

# **Uma abordagem de código único para aplicações independentes de provedor de bases de dados relacionais**

**Willian Eduardo de Moura Casante<sup>1</sup>**

<sup>1</sup>Fundação CPqD – Centro de Pesquisa e Desenvolvimento em Telecomunicações  
Rod. SP-340 km 118,5 CEP 13.086-902 – Campinas – SP – Brasil

wcasante@cpqd.com.br

**Abstract.** *Relational databases uses specific SQL extensions for their product functions, which makes the development of complex applications unfeasible mainly because compatibility with more than one relational database would require specific codes for each database. This paper describes the development of multi-database applications based on a methodology of creation of the installation artifacts for the database and a technique to develop multi-database sources codes.*

**Resumo.** *Os bancos de dados relacionais fornecem funcionalidades específicas de produtos através de extensões da linguagem SQL, viabilizando o desenvolvimento de aplicações complexas e compatíveis com mais de um banco de dados relacional sem que essas aplicações tenham código específico para cada um deles. Este artigo aborda o desenvolvimento de aplicações totalmente independentes de banco de dados através de uma metodologia de criação de artefatos de instalação da estrutura de dados e de uma técnica de desenvolvimento de código-fonte independentemente de produto.*

## **1. Introdução**

A linguagem SQL (*Structured Query Language*) [Chamberlin et al. 1981] é uma linguagem de pesquisa e definição de estrutura de dados, inicialmente criada para SGBDs (Sistemas Gerenciadores de Banco de Dados), fortemente inspirada na álgebra relacional [Elmasri et al. 2005].

Necessidades não mapeadas no padrão SQL desenvolvido e aceito pela indústria exigiram dos fabricantes de SGBDs a definição de construções SQL específicas para seus produtos e a criação de extensões da linguagem que possibilitam o acesso das suas funcionalidades proprietárias.

Essas diferenças entre as diversas implementações da linguagem SQL prejudicam o desenvolvimento de um sistema computacional compatível com dois ou mais SGBDs pois, para cada SGBD atendido, um conjunto de arquivos específicos para aquele deverá ser mantido, tanto para a instalação do sistema quanto para o funcionamento correto de seu código-fonte.

## **2. Análise da incompatibilidade entre SGBDs**

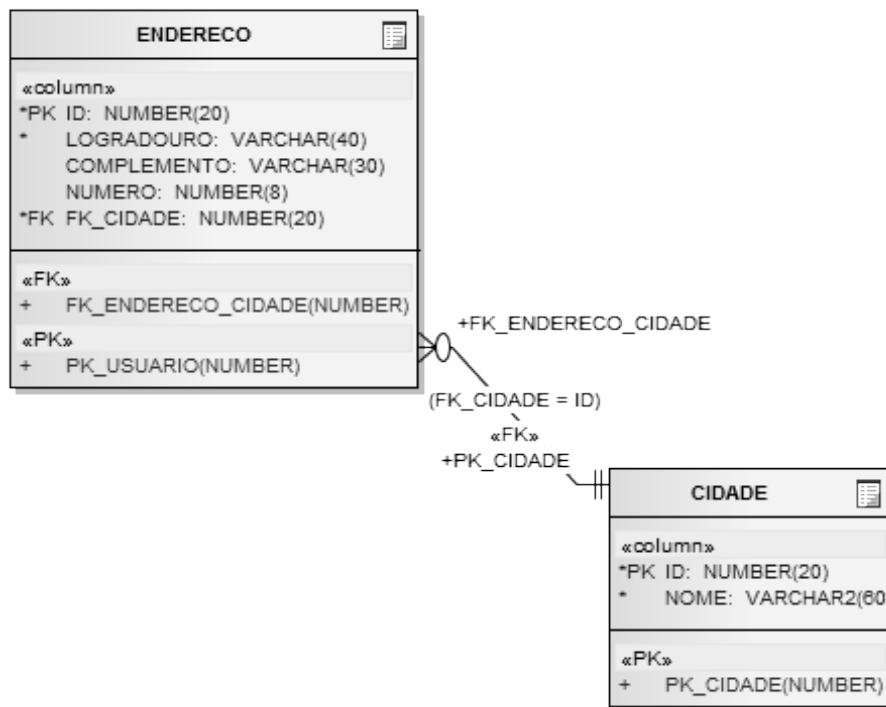
Um sistema computacional que realiza o armazenamento de suas informações em SGBDs possui seus arquivos de instalação organizados em DDLs (*Data Definition Language*) e DMLs (*Data Manipulation Language*).

DDL é um arquivo de metadados, que define a estrutura das tabelas de um banco de dados, composto por comandos SQL como, por exemplo, *CREATE*, *ALTER* e *DROP*. Os arquivos do tipo DDL são os primeiros a serem executados durante a instalação de um sistema computacional, visto que criam o arcabouço do sistema no SGBD.

DML é um arquivo de dados, cuja estrutura respeita fielmente a estrutura definida em seu respectivo arquivo DDL. Durante o desenvolvimento de um sistema pode, se necessário, inicializar o banco de dados; essa inicialização é colocada em arquivos do tipo DML. Os arquivos DML possuem comandos de *INSERT*, *UPDATE*, *DELETE* e *SELECT*, também utilizados de forma embutida dentro do código-fonte do sistema.

## 2.1. A estrutura dos dados em um SGBD

Durante a análise de um sistema, uma das tarefas é a definição de seu modelo de banco de dados. Essa definição normalmente envolve a construção de entidades básicas do sistema, em uma linguagem visual de alto nível, como, por exemplo, o modelo de entidades e relacionamentos [Barbieri 1994] ou a UML (*Unified Modeling Language*) [Jacobson et al. 1996]. Um exemplo simples de modelo UML pode ser observado na Figura 1. Através desse modelo inicial, os analistas de sistemas realizam um aprofundamento na modelagem, através da definição de todas as entidades e de seus relacionamentos, para que o sistema possa ser construído sobre esse modelo proposto. Conforme o sistema é desenvolvido, novas entidades e relacionamentos são incluídos e estes, detalhados e então o sistema evoluído para os contemplar.



**Figura 1. Exemplo de modelagem via UML**

A partir da definição gráfica das entidades do modelo de banco de dados, um especialista deverá gerar um ou mais arquivos DDL, contendo os comandos SQL equivalentes

para a criação daquele modelo em um SGBD específico. Existem ferramentas que geram automaticamente esses arquivos a partir dos diagramas propostos, que estão fora do escopo deste artigo.

Um sistema que se propõe a ser instalado em mais de um SGBD apresenta algumas estratégias que podem ser adotadas, para a criação de seus arquivos DDL:

- Criação de um ou mais arquivos DDL por SGBD – exige a replicação de artefatos de trabalho (arquivos DDL) para cada um dos SGBDs; adiciona risco de erro na conversão, caso realizada de forma manual, como o risco da falta de sincronia entre as atualizações dos artefatos. Como exemplo, um profissional obtém uma DDL para o SGBD Oracle e deve convertê-la para funcionar no SGBD PostgreSQL e causa um erro ao não trocar NUMBER(20) por NUMERIC(20);
- Criação de um único arquivo DDL para um SGBD e utilização de uma ferramenta de conversão para os outros SGBDs – a conversão por ferramenta mitiga os possíveis erros porém adiciona um passo à geração do pacote de instalação do sistema: a conversão dos arquivos DDL. A existência de uma ferramenta que realize essa conversão, bem como sua aquisição, caso seja comercial, é um requisito primário, que pode ser substituído pelo desenvolvimento da própria ferramenta, como realizado no desenvolvimento da ferramenta JExodus [Neto and Passos];
- Geração automática de modelo através de código-fonte – alguns arcabouços de programação (por exemplo a especificação EJB (*Entity JavaBean*) do Java EE [Patel et al. 2006]) permitem a definição de entidades programáticas e a geração do modelo de banco de dados de forma transparente. Esse formato não é aproveitável em sistemas que necessitam controlar com precisão seus modelos de banco de dados ou realizar atualizações pois a instalação e a definição do modelo ficam a encargo do arcabouço. Pode haver conflito no que diz respeito à segurança, onde a instalação da estrutura do banco de dados ficar a cargo de um administrador do SGBD, ou durante a execução, se o sistema obtiver uma conexão na qual apenas controla a atualização dos dados neste SGBD.

## 2.2. A manutenção dos dados em um SGBD

A definição da estrutura, ou modelo, de um sistema em um SGBD, conforme descrito na Subseção 2.1, deverá permitir a inserção, alteração, exclusão e consulta dos dados presentes nessa estrutura. Para realizar essas operações existem os comandos DML.

A instalação de um sistema pode necessitar de dados iniciais, inseridos durante a própria instalação ou configurados por uma equipe de implantação, dados estes presentes em scripts DML e também, durante a utilização do sistema, realiza esses comandos no SGBD para que os dados sejam manipulados na citada estrutura.

Portanto, podemos listar dois cenários a serem tratados:

- A criação dos dados iniciais em uma estrutura, durante a instalação de um sistema, cujos comandos DML deverão ser compatíveis com diversos SGBDs ou convertidos conforme necessidade;
- O ciclo de produção do sistema, pós-instalação, no qual o sistema emitirá comandos DML para a manipulação dos dados presentes na estrutura instalada. Esses comandos deverão ser genéricos ou estruturados de forma que o sistema não

conheça as diferenças entre os diversos SGBDs e suas peculiaridades, permitindo que o código-fonte, geralmente em linguagem de alto nível (como o Java), seja único e independente de SGBD.

### 2.3. Linguagens específicas

Cada SGBD pode prover uma ou mais linguagens de programação específicas, conhecidas como linguagem de *script*, nas quais é possível adicionar lógica de programação além dos comandos DDL e DML. Essas linguagens (por exemplo, o PL/SQL do SGBD Oracle [Feuerstein 2002]) possuem sintaxe proprietária e muitas vezes de difícil conversão automática entre os diversos SGBDs.

Podem ser definidas funções (*Stored Procedures*), diretamente no SGBD, utilizando-se essas linguagens de programação e realizar a chamada dessas funções diretamente do código-fonte do sistema.

## 3. Arquiteturas passíveis de utilização

Enumeram-se algumas possíveis arquiteturas de acesso e manutenção dos dados de múltiplos SGBDs, para um sistema de software:

- Desenvolvimento tradicional – com projeto de banco de dados, geração de arquivos DDL e DML, controle manual das alterações e controle de arquivos por SGBD;
- Arcabouço de desenvolvimento, como o Ruby on Rails [St.Laurent et al. 2012] ou o Java Persistence API [Patel et al. 2006], nos quais o próprio arcabouço trata a criação e a evolução do banco de dados;
- Metodologia proposta neste artigo, que unifica os itens citados anteriormente na manutenção de uma única base de artefatos relativos à base de dados.

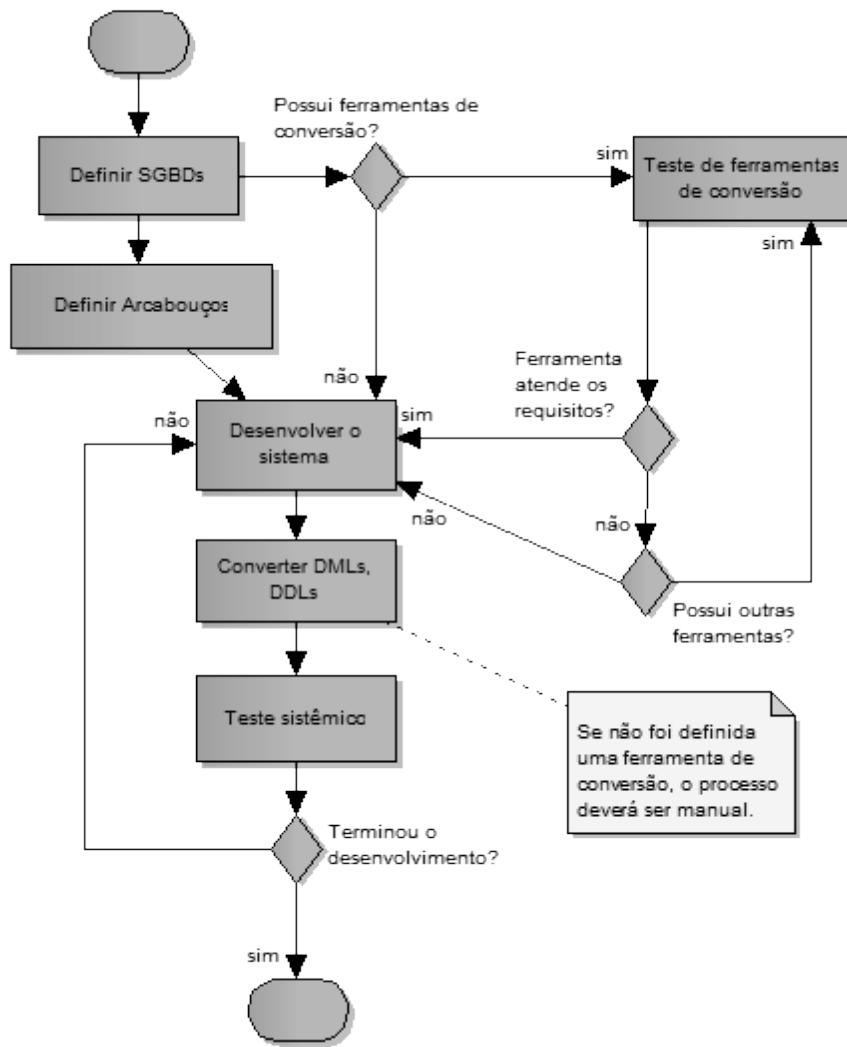
## 4. Adotando uma metodologia única

A metodologia, proposta na Figura 2, une o desenvolvimento tradicional aos arcabouços de desenvolvimento, aproveitando o seu melhor sem limitar-se às fronteiras impostas pelos arcabouços, facilitando assim o uso em sistemas de grande porte, que exigem a modelagem de bases de dados extremamente complexas.

