Proceedings of 1st workshop on social media analytics*. ACM.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and*, pages 226–231.

Gomide, J., Veloso, A., Meira Jr., W., Almeida, V., Benevenuto, F., Ferraz, F., and Teixeira, M. (2011). Dengue surveillance based on a computational model of spatio-temporal locality of twitter. In *Proceedings of the ACM WebSci Conference*.

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Transaction of the ASME: Journal of Basic Engineering*.

Lampos, V. and Cristianini, N. (2010). Tracking the flu pandemic by monitoring the social web. In *Proceedings of 2nd Workshop on Cognitive Information Processing*. IAPR.

Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of International Conference on World Wide Web*, pages 851–860. ACM.

Santos, J. C. and Matos, S. (2014). Analysing twitter and web queries for flu trend prediction. *Theoretical Biology and Medical Modelling*, 11.

Souza, R. C. S. N. P., de Brito, D. E. F., Cardoso, R. L., de Oliveira, D. M., Meira, Wagner, J., and Pappa, G. L. (2014). An evolutionary methodology for handling data scarcity and noise in monitoring real events from social media data. In *14th Ibero-American Conference on Artificial Intelligence*, pages 295–306.

30th Brazilian Symposium on Databases

tutorials

October 13-16, 2015
Petrópolis – RJ – Brazil

TUTORIALS

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

National Laboratory of Scientific Computing – LNCC
Federal Center of Technological Education of Rio de Janeiro – CEFET/RJ

Tutorials Chair

Javam Machado (UFC)

Editorial

The tutorials at the Brazilian Symposium on Databases (SBBD) have been a large success over the years. SBBD Tutorials present introductory and advanced discussions on topics within the area of databases. Introductory tutorials target an audience consisting of advanced undergraduate and graduate students, as well as attendees from industry. Hence, they can be thought of as "recycling" courses, though they are not about traditional "textbook" topics. Advanced tutorials, on the other hand, cover a topic's state-of-the-art, motivating and exposing potential research issues.

Tutorial proposals are submitted by expert authors in their subject, and selected by the SBBD tutorial program committee. This year, we had six high quality submissions: five from Brazil and one from the USA. The tutorial program committee selected two of these submissions to be presented at SBBD in Petropolis.

The first tutorial is entitled "Reprodução de Experimentos Científicos: Teoria e Prática" and will be presented in Portuguese by Ary Oliveira, Daniel de Oliveira and Marta Mattoso. It presents challenges in reproducing results from scientific experiments and also current approaches to support reproducibility of scientific research. It covers concepts, the main tools and the role of cloud computing in addressing large-scale experiments' reproducibility. Ary Oliveira, Daniel de Oliveira and Marta Mattoso are all professors respectively at Universidade Federal do Tocantins (UFT), Universidade Federal Fluminense (UFF) and Universidade Federal do Rio de Janeiro (UFRJ), Brazil.

The second tutorial, "Database Kernels Tailored for Data Exploration", will be presented by Stratos Idreos. The goal is to survey recent developments in the emerging area of database systems tailored for data exploration. It presents recent literature on database kernels that are tailored for data exploration by adapting their storage and access patterns to varying workloads. It covers developments in adaptive indexing, data loading and storage, but also gesture-based data exploration. Stratos Idreos is an assistant professor of Computer Science at Harvard University, USA.

I would like to invite you all to attend the tutorials this year in Petropolis. Both tutorials cover very hot topics, and we hope you will take advantage of the expertise of their presenters. I'd also like to thank all the authors for contributing to SBBD 2015.