A premissa para o desenvolvimento do sistema é a capacidade de este ser executado em dois ou mais SGBDs. Se essa necessidade não existir, o desenvolvimento tradicional poderá ser adotado. Do mesmo modo, se a complexidade da estrutura de dados for baixa, independentemente do tamanho do sistema, será possível utilizar por completo e com todas as vantagens, um dos arcabouços de desenvolvimento ágil (como, por exemplo, o Ruby on Rails).

### 4.1. Definição de um SGBD “base”

Antes de iniciar o desenvolvimento, para determinar o SGBD “base”, no qual os arquivos DML e DDL serão escritos, deverão ser realizados uma pesquisa e um estudo das ferramentas disponíveis para conversão de cada SGBD suportado pelo sistema. Muitas ferramentas requerem de licença de uso, o que deve ser levado em conta na análise e na tomada de decisão. Pode-se chegar à conclusão de que não há ferramenta compatível com os requisitos impostos, e, nesse caso, será utilizada a conversão manual. É recomendável adicionar ao projeto o risco de defeitos causados pela conversão manual dos artefatos.



**Figura 2. Metodologia sugerida para o desenvolvimento do sistema**

#### 4.2. Definição dos arcabouços de programação

Uma vez conhecida a linguagem de programação na qual o sistema será desenvolvido, aconselha-se escolher também um arcabouço de abstração de acesso aos dados. Esses arcabouços abstraem detalhes dos diversos SGBDs suportados, padronizam o código-fonte, e facilitam o desenvolvimento do acesso aos dados, pois normalmente oferecem facilidades para tal.

Deve ser feito um estudo de uso, conforme feito por Viana e Borba [Viana and Borba 1999], a partir de arcabouços atuais da linguagem de programação escolhida (por exemplo, para o Java, o Hibernate [Bauer and King 2005] ou a especificação Java Persistence API [Patel et al. 2006]), para determinar o melhor conjunto de arcabouços. Caso os SGBDs escolhidos suportem a orientação à objetos [Boscarioli et al. 2006], deve-se observar se os arcabouços de programação escolhidos possuem o suporte necessário para as funcionalidades que serão utilizadas.

### **4.3. Ciclo de vida do sistema**

Definidos o SGBD “base” e a ferramenta de conversão para os outros SGBDs (se for o caso), são gerados os scripts DML e DDL e realizadas as conversões necessárias, conforme o sistema é desenvolvido. Eventuais scripts de atualização de versão do sistema precisarão ser escritos para cada SGBD, pois muitas vezes envolvem migração de dados.

O uso de funções SQL diretamente em banco de dados deve ser evitado, pois aumenta a complexidade da solução, exigindo sua manutenção para cada SGBD, bem como a coerência entre as diferentes codificações, inerentes a cada SGBD.

É importante que todos os envolvidos no desenvolvimento do sistema sigam o processo a partir das soluções adotadas, tanto para a conversão entre os diversos SGBDs quanto para o desenvolvimento do próprio sistema na sua linguagem de programação. Esse cuidado garante que todos trabalhem para que a aplicação continue compatível com cada SGBD escolhido.

O teste sistêmico deverá ser realizado com o sistema instalado em cada SGBD, garantindo assim que esteja compatível com todos.

É importante entender que, quanto maior o nível de abstração oferecido pelo arcabouço escolhido e de automação na conversão entre os SGBDs menor o risco de incompatibilidade entre eles. Para um sistema totalmente abstraído em relação ao seu modelo relacional, o teste sistêmico poderá ser realizado por amostragem, sendo que os casos de teste são executados parcialmente em um SGBD e parcialmente em outro, evitando que seja necessário realizar todos os casos de teste em todos os SGBDs, com expressivo ganho de tempo durante seu desenvolvimento.

## **5. Prova de conceito**

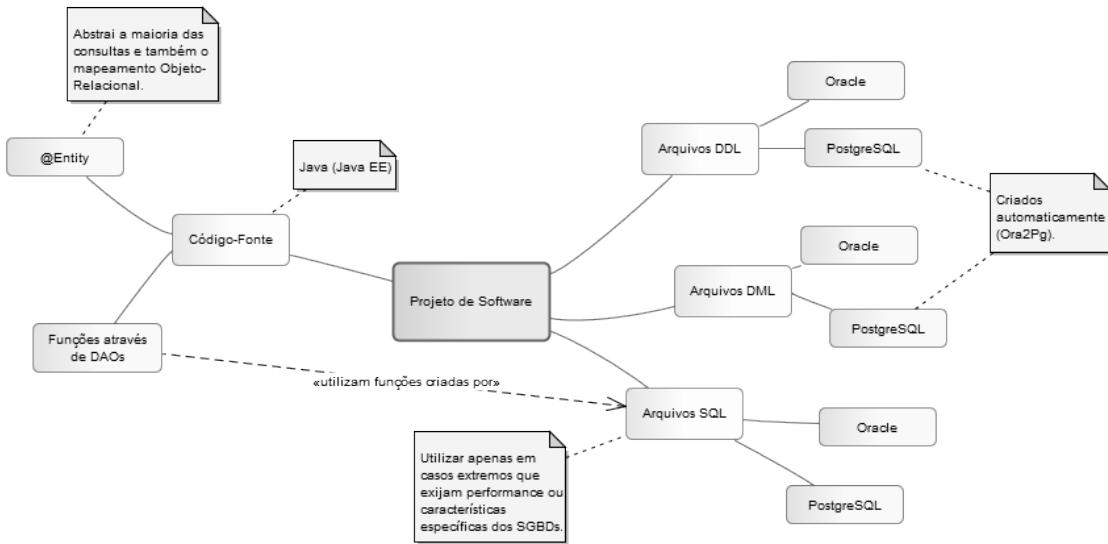
Para a realização da prova de conceito foi considerado um sistema inicialmente construído para o SGBD Oracle, que deveria ser compatível também com o SGBD PostgreSQL. O requisito de “portabilidade” entre ambos SGBDs, durante a vida do sistema e suas evoluções, possibilitou a aplicação da metodologia definida na Seção 4, garantindo a qualidade dos artefatos gerados e a ausência de impacto nos prazos de entrega, eliminando também o risco produzido por conversões manuais realizadas pelos desenvolvedores.

A estrutura adotada, para os artefatos gerados para o projeto é apresentada na Figura 3.

Conforme apresentado na Seção 2, existem arquivos DDL, de instalação da estrutura de tabelas do sistema e arquivos DML, que realizam a população dos dados iniciais necessários para o funcionamento do sistema, bem como os comandos DML presentes em código-fonte, no caso, escrito em Java.

Os comandos DDL, originalmente em Oracle, foram convertidos utilizando-se a ferramenta Ora2Pg [Ora2Pg 2012]. Esses arquivos precisam inicialmente ser instalados em um SGBD Oracle para que a ferramenta consiga extrair as DDLs no formato PostgreSQL. Da mesma forma que as DDLs, as DMLs de carga do sistema foram instaladas no banco de dados Oracle e extraídas com a mesma ferramenta para o formato PostgreSQL.

Os arquivos DDL e DML gerados para o PostgreSQL são colocados em controle



**Figura 3. Estrutura do projeto da prova de conceito**

de versão, sendo disponibilizados para a instalação, juntamente com o produto. Dessa forma o instalador poderá optar por fazer a instalação em um SGBD Oracle ou PostgreSQL, utilizando para isso um único pacote de instalação.

Funções SQL podem ser criadas diretamente em banco de dados e poderão ser utilizadas, a partir do código-fonte Java, através do padrão de projeto DAO (*Data Access Object*) [Alur et al. 2003]. Essas funções devem ser criadas com uma mesma assinatura de chamada, mesmo que para diferentes SGBDs, para que a chamada possa ser única e a construção do DAO independente do SGBD, bem como devem ser construídas para todos os SGBDs definidos.

O sistema foi construído totalmente sobre o arcabouço Java EE EJB e todas as entidades de banco de dados foram mapeadas utilizando-se as funcionalidades de abstração fornecidas por esse arcabouço. Todas as consultas a essas entidades foram construídas utilizando-se a linguagem de consulta JPQL (*Java Persistence Query Language*), que abstrai a linguagem SQL, garantindo que o código-fonte Java seja único e independente do SGBD adotado.

O resultado da prova de conceito é um sistema capaz de trabalhar tanto no SGBD Oracle quanto no SGBD PostgreSQL, com um único pacote de instalação e um único código-fonte, sendo que os processos de conversão entre esses SGBDs são completamente automatizados, validando a metodologia proposta.

## 6. Conclusão

Propor uma solução para a abstração de acesso aos dados de um SGBD, bem como centralizar e automatizar a geração de comandos de definição do modelo relacional, garante a longevidade do sistema desenvolvido. O sistema torna-se independente de SGBD, permitindo a redução dos custos de implantação pela adoção de soluções de software livre sem adaptações do código-fonte original ou mesmo troca estratégica do SGBD. O conhecimento dos desenvolvedores envolvidos na produção do sistema pode ser nivelado, pois não são mais necessários especialistas para cada SGBD, os riscos são mitigados e os

pontos passíveis de erro reduzidos.

Por fim, a automação da geração e do controle dos artefatos de cada SGBD evita erros relacionados à migração de código-fonte específico de determinado SGBD, que são maiores no caso de migração manual, bem como reduz o custo do desenvolvimento por eliminar as horas necessárias para a conversão.

## Referências

- Alur, D., Crupi, J., and Malks, D. (2003). *Core J2EE patterns: best practices and design strategies*. Prentice Hall.
- Barbieri, C. (1994). *Modelagem de Dados*. IBPI Press Rio de Janeiro.
- Bauer, C. and King, G. (2005). *Hibernate in action*. Manning.
- Boscarioli, C., Bezerra, A., Benedicto, M., and Delmiro, G. (2006). Uma reflexão sobre banco de dados orientados a objetos. *IV CONGED*.
- Chamberlin, D., Astrahan, M., Blasgen, M., Gray, J., King, W., Lindsay, B., Lorie, R., Mehl, J., Price, T., Putzolu, F., et al. (1981). A history and evaluation of system r. *Communications of the ACM*, 24(10):632–646.
- Elmasri, R., Navathe, S., Pinheiro, M., Canhette, C., Melo, G., Amadeu, C., and de Oliveira Morais, R. (2005). *Sistemas de banco de dados*. Pearson Addison Wesley.
- Feuerstein, S. (2002). *Oracle pl/sql Programming*. O'Reilly.
- Jacobson, I., Booch, G., and Rumbaugh, J. (1996). *The Unified Modeling Language*. University Video Communications.
- Neto, J. and Passos, E. Jexodus: uma ferramenta para migração de dados independente de sgbd.
- Ora2Pg (2012). Ora2pg software. <http://ora2pg.darold.net/>.
- Patel, R., Brose, G., and Silverman, M. (2006). *Mastering Enterprise JavaBeans 3.0*. Wiley Pub.
- St.Laurent, S., Dumbill, E., and Gruber, E. (2012). *Learning Rails 3*. O'Reilly.
- Viana, E. and Borba, P. (1999). Integrando java com bancos de dados relacionais. *III Simpósio Brasileiro de Linguagens de Programação, Porto Alegre Brasil*, 5.

# Lógica Paraconsistente aplicada como Solução de Inconsistências entre Classificadores de Aprendizagem Simbólica

Luiz Gustavo Moro Senko<sup>1</sup>

<sup>1</sup>Instituto Federal Catarinense – Campus Ibirama  
Rua Getúlio Vargas, 3006 – CEP 89140-000 Ibirama – SC – Brazil

gustavo.senko@ibirama.ifc.edu.br

**Abstract.** This paper presents a technique for handling inconsistencies in symbolic classifiers. Very often, ensemble-based techniques are used to improve performance and classification accuracy. However such methods compromise the knowledge understandability and decisions explanation because the knowledge is partitioned among the classifiers. Furthermore, the classifiers generated from data samples are likely inconsistent, demanding a complex ensemble combination strategy. On the other hand, knowledge integration techniques are used to merge symbolic classifiers into an understandable and global classifier, merging and selecting good classification rules from the local models. In this paper Paraconsistent Logic concepts are used to gather concepts from local models, generating a global ruleset with good accuracy avoiding data exchange and rules evaluations, saving a lot of computational resources. Whenever local models are gathered together the rules are ordered using Paraconsistent Logic operators and other ones are used for classification of new instances. We could observe good results in the experiments performed on a number of benchmark datasets.

**Resumo.** O desenvolvimento de algoritmos e técnicas em mineração distribuída de dados tem como principal interesse a análise de bases de dados fisicamente distribuídas. Essas abordagens produzem melhores soluções em termos de custos e complexidade computacional. Nesse cenário, a manipulação de regras de classificação inconsistentes é uma tarefa importante, pois inconsistências podem comprometer o desempenho dos classificadores. O objetivo deste trabalho foi o desenvolver uma nova metodologia para a integração de conhecimento (classificadores à base de regras) baseado em Lógica Paraconsistente. O método permite tomar decisões confiáveis na ocorrência de regras inconsistentes encontradas em diferentes classificadores. A vantagem do método proposto é evitar o alto fluxo de mensagens e dados entre processadores distribuídos durante a análise das regras geradas de diferentes bases de dados. Isso ocorre porque apenas informação local é utilizada para a geração de um conjunto de regras global. Lógica Paraconsistente é utilizada como um mecanismo de tratamento de incertezas para inferir a classe de um determinado exemplo de teste. Nós pudemos observar nos experimentos realizados sobre diferentes bases de dados que o método pode gerar excelentes resultados.

Keywords: Artificial Intelligence, Knowledge Acquisition, Knowledge based systems, Machine Learning

## 1. INTRODUÇÃO

Muitas pesquisas têm sido desenvolvidas para viabilizar a análise de grandes volumes de dados gerados diariamente em centros de pesquisas, organizações comerciais, industriais, etc. Os métodos desenvolvidos são geralmente baseados em mineração distribuída de dados. Entre os principais métodos desenvolvidos, destacam-se os de mineração distribuída orientada a dados [Freitas, A. and Lavington, S. H. 1998]. Esses métodos têm como princípio a divisão dos dados em subconjuntos menores, que são minerados individualmente, produzindo classificadores locais. Posteriormente, esses classificadores são combinados em um único classificador global. Nesse caso, um dos problemas que podem ocorrer é a existência de dados inconsistentes nos subconjuntos, levando o sistema a gerar classificadores com opiniões contraditórias. Essas inconsistências podem ser geradas pelo método de amostragem escolhido, ou podem ser inerentes às bases de dados distribuídas em diferentes sites (por exemplo, bases de dados de uma rede de lojas). Em mineração distribuída, a ocorrência de dados conflitantes ou inconsistentes compromete significativamente o desempenho dos algoritmos de aprendizagem da máquina. Não é difícil encontrar situações nas quais existam inconsistências explícitas. Por exemplo, considere uma técnica de combinação de classificadores que tem como objetivo unificar a decisão de diversos classificadores gerados a partir de bases de dados distribuídas. Assumindo que uma instância pode pertencer apenas a uma classe, pode haver opiniões contraditórias entre os classificadores se três deles previrem a classe “A”, quatro deles previrem a classe “B” e os últimos três classificadores previrem a classe “C”. Nesse caso, uma técnica como votação simples seria de difícil aplicação e poderia gerar erros de classificação. Os dados armazenados nas bases são considerados precisos se for possível assegurar que representam fielmente os dados do “mundo real”, o que nem sempre ocorre.

Segundo da Costa [da Costa, N. C. A and Abe, J. M. 2000], inconsistências surgem naturalmente no mundo real. Elas podem ocorrer em vários contextos e a automatização de um raciocínio adequado requer o desenvolvimento de teorias formais apropriadas. A existência de informações conflitantes pode estar relacionada a diversas razões, como, por exemplo, à união de bases de dados geradas a partir de diferentes fontes distribuídas, ruído nos dados, atributos com valores faltantes, erro na coleta de dados, entre outras razões, notadamente a falta de variáveis necessárias para a aquisição de conhecimento.

O método proposto neste trabalho utiliza o conhecimento obtido a partir de algoritmos de indução de regras do tipo “SE (condições) ENTÃO classe”, nas quais o antecedente (condições) contém conjunções de condições formadas por pares de atributos e valores e o consequente (classe) contém a classe definida para um conjunto de exemplos que satisfazem as condições do antecedente da regra.

Em mineração distribuída, a união dos modelos obtidos por diversos classificadores pode resultar em um conhecimento global inconsistente, se as regras obtidas em classificadores diferentes são mutuamente exclusivas, mas prevêem a mesma classe. Imagine, por exemplo, que um classificador A possui a seguinte regra: SE idade > 25 ENTÃO classe = homem; enquanto o classificador B possui a seguinte

regra: SE idade < 12 ENTÃO classe = homem. Uma vez que os classificadores foram gerados a partir de dados da mesma natureza, essa contradição não é aceitável. A inconsistência também pode ser gerada em função do consequente das regras. Por exemplo, se um classificador A possui a regra “SE idade > 15 ENTÃO classe = homem” e o classificador B possui a regra “SE idade > 12 ENTÃO classe = mulher”, pode-se dizer que existe uma contradição entre os conceitos das classes homem e mulher, pois um exemplo de teste com “idade = 21” seria classificado ao mesmo tempo como “homem” e “mulher”.

Nesse caso, um critério de ponderação de regra ou classificador deveria ser utilizado para a escolha final. O objetivo deste trabalho foi desenvolver uma metodologia, utilizando conceitos de Lógica Paraconsistente, para auxiliar na obtenção de melhores interpretações sobre conjuntos de regras em que situações como as descritas no parágrafo anterior podem ocorrer, sem, entretanto, prejudicar o desempenho do sistema. A utilização dos formalismos da Lógica Paraconsistente permitiu associar a cada regra fatores evidenciais anotados que representam, respectivamente, o grau de crença (o quanto se acredita que a regra seja verdadeira) e o grau de descrença (o quanto a regra pode ser considerada falsa). Os fatores evidenciais serviram de base para aplicar o critério de decisão na escolha da regra que satisfaça as condições dos exemplos de teste e seja a mais próxima do estado lógico verdade da Lógica Paraconsistente.

## 2. LÓGICA PARACONSISTENTE

Informações contraditórias podem ser importantes no processo de raciocínio e tomada de decisão. Eliminá-las pode causar impacto negativo na solução de determinados problemas. A presença de informações inconsistentes em sistemas computadorizados [Enembreck, F. 1999] pode ser facilmente observada, uma vez que, em diversas aplicações, a inconsistência é inerente ao problema.

De acordo com Enembreck [Enembreck, F. 1999], existem basicamente duas formas para tratar inconsistências: i) atribuir ao algoritmo de aprendizado a habilidade de manipular adequadamente as informações contraditórias durante o processo de aprendizado, com a finalidade de gerar conceitos consistentes e confiáveis; ou ii) aplicar sobre o conhecimento, obtido por meio de um algoritmo de aprendizado qualquer, um método de raciocínio que possibilite a inferência confiável de informações: transformar a base de conhecimento de modo a torná-la consistente, gerar outra base de conhecimento a partir dos dados consistentes existentes na base inicial, ou ainda aplicar técnicas de gerenciamento de incerteza que possibilitem o raciocínio sobre conceitos inconsistentes. Neste trabalho, optou-se por essa última hipótese. Esta escolha possibilita o raciocínio sobre quaisquer conjuntos de regras do tipo SE-ENTÃO, tornando o método independente do algoritmo de aprendizado simbólico utilizado. Além disso, podem-se inferir decisões a partir de conjuntos de regras existentes em diferentes locais, gerados a partir de diferentes bases de dados, mesmo que existam variações de distribuição.

O modelo de inferência proposto está baseado no modelo da Lógica Paraconsistente [da Costa, N. C. A and J. M. Abe 2000] [Blair, H.A. and Subrahmanian V. S. 1998]. A Lógica Paraconsistente, diferentemente da Lógica Clássica, permite representar e realizar inferências sobre informações contraditórias e

também distinguir as situações em que uma determinada proposição é realmente falsa daquelas em que não se tem conhecimento suficiente para se chegar a uma conclusão. Ela permite que informações que são contraditórias  $p$  e  $\neg p$  estejam simultaneamente presentes e fornece mecanismos para o raciocínio sobre informações com essas características. A conclusão obtida pode ser muito útil para a tomada de decisões em que não há informações suficientes, ou elas são contraditórias.

Lógica Evidencial Paraconsistente (LEP) [Subrahmanian, V. S. 1987] é um formalismo desenvolvido que utiliza conceitos de Lógica Paraconsistente.

Os valores-verdade utilizados em LEP são compostos por dois fatores evidenciais pertencentes a um reticulado  $\{x \in \mathfrak{R} \mid 0 \leq x \leq 1\} \times \{x \in \mathfrak{R} \mid 0 \leq x \leq 1\}$ .

A cada item do conhecimento de um sistema, nesse caso, a cada regra em um subconjunto, são associados fatores evidenciais de crença e descrença. O grau de crença representa a força que apóia a verdade da evidência, ao passo que o grau de descrença denota a força associada à falsidade da evidência. Ambos os fatores pertencem ao intervalo  $[0.0, 1.0]$ , ou seja, infinitos valores podem ser associados às premissas pertencentes ao sistema. Portanto, pode ser definido um reticulado infinito  $\tau = \langle |\tau|, \leq \rangle$ , tal que:

$$|\tau| = \{x \in \mathfrak{R} \mid 0 \leq x \leq 1\} \times \{x \in \mathfrak{R} \mid 0 \leq x \leq 1\}$$

O reticulado  $\tau$  que pode ser demonstrado pelo diagrama de Hasse, possui um ponto máximo  $[1.0, 1.0]$  que representa a situação de inconsistência máxima, pois se acredita tanto na verdade quanto na falsidade da premissa, simultaneamente. A verdade de uma premissa é representada pelos fatores evidenciais  $[1.0, 0.0]$ , pois se acredita totalmente na verdade e nada é conhecido sobre a falsidade. Por outro lado, a falsidade é representada por  $[0.0, 1.0]$ , porque se acredita totalmente na falsidade e sobre a verdade da premissa nada é conhecido.

Além de ser possível representar a inconsistência, em LEP, diferentemente da Lógica Clássica, é possível representar o desconhecido ou indeterminado –  $[0.0, 0.0]$ . Nesse caso, não se tem informação nem sobre a verdade nem sobre a falsidade da premissa.

### 3. METODOLOGIA

Estudos realizados [Ladeira, M., Viccari, R. M. 1996] indicam que podem existir várias causas de incertezas em sistemas computacionais e de Inteligência Artificial. Entre elas, está a presença de informações imprecisas e inconsistentes. A inconsistência pode ocorrer, por exemplo, quando informações geradas a partir de fontes distribuídas são unidas para que inferências possam ser realizadas.

Com o objetivo de fornecer um raciocínio mais adequado nessas situações, este trabalho propõe a aplicação dos conceitos de Lógica Evidencial Paraconsistente [da Costa, N.C.A. et al. 1999] [da Costa, N. C. A and Abe, J. M. 2000] [Blair, H.A. and Subrahmanian, V. S. 1998] [Subrahmanian, V. S. 1987] em mineração distribuída [Freitas, A. and Lavington, S. H. 1998]. A primeira etapa para o desenvolvimento do

método baseado na linguagem Paralog\_e [Ávila, B. C. 1996] é a preparação e mineração das bases de dados.

Os diferentes conjuntos de regras são obtidos por meio da aplicação do algoritmo RIPPER [Cohen,W.W. 1995] sobre os diferentes subconjuntos de dados. Utilizou-se a implementação do RIPPER existente no WEKA [Frank, E.]. Esse algoritmo executa a aprendizagem de regras ordenadas e, dessa forma, permite que um exemplo possa ser classificado por mais de uma regra. Ele utiliza critérios de poda incremental para reduzir erros e produzir regras de boa qualidade mesmo em domínios ruidosos.

A segunda etapa consiste na transformação das regras obtidas anteriormente para o formato Paralog\_e [Ávila, B. C. 1996]. Este último utiliza conceitos de programação lógica evidencial e permite inferir a partir de regras anotadas. Cada regra está associada a um grau de crença e outro de descrença.

Dessa forma, todas as regras do tipo SE (condições) ENTÃO (classe) são transformadas em regras Paralog\_e, que possuem o formato *Corpo*← *Cabeça*. A Cabeça representa a conclusão da regra e é formada pela classe e pelos fatores evidenciais de crença e descrença que representam a regra. O Corpo é composto por uma conjunção de condições. A conjunção é denotada pelo símbolo “&”. O conjunto de conjunções forma as condições que compõem a regra. Cada condição da regra é representada por um predicado avaliador que possibilita posteriormente verificar se as condições das regras são verdadeiras quando exemplos de teste são submetidos à inferência. Para ilustrar o procedimento de transformação de regras no formato Paralog\_e, foi criada a seguinte regra no formato RIPPER [Cohen, W. W. 1995] sobre uma base de dados hipotética:

(producaolacrimal = normal) and (astigmatismo = sim) and  
(espectropia = hipermetropia) => class=nenhuma (2.0/1.0)

Ao submeter a regra à transformação, obtém-se a mesma regra no seguinte formato:

```
class('nenhuma')[2.0,1.0] <--  
avaliador(producaolacrimal,V_0)[1.0,0.0] &  
V_0 = normal &  
avaliador(astigmatismo,V_1)[1.0,0.0] &  
V_1 = sim &  
avaliador(espectropia,V_2)[1.0,0.0] &  
V_2 = hipermetropia.
```

É importante observar, nesta etapa, que os fatores evidenciais associados ainda não correspondem ao intervalo [1.0,0.0], pois na etapa seguinte esses valores serão modificados.