Javam Machado (UFC)
SBBD 2015 Tutorials Chair

Table of Contents (Tutorials)

tutorial:01

Database Kernels Tailored for Data Exploration

Stratos Idreos [1]

[1]Harvard University
Cambridge, MA - USA

stratos@seas.harvard.edu

Abstract. *Data exploration is about efficiently extracting knowledge from data even if we do not know exactly what we are looking for. In this tutorial, we survey recent developments in the emerging area of database systems tailored for data exploration. Specifically, we survey recent literature that focuses on the vision of database kernels that are tailored for data exploration by adapting their storage and access patterns to varying workloads.*

1. Description

Traditional data management systems assume that when users pose a query a) they have good knowledge of the schema, meaning and contents of the database and b) they are certain that this particular query is the one they wanted to pose. In short, we assume that users know what they are looking for. In addition, traditional DBMSs are designed for static scenarios with numerous assumptions about the workload. For example, state-of-the-art systems assume that there will be a tuning phase where a database administrator tunes the system for the expected workload. Again, this assumes that we know the workload, we know that it will be stable and we also have enough idle time and resources to devote to tuning.

The above assumptions were valid for the static applications of the past and they are still valid for numerous applications today. However, as we create and collect increasing amounts of data, we are building more dynamic and data-driven applications that do not always have the same requirements that database systems have tried to address during the past five decades. This is true across many businesses as well as scientific research that continuously becomes more data-driven.

In this tutorial we survey recent literature on designing from the ground up new database kernels that are tailored to provide efficient access to data even if we do not know what we are looking for up front, i.e., systems that are tailored for data exploration [Idreos, Papaemmanouil and Chaudhuri (2015), Idreos 2013]. Specifically, we will survey recent developments in adaptive indexing [Idreos, Kersten and Manegold 2007], adaptive data loading [Idreos, Alagiannis, Johnson and Ailamaki 2011], adaptive storage [Alagiannis, Idreos and Ailamaki 2014] as well as new ideas on how to provide exploration capabilities by co-designing user-interfaces with database kernels [Idreos and Liarou 2013].

Adaptive Indexing. In a data exploration scenario we are searching for interesting data patterns without knowledge of what we are looking for. Yet traditional database systems rely heavily on tuning actions and accurate workload knowledge to achieve good performance. One of the most critical tuning actions is that of choosing the proper set of indexes. Making strict a priori choices means that a system is not well prepared for an exploratory scenario where users may focus on arbitrary data parts at

different times. Research on adaptive indexing introduces the idea of creating indexes incrementally and adaptively during query processing based on the columns, tables and value ranges that queries request. Indexes are built gradually; as more queries arrive indexes are continuously fine-tuned. Adaptive indexing has been studied to improve selections in column-stores, and has been shown to work in late materialization architectures, to allow for incremental and partial projections, to be robust in workload changes, to absorb updates efficiently and adaptively and to enable multi-query processing via concurrency control. In addition, the basic algorithms have been studied in depth in the face of trade-offs such as adaptation speed and initialization costs as well as they have been optimized for modern hardware.

Adaptive Loading. During data exploration not all data is needed. Adaptive loading exploits this fact and introduces the notion that users can start querying a database system (with efficient response times) even before all data is loaded or even leaving some parts of the data unloaded, effectively enabling efficient raw data access.

Adaptive Storage. The way we store data defines the best possible ways to access it. There is no perfect storage layout; instead there is a perfect layout for each individual data access pattern. Modern systems rely on static layouts and build the whole architecture around a single layout. In a data exploration scenario we cannot a priori decide what is a good layout, as we do not know the exact query patterns up front, leading to sub-optimal performance for traditional static systems. In this part of the tutorial, we discuss recent work that aims at removing this problem through adaptive storage.

Gesture-based Data Exploration. Last we will discuss the vision towards systems that make interaction with a data simple and intuitive without requiring significant expertise from data scientists. Users interact with the system via gestures as opposed to complex languages. The user interface is co-designed with the database kernel in order to provide very fast reaction to user input and the illusion of "touching the data".

Summary. This tutorial takes a close look into database kernel design and discusses designs that do not require any set-up or workload knowledge but still provide efficient response times by being able to adapt to incoming queries and data. The ideas on data exploration kernels will be presented in the context of modern main-memory optimized systems with columnar or hybrid data layouts that fully utilize modern hardware.

About the Author

Stratos Idreos is an assistant professor of Computer Science at Harvard University where he leads DASlab, the Data Systems Laboratory@Harvard SEAS. Stratos works on data systems architectures with emphasis on designing systems for big data exploration. For his doctoral work on Database Cracking, Stratos won the 2011 ACM SIGMOD Jim Gray Doctoral Dissertation award and the 2011 ERCIM Cor Baayen award as from the European Research Council on Informatics and Mathematics. In 2010 he was awarded the IBM zEnterpise System Recognition Award by IBM Research, and in 2011 he won the VLDB Challenges and Visions best paper award. In 2015 he

received an NSF CAREER award and was awarded the 2015 IEEE TCDE Early Career Award from the IEEE Technical Committee on Data Engineering.

References

Idreos, S., Papaemmanouil O., and Chaudhuri S. (2015) Overview of Data Exploration Techniques. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 2015

Idreos, S. (2013) Big Data Exploration. Taylor and Francis, 2013

Idreos, S., Kersten, M. L., and Manegold, S. (2007) Database cracking. In Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR), 2007

Idreos, S., Alagiannis, I., Johnson, R. and Ailamaki, A. (2011) Here are my Data Files. Here are my Queries. Where are my Results? In Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR), 2011

Alagiannis, I., Idreos, S., and Ailamaki, A. (2014) H2O: a hands-free adaptive store. In Proceedings of the ACM SIGMOD Conference on Management of Data, 2014

Idreos S. and Liarou E. (2013) dbtouch: Analytics at your fingertips. In Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR), 2013

tutorial:02

Reprodução de Experimentos Científicos: Teoria e Prática

Ary H. M. Oliveira[1,3], Daniel de Oliveira[2], Marta Mattoso[3]

[1]Ciência da Computação, Universidade Federal do Tocantins (UFT)
Palmas, Tocantins

[2]Instituto de Computação, Universidade Federal Fluminense (UFF)
Niterói, RJ

[3]COPPE, Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ

aryhenrique@uft.edu.br, danielcmo@ic.uff.br, marta@cos.ufrj.br

Resumo. *A capacidade de reproduzir um experimento científico computacional (que chamaremos apenas de experimento) faz parte do princípio básico do método científico. Reproduzir um experimento passa pela tentativa de terceiros em reproduzir os procedimentos e os dados apresentados em um artigo científico. Por meio da reprodução, é possível comparar resultados, e avaliar os métodos empregados, para poder dar credibilidade e confiança à pesquisa científica. Neste tutorial, analisamos desafios em reproduzir resultados de experimentos e as abordagens existentes para apoiar tal reprodução. Apresentamos os conceitos, as principais ferramentas, o uso de nuvens computacionais e discutimos os problemas envolvendo reprodução de experimentos em larga-escala.*

1. Descrição

O tutorial está organizado em três partes. A primeira aborda os conceitos principais, a motivação, exemplos e os desafios da reprodução de experimentos científicos na computação. A segunda parte aborda um detalhamento dos conceitos da reprodução de experimentos, caracterizando o ciclo de vida da reprodução, envolvendo a relação da publicação de artigos com a reprodução dos resultados publicados e as principais tecnologias que vêm servindo de base de apoio para a reprodução. Finalmente, a terceira parte apresenta as principais ferramentas existentes e uma reprodução interativamente por meio do *software Reproescience*, alvo de desenvolvimento de pesquisas dos autores deste tutorial.

Parte I

a) Conceitos principais: Serão abordados os termos e definições relacionados com a reprodução de experimentos científicos na computação.

b) Motivação: Serão apresentados exemplos de experimentos científicos, fazendo uso dos conceitos definidos. A motivação será mostrar o porquê de o desenvolvimento de um experimento ter de vir acompanhado de recursos que viabilizem a sua reprodução, para que seja possível verificar e validar os resultados gerados.

c) Desafios: Apresentação das dificuldades em prover os recursos de reprodução de resultados e perspectivas de como as áreas de banco de dados e computação em nuvem podem contribuir com soluções centradas em dados e infraestrutura computacional para permitir que um experimento científica seja passível de reprodução.