Na terceira etapa os valores dos fatores evidenciais são modificados a partir de uma ponderação linear da regra na forma de uma progressão aritmética. Essa atualização é realizada localmente, ou seja, antes da integração dos conjuntos de regras.

O fator evidencial favorável ( $c$ ) (o grau de crença) é atualizado com o valor obtido por meio da ponderação linear do subconjunto de regras. O fator evidencial desfavorável ( $d$ ) (o grau de descrença) é o complemento da ponderação linear no subconjunto de regras ( $d = 1 - c$ ). Para se obter a razão da progressão aritmética, é necessário aplicar a seguinte Equação 1.

$$r = \frac{1}{\text{número\_de\_regras\_no\_subconjunto}} \quad (1)$$

O valor de crença atribuído à  $i$ -ésima regra corresponde à progressão aritmética de  $i$  por  $r$  partindo a última regra para a primeira. A última regra sempre possui  $c = r$  e a primeira regra sempre possui  $c = 1,0$ . Um exemplo desta etapa pode ser visto.

Em Lógica Paraconsistente, os valores dos graus de crença e descrença são independentes entre si, pois não são valores complementares. Uma vez que os valores de crença ( $c$ ) e descrença ( $d$ ) calculados até então são complementares, eles devem ser modificados para se tornarem comparáveis à verdade absoluta, caracterizada pelo par ordenado (1.0,0.0). A partir da crença e descrença, são obtidos  $\text{valor}_1$  e  $\text{valor}_2$  da seguinte forma:

- $\text{valor}_1$  é denominado de Grau de Certeza e é calculado como o grau de crença diminuído do grau de descrença da regra, ( $\text{valor}_1 = c - d$ );
- $\text{valor}_2$  é calculado pela multiplicação do grau de descrença da regra por 2, ( $\text{valor}_2 = 2 \times d$ ).

O grau de certeza  $\text{valor}_1$  mede quanto de certeza está associado à verdade da proposição. Por outro lado, empiricamente foi definido o  $\text{valor}_2$  como o dobro da descrença associada à regra. Acredita-se que toda regra produziria um erro maior (portanto, uma maior descrença) caso fosse avaliada sobre conjuntos de dados diferentes.

Como não há indícios de quanto seria essa propagação, o dobro da descrença foi utilizado empiricamente. Após o cálculo dos fatores evidenciais, os subconjuntos de regras são unidos em um único conjunto. Nesta fase, as regras são ordenadas de acordo com a menor distância euclidiana – conforme Equação 2 – em relação aos fatores evidenciais (1.0,0.0) que representam a verdade. onde  $x_i$  é o par ordenado formado por ( $\text{valor}_1$ ,  $\text{valor}_2$ ) de uma regra e  $x_j$  é (1.0,0.0).

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2} \quad (2)$$

Considera-se que quanto menor a distância obtida entre os valores ( $\text{valor}_1$ ,  $\text{valor}_2$ ), que denotam os fatores evidenciais da regra e o estado lógico que representa a verdade (1.0,0.0), mais próxima a regra está da verdade em termos quantitativos.

Na última etapa, cada exemplo de teste é transformado em um conjunto de fatos submetidos à prova a partir da base de conhecimento formada pelo conjunto de regras. A representação das regras em cláusulas Horn [Casanova, M.A; Giorno, F.A.C.; Furtado, A.L.F. 1987] permite verificar se as condições das regras são verdadeiras em relação ao exemplo de teste fornecido. A linguagem Paralog\_e trabalha de maneira semelhante à linguagem Prolog, utilizando um procedimento de Resolução SLD [Casanova, M.A; Giorno, F.A.C.; Furtado, A.L.F. 1987] para encontrar a verdade ou falsidade de cada condição de uma regra.

Em seguida, é feito um questionamento  $Q$  para cada classe pertencente ao domínio da base de dados. As respostas são as evidências obtidas para cada classe. Como um questionamento é feito para cada classe pertencente ao domínio, o critério de decisão adotado elege a regra capaz de cobrir o exemplo cujos fatores evidenciais possuem a menor distância euclidiana em relação ao estado lógico da verdade — par ordenado (1.0,0.0). Para avaliar o método baseado na linguagem Paralog\_e, a classe eleita é comparada à classe que rotula o exemplo de teste. Se a classe eleita for a mesma do exemplo, houve acerto na classificação.

#### 4. EXPERIMENTOS, RESULTADOS E DISCUSSÕES

Nos experimentos, foram utilizadas dez bases de dados públicas, pertencentes ao diretório de bases para aprendizagem de máquina e mineração de dados, mantido pela Universidade da Califórnia [Blake, C.L.; Newman, D.J.; Hettich, S.; Merz, C.J. 1998].

Cada base utilizada nos experimentos foi dividida aleatoriamente para compor o conjunto de treinamento e testes. O conjunto de treinamento possui 80% dos exemplos existentes na base e o conjunto de teste contém os 20% restantes. O conjunto de treinamento foi dividido em 10 partições de treinamento na forma de amostras de 10% com recobrimento. Portanto, cada conjunto de regras foi gerado em partições de cerca de  $(0,8 \times 0,1) \times 100 = 8\%$  de todos os dados originais. Esse percentual foi escolhido para que a ocorrência de conflitos seja favorecida, uma vez que o algoritmo de aprendizagem utiliza um subespaço reduzido do conjunto de tuplas. Cada classificador gerado em uma partição é avaliado sobre os 20% dos exemplos de teste separados anteriormente. Esse procedimento foi repetido iterativamente 10 vezes para cada base de dados.

As características de cada base utilizada, bem como o número de exemplos presentes e a existência ou não de valores faltantes nos exemplos que compõem a base, entre outras características, foram detalhadas na Tabela I.

É importante considerar que, segundo Freitas [Freitas, A. and Lavington, S. H. 1998], em algoritmos de indução o produto cartesiano das variáveis e o número de

valores que estes atributos podem assumir aumentam exponencialmente o tamanho do espaço de tuplas e consequentemente o espaço de regras a ser pesquisado.

**Tabela I. Característica das bases utilizadas**

	Base de Dados	#Número de Exemplos	Valores Faltantes	#Total de Atributos	#Atributos Nominais	#Atributos Numéricos	#Classes	Distribuição de Classes
1	Zoo	101	não	17	16	1	7	desbalanceada
2	Audiology	226	sim	69	69	–	24	desbalanceada
3	Monk 1	432	não	6	6	–	2	desbalanceada
4	Monk 2	432	não	6	6	–	2	desbalanceada
5	Soybean	683	sim	35	35	–	19	desbalanceada
6	Vehicle	846	não	18	–	18	4	desbalanceada
7	Tic-Tac-Toe	958	não	9	9	–	2	desbalanceada
8	Vowel	990	não	13	3	10	11	balanceada
9	Car	1728	não	6	6	–	4	desbalanceada
10	Segment	2310	não	19	–	19	7	balanceada

Apesar disso, o número de atributos, bem como seus possíveis valores, contribuiu para gerar conjuntos de regras mais expressivos, gerando possivelmente regras formadas por um maior número de condições. Dessa forma, tais regras possivelmente representam melhor as características da base, cobrindo regiões específicas do espaço de tuplas e gerando menos interseções entre as regras. Quando os subconjuntos de regras são considerados individualmente, as interseções e inconsistências são ignoradas, gerando muitos erros de classificação. Por outro lado, o método proposto é capaz de identificar essa situação e selecionar a regra mais adequada para classificar o exemplo.

Deve-se notar que, à medida que o número de atributos diminui, o espaço de tuplas é drasticamente reduzido. As bases Tic-Tac-Toe e Car são densamente populadas (em termos do número de exemplos). Dessa forma, a amostragem dos dados não tem um forte impacto no desempenho dos classificadores simbólicos locais. Por outro lado, quando esses classificadores são unidos, o sistema não é capaz de diferenciar a melhor regra, gerando erros de classificação.

Acredita-se, portanto, que a técnica apresentada neste trabalho é indicada para mineração de conjuntos de dados distribuídos que representam visões parciais do espaço de *tuplas*.

O objetivo do método proposto foi tratar possíveis inconsistências provocadas pela união dos N subconjuntos de regras formados a partir da divisão de dados e evitar o fluxo de comunicação entre os processadores, reduzindo o tempo de processamento

e, principalmente, garantindo uma taxa de acerto aceitável. A utilização dos conceitos de Lógica Paraconsistente permitiu atribuir às regras fatores evidenciais de crença e descrença, quantificando o grau de importância da regra em relação aos subconjuntos. Com a ordenação do conjunto de regras, foi possível mensurar as regras em relação ao estado lógico que representa a verdade e determinar um critério de decisão na escolha entre as regras candidatas.

**Tabela II. Comparação de resultados entre o método Paralog\_e e o algoritmo RIPPER**

Base de Dados	Média geral no Paralog_e (%)	Média Geral no Ripper (%)	Diferença Percentual entre os Métodos	Relação entre #Atributos e # Total de Exemplos
Audiology	<b>48,69 ± 2,94</b>	32,65 ± 3,88	1,49	0,305
Zoo	<b>61,43 ± 4,73</b>	49,14 ± 4,14	1,25	0,168
Soybean	<b>63,06 ± 7,59</b>	51,21 ± 2,32	1,23	0,051
Vowel	<b>43,89 ± 4,35</b>	33,29 ± 1,14	1,32	0,013
Segment	<b>87,87 ± 3,83</b>	82,83 ± 1,43	1,06	0,008
Monk2	<b>63,70 ±10,83</b>	60,48 ± 2,25	1,05	0,014
Vehicle	<b>52,35 ± 4,12</b>	52,24 ± 2,18	1,00	0,021
Monk1	55,23 ± 5,66	<b>55,50 ± 4,25</b>	1,00	0,014
Tic-Tac-Toe	65,22 ± 0,94	<b>67,76 ± 2,27</b>	0,96	0,009
Car	58,24 ± 4,45	<b>71,98 ± 1,36</b>	0,81	0,003

Na análise dos resultados, foi possível identificar que a relação entre o número de atributos que compõem a base pode ter impacto significativo no desempenho do método. Pode-se observar que o método apresenta um melhor desempenho quando a relação entre número de atributos e número de exemplos é elevada (o espaço de tuplas é esparsa), pois os modelos locais gerados cobrem regiões distintas do espaço de tuplas.

## 5. REFERÊNCIAS

Freitas, A. and Lavington, S. H. Approaches to Speed Up Data Mining.Mining Very Large Databases with Parallel Processing. ISBN 0-79238048-7.Pages 89-108. Kluwer Academic Publishers, The Netherlands,1998.

da Costa, N. C. A and J. M. Abe. Paraconsistência em Informática e Inteligência Artificial. Revista Estudos Avançados n. 14, vol. 39 – USP. Estudos Avançados n. 14, vol. 39 – USP. São Paulo, maio/agosto 2000.

Enembreck, F. Um Sistema Paraconsistente para Verificação Automática de Assinaturas Manuscritas. Dissertação de Mestrado - PPGIA – PUCPR. Curitiba, 1999.

Blair, H.A. and Subrahmanian, V. S. Paraconsistent Foundations for Logic Programming. *Journal of Non-Classical Logic*, 5, 2, pag. 45-73, 1988

Subrahmanian, V. S. On the Semantics of Quantitative Logic Programs. Proceedings of 4th IEEE Symposium on Logic Programming, San Francisco, September, pp. 173-182, 1987.

Ladeira, M.; Viccari, R. M. Representação do Conhecimento Incerto. XIII Symposium on Artificial Intelligence SBIA'96, Curitiba, October, 1996.

da Costa, N.C.A. et al., N. A. Lógica Paraconsistente Aplicada. Atlas, São Paulo, 1999.

Subrahmanian, V. S. On the Semantics of Quantitative Logic Programs. Proceedings of 4th IEEE Symposium on Logic Programming, San Francisco, September, pp. 173-182, 1987.

Ávila, B. C. Uma Abordagem Paraconsistente Baseada em Lógica Evidencial para Tratar de Exceções em Sistemas de Frames com Múltipla Herança. Tese de Doutorado, Escola Politécnica da Universidade de São Paulo. São Paulo, 1996.

Cohen, W. W. Fast Effective Rule Induction. In Proceedings of the 12th Int. Conference in Machine Learning (ICML '95). Pages 115-123. 1995.

Casanova, M.A; Giorno, F.A.C.; Furtado, A.L.F. Programação em Lógica e a Linguagem Prolog. Ed. Edgard Blücher Ltda. São Paulo, 1987.

Blake, C.L.; Newman, D.J.; Hettich, S.; Merz, C.J. UCI Repository of machine learning databases. Available on:

[<http://www.ics.uci.edu/~mlearn/MLRepository.html>]. University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

Frank, E. *WEKA Machine Learning Software*.

[<http://www.cs.waikato.ac.nz/ml/weka>]

# **Um Método para Indexação de Formulários Web visando Consultas por Similaridade<sup>1</sup>**

**Willian Ventura Koerich, Ronaldo dos Santos Mello**

Centro Tecnológico (CTC) - Departamento de Informática e Estatística (INE)  
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC - Brasil

{willian.vkoerich, ronaldo}@inf.ufsc.br

**Abstract:** *Search engines do not support specific searches for web forms found on Deep Web. Within this context, the WF-Sim project proposes a query-by-similarity system for Web Forms to deal with this lack. This paper presents an indexing technique for querying-by-similarity web forms as a WF-Sim system component. This technique is centered on suitable index structures to the main kinds of queries applied to web forms, as well as some optimizations in these structures to reduce the number of index entries. To evaluate the indexes' performance, we ran experiments on two persistence strategies: file system and database. The performance of accessing the database was higher. We also compare the performance of our indexes with the traditional keyword-based index, and the results were also satisfactory.*

**Resumo:** *Motores de busca atuais não possuem suporte à buscas específicas por formulários web relacionados à Deep Web. Neste contexto, o projeto WF-Sim propõe um processador de consultas por similaridade para formulários Web para lidar com esta limitação. Este artigo apresenta uma técnica de indexação para buscas por similaridade em formulários web, atuando como um componente do sistema WF-Sim. Esta técnica está centrada em estruturas de índice adequadas aos principais tipos de consulta aplicados a formulários web, bem como otimizações nestas estruturas para reduzir a quantidade de entradas no índice. Experimentos preliminares sobre duas estratégias de persistência de dados suportados pelo WF-Sim foram realizados: sistema de arquivos e banco de dados. O desempenho de acesso ao banco de dados foi superior. Comparou-se também o desempenho dos índices propostos contra o tradicional índice de palavras-chave, e o resultado também foi satisfatório.*

## **1. Introdução**

Uma grande quantidade de serviços está atualmente à disposição das pessoas através da Web, como locação e vendas de veículos, reserva de hotéis, compra de livros, oferta de empregos, etc. Esses serviços disponibilizam diversos dados para consultas aos usuários como por exemplo, veículos de diversos fabricantes e modelos, no caso de um web site de uma concessionária. O acesso a esses bancos de dados é possível através de formulários existentes em páginas Web. Estes formulários exibem atributos do banco de dados sobre os quais o cliente especifica filtros e então submete consultas. Estes bancos de dados na Web são denominados banco de dados escondidos (*hidden databases* ou

---

<sup>1</sup> Este trabalho é parcialmente financiado pelo CNPq através do projeto WF-Sim (Nro. processo:481569/2010-3).

*deep-Web*)<sup>2</sup>, uma vez que a sua estrutura e o seu conteúdo não estão completamente visíveis ao usuário. Somente alguns atributos (e alguns eventuais valores que permitem a definição de filtros) estão visíveis nos formulários [Madhavan et al. 2009].

O projeto WF-Sim visa desenvolver uma solução para esta problemática: um processador de consultas por similaridade para formulários Web [Gonçalves 2011]. Este processador caracteriza-se por ser um software responsável pela execução de todas as tarefas necessárias à geração de um resultado adequado a uma consulta por similaridade, como especificação de uma consulta por parte do usuário e métricas adequadas para definição do grau de similaridade entre os formulários. Este projeto propõe ainda um método de busca por campos dos formulários web, internamente chamados de *elementos* de formulários. A Figura 1 mostra um exemplo de formulário web no domínio de veículos. Cada atributo (elemento) de um formulário geralmente possui um rótulo e uma série de valores possíveis associados a ele, como por exemplo, “*Make*” e “*Model*”.

Buscas no WF-Sim ocorrem sobre elementos de formulários indexados. As estruturas de índice são definidas a partir de clusters gerados por um processo de *matching* de elementos de formulários, permitindo recuperação de formulários com elementos similares. Estas estruturas de índice foram devidamente projetadas para facilitar as consultas típicas por formulários web.

**Figura 1. Exemplo de Formulário Web**

Este artigo apresenta a estratégia de indexação por similaridade para formulários web desenvolvida para o WF-Sim, visando o acesso a dados mantidos em dois tipos de mecanismos de persistência: arquivos e banco de dados. Avalia-se aqui não apenas o desempenho das estruturas de índice para cada tipo de persistência, mas também quais estruturas de índice apresentaram melhor desempenho.

As demais seções detalham o desenvolvimento deste trabalho. A seção 2 aborda o projeto WF-Sim, com foco na atividade de indexação. A seção 3 detalha o módulo de indexação e as estruturas de índice propostas. A seção 4 descreve os experimentos realizados e a seção 5 é dedicada à conclusão.

---

<sup>2</sup><http://brightplanet.com/wp-content/uploads/2012/03/12550176481-deepwebwhitepaper1.pdf>

## 2. WF-Sim

As técnicas de busca por formulários web atualmente utilizadas geralmente são baseadas no modelo de busca por palavra chave, que executa o *matching* entre a entrada com termos existentes nos formulários. O principal exemplo é a máquina de busca *DeepPeep*<sup>3</sup>. Visando adicionar capacidade de busca por similaridade a formulários web, o projeto WF-Sim, desenvolvido na Universidade Federal de Santa Catarina pelo grupo de Banco de dados (GDB/UFSC)<sup>4</sup>, com financiamento do CNPq, se inspira no fato de que os dados disponíveis na *Deep Web* são relevantes para usuários que desejam encontrar formulários web que satisfaçam suas necessidades em termos de serviços online para os mais diversos fins.

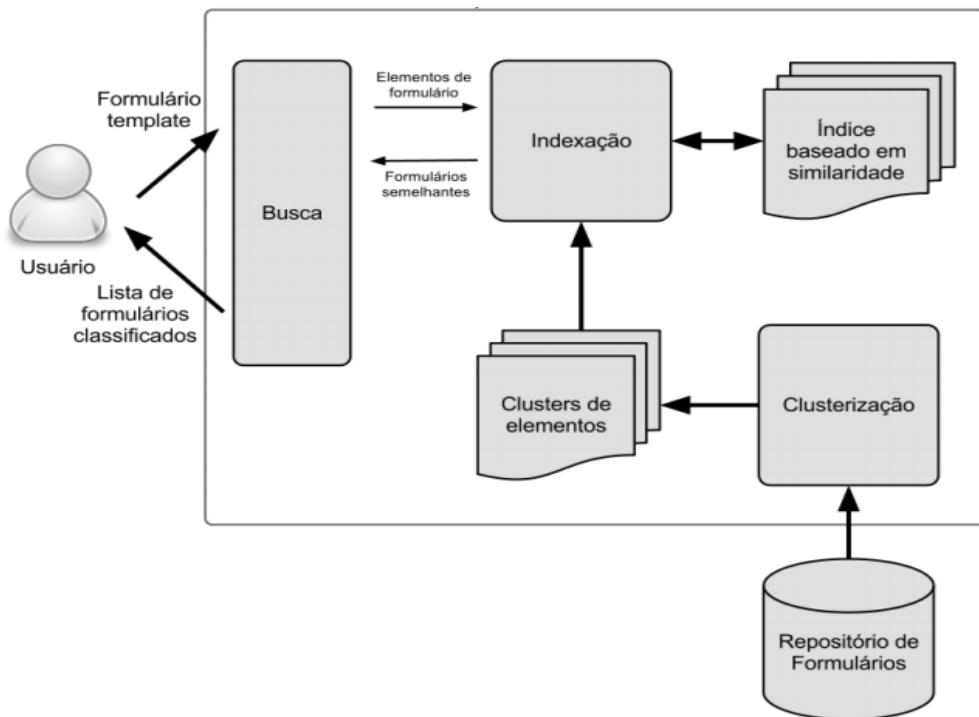


Figura 2. Arquitetura do Sistema WF-Sim

O WF-Sim é um processador de busca por similaridade para formulários web baseados nos seus elementos. A Figura 2 mostra a arquitetura do sistema. Com base em uma consulta de entrada, no caso, um conjunto de elementos denominado *formulário template*, o sistema retorna um conjunto de formulários web que possuem elementos similares. O sistema possui os módulos de busca, indexação e clusterização. O componente de clusterização foi alvo de um trabalho anterior [Silva & Mello 2012], estando, assim, fora do escopo deste artigo. Neste componente são aplicadas métricas de similaridade de modo a agrupar os formulários em clusters com elementos similares.

O módulo de indexação é o foco deste artigo e visa criar uma estrutura de índice, garantir o acesso a estas estruturas e a recuperar os formulários relevantes. Dado um *formulário template* de entrada, a busca por elementos similares acessa um dicionário de sinônimos que direciona elementos do *template* a elementos sinônimos ditos elementos centroides, ou seja, elementos representativos de um cluster de elementos

<sup>3</sup><http://www.deeppeep.org/>

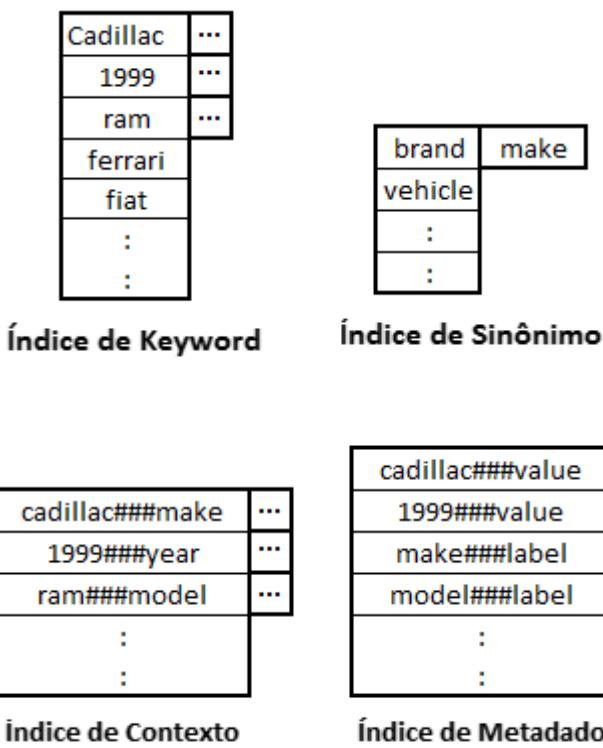
<sup>4</sup><http://www.gbd.inf.ufsc.br>

similares. Um exemplo seria um cluster formado pelos elementos “*Make*”, “*Brand*”, “*Select a Make*”, sendo “*Make*” eleito como centróide. A próxima seção descreve o módulo de indexação.

### 3. Módulo de Indexação

Três estruturas de índice foram definidas para o acesso a dados de formulários Web: *palavra-chave*, *contexto* e *metadado*. A idéia destas estruturas foi obtida de um trabalho anterior do GBD/UFSC [Mello et. al. 2010] e adaptada à problemática de busca por similaridade em formulários Web. Estas estruturas, em particular os índices de contexto e metadado, são adequadas aos tipos de consulta mais frequentes sobre formulários web.

Consultas por contexto associam um rótulo aos seus respectivos valores presentes em um formulário. A idéia é recuperar formulários com base na contextualização de campos por domínio, ou seja, permite que o usuário recupere formulários com determinado tipo de conteúdo de campo que lhe interessa. Já consultas por metadado permitem indicar se um termo de interesse é um metadado do tipo rótulo ou valor.



**Figura 3. Estrutura dos Índices Propostos**

A Figura 3 mostra exemplos dos tipos de índice desenvolvidos. O índice de palavra-chave (*keyword*) possui entradas tradicionais de termos para cada possível valor ou rótulo existente. O índice de contexto define entradas de índice para as combinações de rótulos com seus respectivos valores presentes em elementos, no formato *nome\_valor###nome\_rótulo*. O índice de metadado classifica o tipo de informação que se deseja recuperar(rótulo ou valor), com entradas no formato *nome\_valor###VALUE* ou *nome\_valor###LABEL*. Priorizou-se os termos cuja variação de conteúdo é maior (nomes de valores) no início da entrada do índice, para tornar a ordenação do índice mais adequada para fins de busca. Esta ordenação é relevante principalmente para

buscas por desigualdade ou por *range*, que requerem uma navegação seqüencial em parte da estrutura do índice.

O índice de sinônimos é uma estrutura auxiliar ao funcionamento dos índices de contexto, metadado e palavra-chave. No momento da indexação de um rótulo de um elemento de formulário Web, é feita uma consulta a um banco de dados de sinônimos para verificar se o rótulo é um sinônimo de um centróide. Em caso positivo, é adicionado ao índice de sinônimos uma entrada com o rótulo em questão e sua associação para o centróide sinônimo que está indexado nas outras estruturas de índice. Desta forma, quando o rótulo informado no *template* não estiver explicitamente indexado nas estruturas propostas, mas for um sinônimo de um centróide, é possível encontrar formulários similares através da procura por sinônimos, garantindo, assim, uma busca por similaridade. O índice de sinônimo é fundamental na estratégia de indexação proposta, visto permitir que os índices de palavra-chave, contexto e metadado indexem apenas os rótulos dos elementos centróides de cada cluster, viabilizando, estruturas de índice com número reduzido de entradas.