O progresso da ciência depende da efetiva disseminação e reprodução de pesquisas existentes, porém, é necessário ter acesso ao material usado na produção dos resultados dos experimentos para que seja possível validá-los [Brammer et al. 2011]. A computação vêm auxiliando a ciência aumentando a produtividade dos pesquisadores e a qualidade da sua produção na análise de grande quantidade, complexidade e variedade de dados [Karpathiotakis, Branco, Alagiannis and Ailamaki 2014]. Entretanto, para que a comunidade científica possa dar prosseguimento às novas descobertas, tal desenvolvimento deve vir acompanhado de abordagens que viabilizem a reprodução do experimento, para que seja possível verificar e validar os resultados gerados.

Por meio da reprodução é possível comparar resultados, e consequentemente, avaliar os métodos empregados, e com isso, dar credibilidade e confiança à pesquisa científica [Goble 2012]. O termo reprodução está ligado ao conceito de experimento reprodutível que é a capacidade de se fornecer mecanismos que possibilitem a repetição e a reprodução de um experimento por outro cientista. O experimento reprodutível permite que novos trabalhos sejam produzidos a partir de pesquisas já consolidadas, de forma que uma nova solução seja desenvolvida a partir de um ponto mais avançado. Isto elimina a necessidade de se reconstruir teorias e/ou refazer experimentos já existentes a partir de um ponto inicial para se confirmar (ou refutar) uma determinada hipótese científica. [Freire, Bonnet and Shasha 2012] definem a reprodução na computação, como: "um experimento computacional desenvolvido no tempo *t* em *hardware*/sistema operacional *s* utilizando os dados *d* é reprodutível se ele puder ser executado no momento *t'* em um sistema *s'* com os dados *d'*, que são semelhantes (ou, potencialmente, os mesmos) que *d'*.

Parte II

a) Experimentos reprodutíveis: Apresentação dos principais conceitos do ciclo de vida da reprodução de resultados de experimentos, descrevendo as características e funções, para orientar na seleção dos mecanismos necessários para armazenar, gerenciar, recuperar e reproduzir os elementos de um experimento

b) Verificação e validação de resultados: Apresentar as inciativas de submissão de artigos e artigos executáveis.

c) Tecnologias de apoio: Apresentação dos conceitos de *workflows* científicos, proveniência de dados e o uso de nuvens de computadores.

Um experimento reprodutível é aquele cujo produto final é composto pelo artigo publicado, no formato digital ou físico, acompanhado de todos os objetos de pesquisa utilizados para a sua concepção. Os objetos de pesquisa são compostos pelos diferentes artefatos usados ou gerados através de um experimento científico [Belhajjame, Roure and Goble 2012], dentre eles: o artigo científico (manuscrito), hipóteses, conjunto de dados usados e/ou produzidos (entrada, intermediários, finais, metadados e proveniência), anotações e documentação, infraestrutura de *hardware* e *software*,

plataforma de *software* e os parâmetros de configuração. Os objetos de pesquisa são como uma abstração usada para a comunicação, compartilhamento e reuso de resultados de pesquisas [Belhajjame, Roure, and Goble 2012]. O compartilhamento dos objetos de pesquisa facilita o desenvolvimento da ciência, não apenas no processo de validação dos resultados, mas também pela possibilidade de exploração de novas hipóteses e combinação com outros métodos. A gerência de objetos de pesquisa traz oportunidades de pesquisas e desenvolvimento na área de Bancos de Dados.

A verificação da reprodução de experimentos tem sido alvo de preocupação da comunidade científica e tem sido discutida em diversos periódicos e chamadas de trabalhos em diferentes áreas do conhecimento apoiadas pela computação. Trabalhos submetidos a eventos como o congresso SIGMOD da ACM são rotulados como reprodutíveis ou compartilhados [Bonnet et al. 2011]. O objetivo é que os trabalhos reprodutíveis sejam submetidos juntamente com o protocolo e os recursos necessários para reproduzir os resultados obtidos no artigo científico, englobando: (1) o protótipo do sistema, com o código-fonte, arquivos de configuração e o ambiente operacional, (2) o conjunto de experimentos, contendo os sistemas de configuração e inicialização, scripts e os protocolos de avaliação usados para produzir os resultados experimentais, e (3) os *scripts* necessários para transformar os dados em equações, gráficos e tabelas que são apresentados no artigo. O Grande Desafio de Artigos Executáveis, organizado pela Elsevier foi um evento realizado na forma de uma competição cujo objetivo foi selecionar as melhores abordagens de concepção e gerenciamento de artigos executáveis. Os artigos executáveis são definidos como um objeto digital publicado com todo o código necessário para reproduzir os cálculos e visualizar os resultados obtidos com dados e programas do estudo [Nowakowski et al. 2011]. O objetivo é aumentar a compreensão e a reprodução das publicações eletrônicas permitindo que os leitores e avaliadores possam interagir, validar e explorar os experimentos [Koop et al. 2011].

Parte III

a) Estado da Arte: Apresentação das principais abordagens e ferramentas que apoiam a reprodução de experimentos científicos (por ex.: Collage [Nowakowski et al. 2011], Paper Maché [Brammer et al. 2011] e Reprozip [Chirigati, Shasha and Freire 2013]).
b) Parte prática: Exemplo da plataforma de compartilhamento e reprodução de experimentos utilizando a virtualização em nuvens computacionais e a ferramenta *ReproeScience*.

Agradecimentos

Este trabalho foi parcialmente financiado pelas agências CAPES, CNPq e FAPERJ.

Sobre os Autores

Ary H. M. Oliveira é professor assistente no Curso de Ciência da Computação da Universidade Federal do Tocantins desde 2.008. Cursa o doutorado no Programa de Engenharia de Sistemas e Computação da COPPE/Universidade Federal do Rio de Janeiro sob orientação da Profa. Marta Lima de Queirós Mattoso e do Prof. Daniel de Oliveira. Seus interesses de pesquisa incluem bancos de dados, computação em nuvem,

gerência de *workflows* científicos, paralelismo de dados, bioinformática e mineração de dados.

Daniel de Oliveira é professor do Instituto de Computação da Universidade Federal Fluminense (UFF) desde 2013. Recebeu o grau de Doutor em Engenharia de Sistemas e Computação pela COPPE/UFRJ em 2012 sob a orientação da Profa. Marta Lima de Queirós Mattoso. Seus interesses de pesquisa incluem bancos de dados, computação em nuvem, gerência de workflows científicos, paralelismo de dados, bioinformática e mineração de dados. Publicou mais de 70 artigos em periódicos indexados e em congressos nacionais e internacionais. É membro da ACM, IEEE e SBC. Vem publicando com frequência em eventos de prestígio internacional de computação em nuvem como o *IEEE Cloud* e o *IEEE e-Science*, além de ter recebido o prêmio de melhor artigo do *2nd International Workshop on Cloud Computing and Scientific Applications* (CCSA) realizado em conjunto com o *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (CCGrid 2012).

Marta Mattoso é professora titular no Programa de Engenharia de Sistemas e Computação da COPPE/Universidade Federal do Rio de Janeiro e bolsista de produtividade em pesquisa do CNPq nível 1B. Concluiu o doutorado na COPPE/UFRJ em 1993. Orientou 15 teses de doutorado e 55 de mestrado. Publicou mais de 200 artigos completos em revistas e congressos internacionais e nacionais. Vem participando de Comitês de Programa de congressos nacionais e internacionais, revisora ad-hoc de revistas nacionais e internacionais. É a *Scientific Chair da Provenance Week* 2016, reunindo os eventos IPAW e TaPP. Sócia da SBC, IEEE e ACM. Foi diretora de publicações da SBC no período de 2005 a 2007. Coordena diversos projetos de pesquisa com financiamento do CNPq, Capes/Cofecub, INRIA, FAPERJ e Finep, nas áreas de distribuição e paralelismo em bancos de dados, *workflows* científicos em ambientes de paralelismo e gerência de dados de proveniência. Trabalha de modo interdisciplinar com pesquisadores de áreas como bioinformática e engenharia do petróleo.