### 3.1 Limpeza de Dados

O método de indexação é responsável também pela execução de procedimentos de limpeza nos dados para melhorar a indexação dos mesmos. Para tanto, esse módulo possui um analisador que faz a uniformização dos termos, passando as letras para o formato minúsculo e removendo variações indesejadas através do processo de *stemming* e remoção de *stop words*. Esses procedimentos garantem que campos de formulários representando uma mesma informação, mas com grafias diferentes, possam ser indexados de maneira uniforme. A remoção de *stop words* reduz o tamanho dos índices, pois evita a criação de entradas nos índices para termos existentes nos rótulos que não sejam relevantes para consultas. O processo de *stemming* também reduz a quantidade de entradas, pois apenas o radical dos termos dos elementos desejáveis dos rótulos é indexado. Um exemplo é mostrado na Figura 4. Um elemento com rótulo “Do ano” e outro elemento com rótulo “Ano” sem o processo de limpeza seriam indexados em posições diferentes no índice. Com a inclusão do analisador, os campos são indexados uma única vez, com letras minúsculas sem a *stop word* “Do”.

<b>Do Ano:</b> 1999	<b>Ano:</b> 1999
------------------------	---------------------

**Figura 4. Exemplo de rótulos extraídos de formulários**

O analisador também é utilizado no processo de busca por formulários web. Neste caso, os rótulos dos elementos do *template* passam igualmente por um processo de limpeza. Após, os termos resultantes são verificados nos índices.

### 3.2 Acesso aos Índices

O acesso a formulários web relevantes através dos índices se baseia no tradicional modelo booleano de recuperação de informação [Yates & Neto 1999]. Esse modelo possibilita a definição de filtros utilizando os operadores lógicos AND, OR e NOT.

Para ilustrar a sua utilização, suponha que um usuário necessita encontrar formulários que possuam veículos da marca (*brand*) GM e rótulo ano (*year*), ou seja, um formulário *template* com esses 2 elementos<sup>5</sup>. O módulo de Busca gera então a seguinte consulta: *GM###brand AND year###LABEL*. Cada um dos filtros gerados é então passado para o módulo de Indexação. Considerando o filtro “*GM###brand*”, o módulo de Indexação o caracteriza como sendo um filtro de contexto (formado por 2 termos – rótulo e um possível valor para ele), verifica a necessidade de limpeza de dados para o rótulo e então acessa o índice de sinônimos. Supondo que o termo “*brand*” tenha apenas um (1) centróide como sinônimo (“*make*”, por exemplo), o filtro é convertido para “*GM###make*” e a entrada correspondente a este filtro e então acessada no índice de contexto. Caso haja mais de um termo sinônimo para “*brand*”, as entradas do índice correspondentes a todos os sinônimos são acessadas e é feita uma união das URLs dos formulários web presentes em cada entrada. O mesmo raciocínio se aplica ao filtro “*year##LABEL*” e o conjunto de formulários web resultante de cada filtro retornado pelo módulo de Indexação é processado posteriormente pelo módulo de Busca conforme os operadores lógicos definidos na consulta.

### 3.3. Implementação

A ferramenta Lucene<sup>6</sup> foi utilizada para a implementação das estruturas de indexação propostas [Hatcher & Gospodnetic 2005]. Lucene é uma biblioteca de software para a recuperação de informação, sendo responsável pela indexação e da informação indexada. Essa ferramenta possibilita a criação de índices invertidos, abordagem típica para recuperação de informação na web [Yates & Neto 1999], [Elmasri & Shamkant 2011]. O índice invertido é uma estrutura de dados que mapeia um conteúdo para os documentos que o contém.

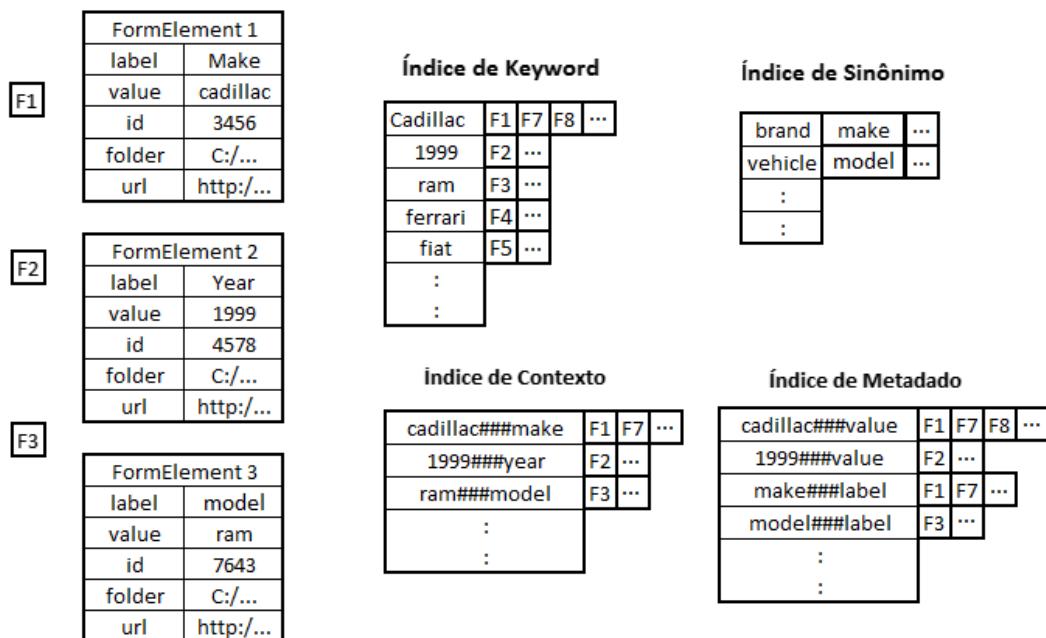


Figura 5. Informações sobre elementos considerados nos índices

<sup>5</sup> Maiores detalhes sobre como o módulo de busca processa templates e gera filtros de consulta para o módulo de indexação estão fora do escopo deste artigo.

<sup>6</sup> <http://lucene.apache.org/core/>

No contexto deste trabalho, os índices invertidos mapeiam um termo, como por exemplo, um valor de um campo, para os formulários que contenham esse determinado termo. Como pode ser visto na Figura 5, o WF-Sim persiste um elemento de formulário com as seguintes propriedades: rótulo, valores que lhe podem ser atribuídos, endereço do formulário web original (URL) e um identificador junto ao banco de dados. Neste caso, os índices definidos no Lucene apontam para a localização destes elementos persistidos em arquivos ou banco de dados.

Para a criação de um banco de dados de sinônimos foi adotada a ferramenta *WordNet*<sup>7</sup> é um banco de dados léxico bastante utilizado como um dicionário de suporte para análise de textos e aplicações envolvendo inteligência artificial. O *WordNet* agrupa palavras da língua inglesa em conjuntos de sinônimos e outros tipos de relações semânticas, além de fornecer definições gerais das mesmas.

Nesse trabalho foi utilizada a biblioteca *Java WordNet Interface (JWI)*<sup>8</sup>, que fornece acesso ao banco de dados de informações do *WordNet*. Sua utilização foi necessária para determinar a similaridade entre elementos e construir o índice de sinônimos.

#### 4. Experimentos

A implementação do módulo de indexação foi avaliada através de experimentos preliminares com o objetivo de medir o desempenho do processamento de filtros de consulta acessando as estruturas de índice desenvolvidas. Foram testados índices para elementos de formulários persistidos em arquivos de dados serializados em disco e elementos de formulários persistidos em um banco de dados relacional.

Para os experimentos foi utilizado um computador equipado com um processador Athlon II X2 de 2.9 GHz, memória de 4 GB, disco rígido de 500 GB, sistema operacional Windows 7 Home Basic SP1 de 64 bits e banco de dados MySQL. A amostra de formulários Web disponível para teste compreende 1090 formulários que possuem na totalidade 6157 elementos.

A Figura 6 mostra a média geral de tempo de execução de um mesmo conjunto de filtros de consulta, específico para cada tipo de índice, acessando arquivos em disco e o banco de dados. Já a Figura 7 mostra resultados de experimentos com uma quantidade incremental de filtros no domínio de veículos, ou seja, sobre um, dois e cinco elementos, visando o acesso ao índice de contexto. O índice de contexto é o índice mais acessado na prática, pois indexa filtros de consulta mais usuais no contexto de formulários web.

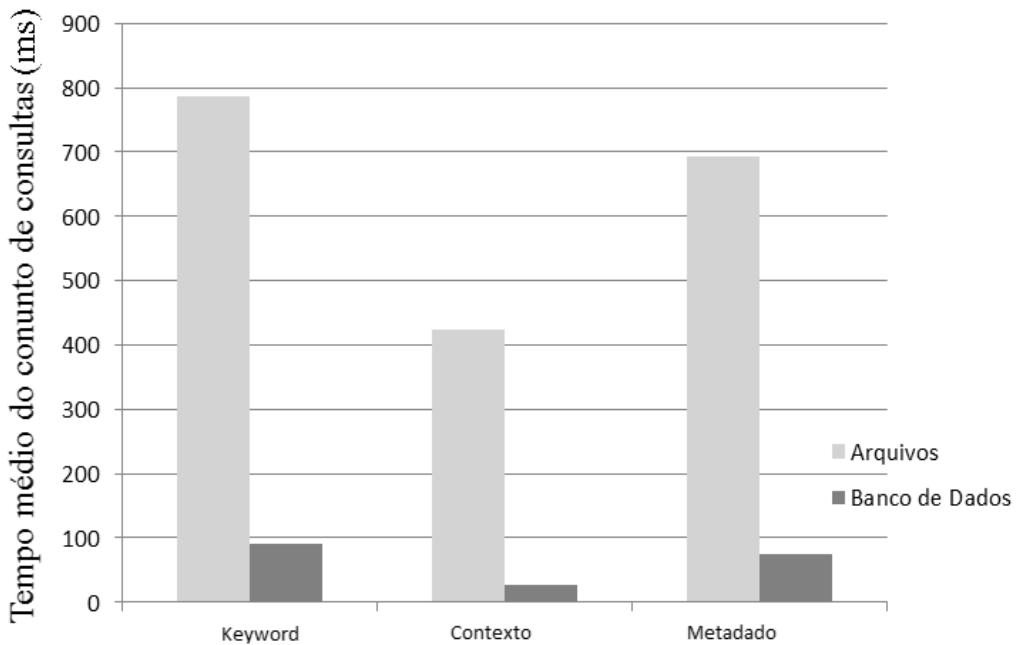
Os testes mostrados na Figura 6 registraram uma grande diferença de desempenho de acesso ao banco de dados frente aos arquivos serializados, ou seja, as consultas gastaram muito mais tempo (8x mais lento, em média) no acesso aos arquivos. Essa superioridade se deve ao fato de que o acesso a arquivos em disco não conta com as otimizações características dos sistemas gerenciadores de banco de dados. Além disso, não se considerou testes com os dados totalmente na *cache* do MySQL, pois a intenção

---

<sup>7</sup><http://wordnet.princeton.edu/>

<sup>8</sup><http://projects.csail.mit.edu/jwi/>

é lidar futuramente um volume bem maior de dados (o volume de dados da *Deep Web* é realmente muito grande!) e isso seria inviável de ser mantido na íntegra em uma *cache*.



**Figura 6. Média dos tempos de processamento dos filtros de consulta para um conjunto de testes**

Já com relação aos testes mostrados na Figura 7, percebe-se, para o acesso ao banco de dados, que o tempo de processamento dobrou a cada variação de quantidade de filtros conjuntivos, enquanto que o aumento de tempo foi bem menor no caso do acesso aos arquivos, inclusive com redução na passagem de 2 para 5 filtros. Essa situação precisa ser melhor avaliada com uma bateria mais ampla de testes sobre volumes maiores de dados. Mesmo assim, a diferença de tempo no acesso a arquivos e banco de dados continuou sendo bastante acentuada, sendo o acesso ao banco de dados de 10x a 30x (aproximadamente) mais rápido para filtros conjuntivos.

CONTEXT	Qtde	Arquivos Tempo	Banco de Dados Tempo
jeep###make	355	381 ms	11 ms
corolla###model	155	118 ms	10 ms
jeep###make AND corolla###model	86	486 ms	19 ms
jeep###make OR corolla###model	424	484 ms	21 ms
jeep###make AND NOT corolla###model	269	484 ms	20 ms
acura###make AND audi###make AND chevrolet###make AND 1000###price AND maverick###vehicle"	0	459 ms	36 ms
acura###make OR audi###make OR bentley###make OR 1964###year OR 1999###year	464	490 ms	49 ms
acura###make OR audi###make AND bentley###make OR 1964###year AND NOT 1999###year	355	488 ms	53 ms

**Figura 7. Conjunto de filtros de consulta submetidos ao índice de contexto**

Uma importante contribuição deste trabalho, no contexto de consultas sobre formulários web, foram os menores tempos de busca para as consultas sobre os índices de contexto e metadados, se comparados com os tradicionais índices por palavras-chave. Este melhor desempenho está associado ao fato de que os dados de elementos passam, durante a construção destes índices, por um processo que gera automaticamente filtros por contexto e por metadado que ficam armazenados diretamente nestes índices. Esses tipos de estruturas são muito adequadas para consultas posteriores sobre formulários Web visto que garantem um mapeamento direto do filtro para uma entrada nestes índices. O uso do índice de sinônimo, visando garantir buscas por similaridade, contribuiu para uma redução no número de entradas nesses dois índices, uma vez que somente um dos termos de um conjunto de sinônimos é indexado, permanecendo os demais apenas no índice de sinônimos. Desta forma, diminuiu-se o tamanho das estruturas e o *overhead* de acesso.