Referencias

Belhajjame, K., Roure, D. D., Goble, C. A. (2012) "Research Object Management: Opportunities and Challenges". February 2012.

Bonnet, P et al. (2011) "Repeatability and Workability Evaluation of SIGMOD 2011", SIGMOD Rec., v. 40, n. 2, pp. 45-48, set. 2011.

Chirigati, F., Shasha, D., Freire, J. (2013) "Packing Experiments for Sharing and Publication". In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pp. 977-980, 2013.

Goble, C. (2012) "The Reality of Reproducibility in Computational Science: reproduce? repeat? rerun? and does it matter. Keynotes and Panels", 8th IEEE International Conference on e-Science, v. 327, n. 5964, pp. 415-416, October 2012.

Freire, J., Bonnet, P., Shasha, D. (2012) "Computational Reproducibility: State-of-the-art, Challenges, and Database Research Opportunities". In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD '12, pp. 593-596, New York, NY, USA, 2012.

Brammer, G. R. et al. (2011) "Paper Mâché: Creating Dynamic Reproducible Science", Procedia Computer Science, v. 4, n. 0, pp. 658-667, 2011.

Karpathiotakis, M., Branco, M. Alagiannis, I, Ailamaki, A. (2014) Adaptive Query Processing on RAW Data". In: Proccedings of VLDB Endowment, vol. 7, n. 32, p. 1119–1130, sep. 2014.

Koop, D. et al. (2011) "A Provenance-Based Infrastructure to Support the Life Cycle of Executable Papers", Procedia Computer Science, v. 4, n. 0, pp. 648-657, 2011.

Klinginsmith, J., Mahoui, M., Wu, Y. (2011) "Towards Reproducible e-Science in the Cloud". In: Cloud Computing Technology and Science, 2011 IEEE Third International Conference on, pp. 582-586, Nov 2011.

Nowakowski, P. et al. (2011) "The Collage Authoring Environment", Executable Paper Grand Challenge - ICCS 2011, n. 4, pp. 608-617, 2011.

30th Brazilian Symposium on Databases

talks

October 13-16, 2015
Petrópolis – RJ – Brazil

INVITED TALKS

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

National Laboratory of Scientific Computing – LNCC
Federal Center of Technological Education of Rio de Janeiro – CEFET/RJ

Table of Contents (Invited Talks)

InvTalk:01

AsterixDB: A Counter but Intuitive Approach to Big Data Management

Michael J. Carey

University of California, Irvine
Irvine, CA, USA

mjcarey@ics.uci.edu

We are living in the Big Data era, and we are witnessing a shift in the role of data management system: Rather than "just" being the systems of record at the heart of traditional enterprises, modern Big Data management systems must model, capture, track, and react to the current state of the world. Doing so requires the ingestion of event data, arriving from a variety of devices, as well as enabling query access to the history of captured data over time. These requirements span a variety of scientific disciplines, including the handling of data produced by a variety sensors in health care, environmental monitoring applications, traffic monitoring, dynamic social network data, and many other domains.

AsterixDB is an open source Big Data Management System (BDMS) with a feature set that's very different than those of other platforms in today's Big Data ecosystem. The system was initially co-developed by UC Irvine and UC Riverside, starting in 2009 and leading eventually to its first beta release in mid-2013. It has recently moved to Apache, where AsterixDB is now an active incubating project. Many of the system's key design decisions relate to the aforementioned shift. This talk will first briefly review AsterixDB's data model, query language, and scale-out architecture. It will then examine a number of counter-cultural aspects of the AsterixDB system, including where its data lives, its runtime architecture, its approach to streaming data, its view of transactions, and its features for handling time-based data.

About the Author

Michael J. Carey is a Bren Professor of Information and Computer Sciences at UC Irvine. Before joining UCI in 2008, Carey worked at BEA Systems for seven years and led the development of BEA's AquaLogic Data Services Platform product for virtual data integration. He also spent a dozen years teaching at the University of Wisconsin-Madison, five years at the IBM Almaden Research Center working on object-relational databases, and a year and a half at e-commerce platform startup Propel Software during the infamous 2000-2001 Internet bubble. Carey is an ACM Fellow, a member of the National Academy of Engineering, and a recipient of the ACM SIGMOD E.F. Codd Innovations Award. His current interests all center around data-intensive computing and scalable data management (a.k.a. Big Data).

InvTalk:02

Towards a Web of Semantically Integrated Data

Ana Maria Moura

Laboratório Nacional de Computação Científica
Petrópolis, RJ, Brazil

anamoura@lncc.br

The Semantic Web such as envisaged by Tim Berners Lee in the early years of this century foresaw a universal medium to exchange data, where resources could be published and interconnected with others, providing interoperation between users and machines, in order to facilitate their computational tasks. Since then a flood of semantic technologies have been created, revolutionizing the way to store, access, and communicate digital information. Among these, Linked Data (LD) emerged as an innovative technology for realizing the Semantic Webvision of making the Web a global, distributed, semantics-based information system. Despite being a powerful strategy to link data resources, these links are not always created adequately, i.e., frequently they are not semantically associated with other resources, hindering the interoperability between them. In order to minimize this problem, some LD technologies have been recently developed. This presentation aims to give an evolutionary overview of the state of the art, focusing on the main Semantic Web technologies that have been responsible for: (i) describing data semantically enriched; (ii) linking resources on the Web of data according to their semantic meaning; and (iii) providing the efficient consumption of these resources. In this context, I will show the main works I have been carrying out in the domain for these last 15 years.

About the Author

Ana Maria de C. Moura is a research collaborator at Extreme Data Laboratory (DEXL) at LNCC, where she has been working for 5 years on applying semantic technologies to improve database integration, data retrieval and data publishing. Graduated on Computer Science from the UTC-Université de Technologie de Compiègne (Compiègne, France) in 1984, and M.Sc in Computer Science, COPPE/Federal University of Rio de Janeiro (UFRJ) 1979, she worked for more than 20 years as professor and coordinator of the Database research area at the Computing Engineering Department of IME-Military Institute of Engineering, Rio de Janeiro, where she still contributes as collaborator professor. More than 50 advised Master theses concluded and more than 190 national and international publications. Her main domain interests are: databases, semantic data integration, conceptual modeling, metadata management,ontologies, linked open data.

InvTalk:03

Query Languages for Graph Databases: Bridging the Gap Between Theory and Practice

Pablo Barceló

Universidade do Chile
Santiago do Chile, Chile

pbarcelo@dcc.uchile.cl

Graph databases have been an active research topic as of late, owing to their applications in areas such as the Semantic Web and Social Network Analysis. In this talk, we present features that are used by several real-world query languages for graph databases. These include navigation, pattern matching, path variables, ungrouping and data comparisons. We analyze their properties, having two main objectives in mind. First, we would like to provide simple yet general definitions of key features of graph query languages, to enable analyses of their different uses in practice and to establish a common ground for their theoretical study. Second, we would like to use them for studying tradeoff between expressiveness and efficiency of existing query languages for graph databases.

About the Author

Pablo Barceló is an Associate Professor in the Department of Computer Science at the University of Chile. He received his PhD from the University of Toronto in 2006. His main research interest are in the areas of databases and logic in computer science. He has written over 30 technical papers and served on the program committees of major conferences in his area (ACM PODS, SIGMOD, ICDT, STACS, ACM/IEEE LICS). He has also been an invited tutorial speaker at ACM PODS 2013. He is a member of the editorial board of Logical Methods in Computer Science and the editor of the Database Principles Column of the SIGMOD Record.

Patrocínio Diamante

Prata

Bronze

Organização

Promoção