## 5. Conclusão

Este trabalho apresenta e valida uma estratégia de indexação visando buscas por similaridade para dados de formulários Web no contexto do projeto WF-Sim. Esta estratégia introduz mecanismos de refinamento para o contexto de formulários web. Esses mecanismos consideram a indexação de informações sobre elementos de formulários em estruturas de índice especificadas por contexto, metadado, além da tradicional busca por palavra-chave. As duas primeiras estruturas garantem otimizações para o módulo de busca uma vez que as estruturas já indexam os filtros de consultas mais freqüentes para formulários. Buscas tradicionais considerando apenas palavras-chave teriam que, no caso de uma busca por contexto, por exemplo, recuperar informações primeiro sobre o rótulo, depois sobre o valor desejado, e após computar uma intersecção dos formulários recuperados. Este *overhead* é desnecessário com a introdução do índice de contexto, e o mesmo vale para o índice de metadado.

O trabalho de [Mello et. al. 2010] foi a base escolhida para a construção das estruturas de índice aqui apresentadas. [Mello et. al. 2010] define índices de contexto e de metadado. Entretanto, o escopo do trabalho não é específico para o contexto de buscas por similaridade em formulários web, que é o objetivo do projeto WF-Sim. Este artigo aplica e estende essas idéias para o propósito do projeto. Nenhum outro trabalho relacionado na literatura se propõe a definir estruturas de índice para buscas por similaridade sobre formulários web.

O próximo passo no contexto deste trabalho é avaliar o desempenho do módulo de indexação com um repositório maior de formulários web. A amostra de testes do projeto conta com um número aproximado de 1090 formulários. Essa base é composta de dados públicos de formulários oriundos de *sites* de serviços diversos. Eles foram coletados para a utilização no projeto WF-Sim, não estando disponível ainda ao público. Futuramente, com a disponibilização do WF-Sim como uma aplicação web, esses dados serão passíveis de consulta. Através de uma parceria do GBD/UFSC com a Universidade de Utah, uma amostra de aproximadamente 40000 formulários está sendo disponibilizada para novos experimentos. Uma vez que a natureza dos dados coletados nessas fontes de dados é da língua inglesa, o *WordNet* cumpre seu papel de maneira satisfatória. Entretanto, levando em consideração futuras adições de dados na língua portuguesa (formulários web de sites nacionais), será necessário utilizar dicionários de suporte para o Português.

Outra atividade futura é considerar consultas que testam dependências entre elementos de formulários Web, como por exemplo, um campo “*Make*” cujos valores determinam os valores de um campo “*Model*”, supondo um domínio de veículos. A intenção é considerar filtros que testem a existência de tais dependências e definir estruturas de indexação que facilitem buscas por similaridade neste contexto. Percebe-se que este tipo de consulta é relevante para casos em que o usuário deseja acessar formulários que implementam automaticamente uma cadeia de dependências entre campos, facilitando, assim, a sua intenção de busca por alguma informação.

## Referências

- Baeza-Yates, R.; Ribeiro-Neto, R. Modern Information Retrieval. (1999). ACM Press / Addison-Wesley.
- Elmasri, R.; Shamkant B. (2011). “Sistemas de Banco de Dados”; tradução Daniel Vieira, revisão técnica Enzo Seraphim e Thatyana de Faria Piola Seraphim; 6<sup>a</sup> ed. São Paulo: Pearson Addison Wesley, 2011.
- Gonçalves, R. et. al. (2011). “A Similarity Search Approach for *Web forms*”. In: Proceedings of the IADIS International Conference IADIS WWW/Internet.
- Hatcher, E.; Gospodnetic, O. (2005). “Lucene in Action”. Greenwich: Manning Publications Co, 2005.
- Madhavan, J. et al. (2009) “Harnessing the Deep Web: Present and Future”. In: 4th Biennial Conference on Innovative Data Systems Research (CIDR 2009).
- Mello, R.S., Pinnamaneni, R., Freire, J., (2010) Indexing Web Form Constraints.,In: Journal of Information and Data Management (JIDM), Vol. 5, nº. 3, p.348-358.
- Silva, F. R.; Mello, R. S. (2012). “Estratégias de Persistência de Clusters em uma Técnica de Casamento por Similaridade para Web Forms”. In: VIII Escola Regional de Banco de Dados (ERBD 2012).

# Um Módulo de Fusão de Dados para Mashups

Oliver Moraes Batista<sup>1</sup>, William Komura<sup>1</sup>, Hugo Bulegon<sup>1</sup>, Carmem S. Hara<sup>1</sup>

<sup>1</sup>Departamento de Informática – Universidade Federal do Paraná  
Caixa Postal 19081 – 81531-980 – Curitiba – PR – Brasil

**Resumo.** *Web mashups são aplicações web que integram conteúdo de diversas fontes de dados disponibilizadas por terceiros através de uma interface de serviço. Para permitir que elas possam ser construídas por profissionais que não possuem a habilidade de programação, existem diversas ferramentas que facilitam a combinação de serviços existentes através de uma interface simples e intuitiva. Dentre estas ferramentas pode ser citado o Exhibit. Através de uma análise das funcionalidades disponibilizadas por esta ferramenta, observou-se que ela possui capacidade limitada para a integração de dados sobrepostos. Ou seja, se a mesma informação é disponibilizada por mais de uma fonte, esta ferramenta não fornece meios adequados para combinar os dados e resolver possíveis conflitos de valor. Motivado por esta deficiência, neste trabalho é proposto um novo módulo para a ferramenta Exhibit, chamado de Mashup Importer, que permite criar mapeamentos de dados provenientes de diferentes fontes que indicam que eles se referem ao mesmo item de dado. Este módulo de fusão de dados pode ser combinado com as demais funcionalidades já existentes na ferramenta para a criação de novos serviços web.*

## 1. Introdução

Com a popularização da Internet, serviços de diversas naturezas foram criados para atender seus usuários, tais como portais de notícias, redes sociais e serviços de geolocalização. Consequentemente, a quantidade de informações disponível tem aumentado constantemente. Grandes empresas como Facebook, Twitter, Google e Last.fm possuem um grande volume de dados e de escopo mundial, devido principalmente aos usuários de seus serviços que a cada dia fornecem novas informações. Alguns serviços web passaram a disponibilizar esses dados para que terceiros os utilizem e dessa forma aumentar a interatividade com os serviços já existentes. Hoje é comum que a partir de um portal de notícias seja possível compartilhar uma informação encontrada em uma rede social, ou que o local de um evento informado em um serviço de shows musicais seja registrado usando outro serviço, como por exemplo de geolocalização.

O termo web mashup denota um novo gênero de aplicações web capazes de integrar o conteúdo de fontes independentes. Um mashup é geralmente definido como uma aplicação web apta a agregar múltiplos componentes, que são dados ou funcionalidades de aplicativos, criados por terceiros e acessados via APIs, que servem para o desenvolvimento rápido de aplicativos de curta duração ou situacionais [Bianchini et al. 2010]. De acordo com [Tuchinda et al. 2011], a construção de um mashup envolve cinco etapas: extração de dados de diferentes fontes, transformação dos dados, limpeza para solução de inconsistências de valores e formato, integração e apresentação dos dados de forma integrada em uma interface Web. Exceto pela última etapa, que trata da apresentação do resultado, percebe-se que as etapas são idênticas a um processo de integração de dados.

O que distingue os mashups é que eles tem como objetivo permitir a construção de novos serviços web e integração de informações por profissionais que não possuem a habilidade de programação.

Diversas ferramentas e ambientes de geração de mashups foram desenvolvidos para que usuários pudessem criá-los facilmente. Dentre elas existem soluções como Yahoo Pipes<sup>1</sup>, IBM Damia agregado à solução IBM Mashup Hub<sup>2</sup>, Apatar<sup>3</sup>, Karma [Tuchinda et al. 2011] e Exhibit<sup>4</sup>. Estas ferramentas diferem na ênfase que dão em cada uma das etapas da construção de um mashup bem como na interface do usuário oferecida para o seu desenvolvimento.

Segundo [Zang and Rosson 2009], há dois tipos principais de interfaces: baseado em *workflow* e programação baseada em exemplos. As ferramentas Yahoo Pipes, Mashup Hub e Apatar seguem a abordagem de workflows, enquanto o Karma e Exhibit utilizam a programação baseada em exemplos. Algumas das ferramentas para geração de mashups são proprietárias, como o Mashup Hub, outras geram mashups que ficam hospedadas apenas nos servidores da empresa que desenvolveu a ferramenta, como o Yahoo Pipes e outras não disponibilizam o código, como o Karma. Dentre as de código livre, encontram-se o Apatar e o Exhibit.

Embora todas as ferramentas tenham sido concebidas para que usuários com pouco ou nenhum treinamento em programação pudessem criar novos serviços a partir da integração de serviços já existentes, o estudo reportado em [Zang and Rosson 2009] mostra que mesmo com uma interface gráfica intuitiva como o Yahoo Pipes, usuários apresentam dificuldade para assimilar noções como uma iteração sobre um conjunto (*Loop*). Por outro lado, o entendimento de documentos XML e *RSS feeds* foi relativamente simples e a maioria dos usuários foi capaz de compreender o significado de algumas linhas de código, baseado no nome dos elementos (*tags*). Embora o estudo conclua que as ferramentas analisadas não estejam suficientemente maduras para permitir que usuários leigos construam mashups, a facilidade de compreensão de pequenos trechos de código mostram o potencial de ferramentas baseadas em exemplo.

Como o Exhibit é uma ferramenta de código livre e que segue a abordagem de programação por exemplo, ele foi escolhido para o desenvolvimento deste trabalho. Embora o Exhibit apresente diversas facilidades para a composição de serviços, constatou-se que o suporte dado para a fusão de dados, ou seja, dados que referem-se a uma mesma entidade no mundo real, é limitado. Para exemplificar, considere dois serviços que tenham como saída arquivos JSON possuindo informações sobre eventos musicais, como nas Figura 1(a) e Figura 1(b). Deseja-se criar um mashup que agrupa informações de ambos, para fornecer uma informação mais completa sobre eventos que estão por acontecer.

Suponha que ambas as fontes possuem campos identificadores como por exemplo, *called* na Fonte 1 e *name* na Fonte 2. Quando ambas as fontes apresentam o mesmo valor para um determinado item de dado, pode-se eliminar um desses campos sem perda de informação. Já para outros campos, esse tratamento pode não ser interessante. No

---

<sup>1</sup><http://pipes.yahoo.com/>

<sup>2</sup><http://www-142.ibm.com/software/products/br/pt/mashuphub/>

<sup>3</sup><http://www.apatar.com/>

<sup>4</sup><http://www.simile-widgets.org/exhibit/>

```

1. { items : [
2.   { type: "evento",
3.     called: "Festival de Rock",
4.     price: "$150",
5.     artist: "Jimi Hendrix",
6.     "geo:lat": "25.430401",
7.     "geo:long": "49.280946",
8.     category: "60's",
9.     location: "somewhere",
10.    date: "20/09/2012",
11.    desc: "Melhor show de todos os tempos"
12.  } ] }

```

(a) Fonte 1 - data1.js

```

1. { items : [
2.   { type: "show",
3.     name: "Festival de Rock",
4.     headliner: "Jimi Hendrix",
5.     price: "$300",
6.     genre: "rock",
7.     date: "18/09/2012",
8.     desc: "Tributo ao verdadeiro rei",
9.     location: "anywhere",
10.    } ] }

```

(b) Fonte 2 - data2.js

```

1. { items : [
2.   { type: ["evento", "show"],
3.     label: "Festival de Rock",
4.     headliner: "Jimi Hendrix",
5.     price: "$300",
6.     "geo:lat": "25.430401",
7.     "geo:long": "49.280946",
8.     genre: ["60's", "rock"],
9.     date: "18/09/2012",
10.    desc: "Melhor show de todos os tempos",
11.    location: "somewhere"
12.  } ] }

```

(c) Resultado da fusão

**Figura 1. Fontes de dados e resultado da Fusão**

exemplo da Figura 1 isso ocorre nos campos *category* e *genre*. Neste caso, fazer a união dos valores de ambos e registrar em apenas um campo torna a informação mais relevante. Já em outros casos, como nos campos *price*, *date*, *desc* e *location* a melhor opção pode ser escolher o valor que se deseja manter como resultado da fusão.

Através do Exhibit um usuário comum não conseguiria fazer a integração de dados do exemplo acima. Para isso, seria necessário que ele tivesse conhecimento da linguagem de programação na qual a ferramenta é implementada. Com o Exhibit é possível carregar os dados de ambas as fontes e informar qual campo deve ser tratado como campo identificador. Porém não é possível realizar, de uma forma simples, esse mapeamento para os outros campos e obter uma visão integrada dos dados, como ilustrado na Figura 1(c).

O objetivo deste trabalho é propor um novo importador para a ferramenta Exhibit. O objetivo do importador é permitir que um usuário comum possa integrar dados provenientes de fontes distintas. Isso pode ser feito, bastando que o usuário conheça as estruturas dos dados e faça o mapeamento entre os campos correspondentes, além de definir de que forma as inconsistências encontradas entre os dados devem ser resolvidas.

O restante deste artigo está organizado da seguinte forma. A seção 2 apresenta algumas estratégias de fusão de dados propostas na literatura. A seção 3 apresenta alguns detalhes da ferramenta Exhibit. O módulo importador que estende a ferramenta com

estratégias de fusão de dados é apresentado na seção 4. A seção 5 apresenta trabalhos relacionados. Por fim, o trabalho é concluído na seção 6 com algumas considerações finais e trabalhos futuros.

## 2. Fusão de Dados

Fusão de dados é o processo de combinar múltiplas representações de um mesmo objeto, extraídas de diversas fontes de dados externas, em uma representação única e limpa; ou seja, uma representação sem inconsistências. Ela em geral é a última etapa do processo de integração de dados, realizada após as etapas de casamento de esquemas e identificação de entidades. Em outras palavras, a fusão de dados é realizada após os objetos que se referem a uma mesma entidade no mundo real já terem sido identificados e seus atributos correspondentes terem sido mapeados.

A ferramenta Exhibit dá suporte ao processo de identificação de entidades através da definição de um atributo de cada fonte de dados que é considerado como chave primária. No exemplo da figura 1, é possível definir que o atributo *called* é chave para a Fonte 1, bem como o atributo *name* é chave para a Fonte 2. Assim, sempre que as fontes possuirem itens que tenham valores coincidentes para estes atributos, a ferramenta considera que eles se referem à mesma entidade no mundo real. Contudo, o Exhibit não dá suporte ao casamento de esquemas, considerando como atributos correspondentes somente aqueles que possuem o mesmo nome. Além disso, as inconsistências entre os valores de atributos também não são tratadas pela ferramenta. Ou seja, ela não dá suporte à fusão de dados.

A fusão de dados em geral é baseada em estratégias que determinam como possíveis inconsistências entre os dados são resolvidas. Um tutorial das estratégias existentes pode ser encontrada em [Bleiholder and Naumann 2008]. Neste trabalho, são consideradas duas estratégias:

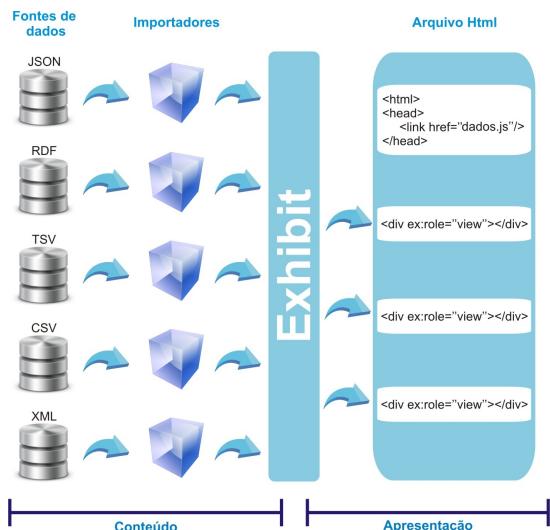
- **concatenação:** esta estratégia concatena todos os valores distintos fornecidos pelas fontes de dados;
- **escolha de fonte:** esta estratégia mantém apenas o valor fornecido pela fonte definida como prioritária.

Um exemplo da estratégia de concatenação é apresentada na figura 1 para obter o valor do atributo *type* no documento apresentado na Figura 1(c), que contém os valores “evento” e “show”, fornecidos pelas Fontes 1 e 2. Já para o atributo *price*, a Fonte 2 foi escolhida como prioritária e para o atributo *location* a Fonte 1 foi escolhida como prioritária. Como resultado, os valores após a fusão dos dados são “\$300” e “somewhere”, fornecidos pelas Fontes 2 e 1, respectivamente.

## 3. A Ferramenta Exhibit

O Exhibit é uma ferramenta livre para a geração de mashups na qual o desenvolvedor da aplicação tem à sua disposição arquivos HTML com exemplos de utilização dos componentes de visualização da ferramenta. Com base nestes exemplos, que podem ser considerados “esqueletos” de arquivos HTML, novas aplicações podem ser criadas através da composição e integração de dados provenientes de diversas fontes. A Figura 2 ilustra a arquitetura do Exhibit do ponto de vista de conteúdo e apresentação. Objetos importadores fornecem dados à ferramenta, obtidos a partir de fontes de dados em vários formatos. Este

conteúdo é utilizado por objetos de apresentação que inserem código nos locais marcados no arquivo HTML.



**Figura 2. Arquitetura do Exhibit**

Uma fonte de dados é um arquivo que contém os dados a serem inseridos no website. Os dados podem estar nos formatos JSON, RDF, TSV, CSV e XML. O formato padrão do Exhibit é o JSON, com uma sintaxe um pouco mais relaxada que o padrão JSON<sup>5</sup> proposto. Ele consiste em um array de itens, como ilustrado na Figura 1(a), 1(b) e 1(c). Cada item pode ser considerado como um registro de um banco de dados e consiste basicamente de um conjunto de pares “propriedade: valor”.

Para cada um dos formatos de descrição de dados, existe um importador correspondente. Importadores são objetos instanciados pelo Exhibit, responsáveis por carregar os dados e repassá-los ao banco de dados da ferramenta. Com exceção do formato padrão, os importadores também realizam um pré-processamento, no qual analisam o conteúdo em seu formato e traduzem para o formato JSON.

O arquivo HTML é um componente de apresentação. Nele é descrita a fonte de dados, dentro da tag *head*, e no corpo do HTML são descritos os marcadores. Marcadores são tags HTML, a princípio tags *div*, que possuem atributos reconhecidos pela ferramenta, como o atributo *ex:role*. É dentro destas tags marcadas, que o Exhibit insere código que produzirá uma visualização dos dados. Objetos de apresentação são elementos da ferramenta responsáveis pela geração de código. É através do valor do atributo de marcação que o Exhibit escolhe qual objeto de apresentação será responsável em gerar código no trecho marcado.

Para desenvolver uma nova aplicação, é necessário apenas um editor de texto. Um conhecimento básico em HTML é útil, mas não essencial, pois em muitos momentos o usuário precisa apenas copiar e colar trechos e modificá-los de acordo com a sua necessidade.

A Figura 3 apresenta um arquivo HTML (*fonte1.html*) para apresentar um conjunto de itens de dados como ilustrado na Figura 1(a). No arquivo HTML existe uma cha-

<sup>5</sup><http://www.json.org/>

```

1. <html>
2. <head>
3.   <title>Eventos Musicais</title>
4.   <link href="fonte1.js" type="application/json" rel="exhibit/data"/>
5.   <script src="http://api.simile-widgets.org/exhibit/2.2.0/exhibit-api.js" type="text/javascript">
6.     </script>
7.   <style> </style>
8. </head>
9. <body>
10.  <h1>Eventos Musicais</h1>
11.  <table width="100%">
12.    <tr valign="top">
13.      <td ex:role="viewPanel"> <div ex:role="view"></div>
14.      </td>
15.      <td width="25%">
16.        controles de navegação são colocados aqui
17.      </td>
18.    </tr>
19.  </table>
20. </body>
21. </html>

```

**Figura 3. Utilização do Exhibit para visualização da Fonte 1.**

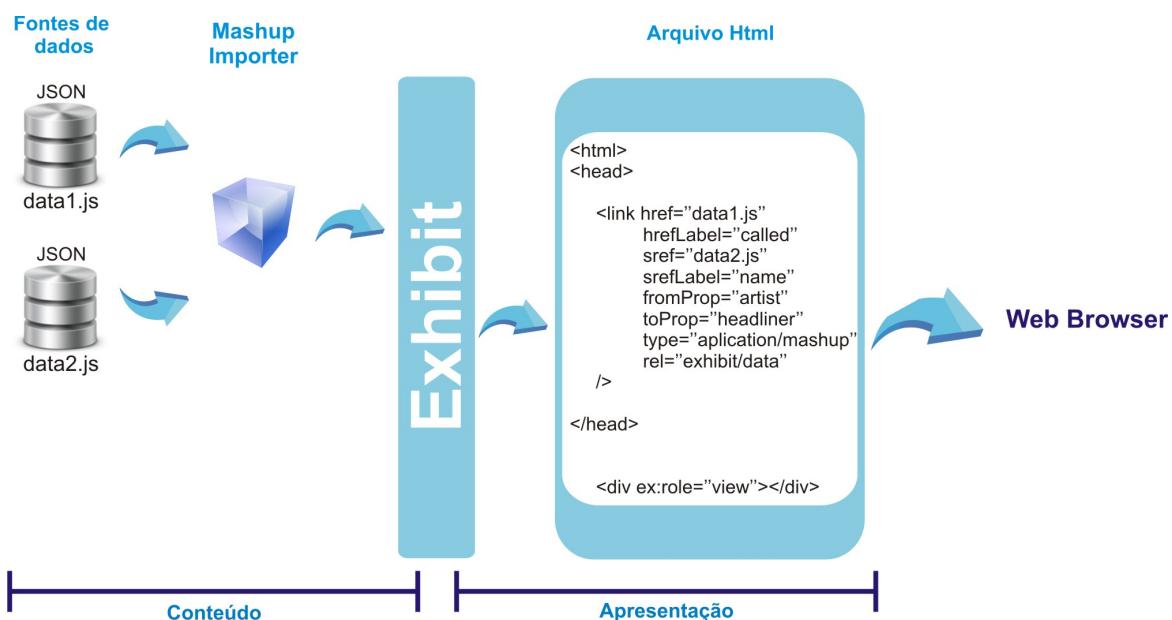
mada para um script externo (Linha 5), que carrega a ferramenta e a inicializa. Existem também dois atributos de elementos que não fazem parte do padrão HTML. Os atributos *ex:role="viewPanel"* e *ex:role="view"* (Linha 13) funcionam como marcadores para o Exhibit. Para ter acesso à nova aplicação, basta abrir o arquivo *fonte1.html* em um navegador. Com apenas estes passos é possível construir um serviço que mostra uma base de dados de uma forma simples. Nota-se aqui a simplicidade para criação de um serviço usando Exhibit, o que vai de encontro com a sua proposta.

O Exhibit, após ser iniciado, instancia um objeto que é responsável por buscar as referências a arquivos JSON descritas no HTML e carregar seus conteúdos através do respectivo importador. As fontes de dados são declaradas através de elementos *link*(Linha 4 da Figura 3) inseridos nos escopo de elementos *head* do HTML. Estes elementos possuem basicamente três atributos. O atributo *rel* informa que o link é do contexto do Exhibit. O nome do arquivo que contém os dados é informado no atributo *href*. Para saber qual importador usar a ferramenta obtém a informação contida no atributo *type*. Como citado anteriormente, o Exhibit já contém importadores para os formatos JSON, RDF, TSV, CSV e XML. Porém, ele não provê suporte para a fusão de dados. Um novo módulo para estender o Exhibit com esta funcionalidade é descrito na próxima seção.

#### 4. Um Módulo de Fusão de Dados para o Exhibit

Quando uma aplicação/serviço web disponibiliza seus dados para uso por terceiros ela define um formato para exportação destes dados. Contudo, na maioria dos serviços que disponibilizam dados, não existe um padrão de nomeação das propriedades ou atributos. As figuras 1(a) e 1(b) mostram exemplos que, embora possuam propriedades diferentes, representam o mesmo evento no mundo real. Estas saídas também apresentam valores diferentes para estes atributos. Um mashup que possua como entrada estes dados deve

agrupá-los em uma representação única e consistente. A Figura 1(c) ilustra um resultado dessa fusão de dados. Para prover esta facilidade pela ferramenta exhibit, nesta seção é apresentado o *Mashup Importer*. Ele é um importador que possibilita que duas fontes de dados no formato JSON sejam carregados, mapeando os atributos correspondentes e a definição de estratégias para a resolução de conflitos. A Figura 4 mostra a arquitetura do importador no escopo da ferramenta Exhibit.



**Figura 4. Arquitetura do Mashup Importer**

Como em qualquer outro importador da ferramenta, para que a fusão de dados seja realizada, no código HTML deve ser criada uma tag *link* dentro do escopo da tag *head*. Entretanto o importador criado necessita de mais informações para poder prover as novas funcionalidades. A figura 5 apresenta um exemplo deste trecho de código.

Para usar o *Mashup Importer* o desenvolvedor deve fornecer algumas informações através de atributos, que são descritos na sequência.

1. *href* - Nome do arquivo da primeira fonte de dados.
2. *hrefLabel* - Nome do campo que atua como identificador na primeira fonte.
3. *sref* - Nome do arquivo da segunda fonte de dados.
4. *srefLabel* - Nome do campo que atua como identificador na segunda fonte.
5. *fromProp* - Nomes de campos da primeira fonte que serão mapeados.
6. *toProp* - Nomes de campos da segunda fonte que serão mapeados.

1. <link href="data1.js" hrefLabel="called"
2. sref="data2.js" srefLabel="name"
3. fromProp="artist, category" toProp="headliner, genre"
4. fusionDefault="concat"
5. fusionAttrib= "(price, sref) (date, sref) (desc, href) (location, href)"
6. type="application/mashup" rel="exhibit/data" />

**Figura 5. Declaração de fusão de dados no Exhibit**

7. *fusionDefault* - define a estratégia padrão para realizar a fusão de dados, que pode ser:
  - **concat**: para a estratégia de concatenação de dados;
  - **href**: para escolher prioritariamente o valor fornecido pela fonte relacionada ao atributo *href*;
  - **sref**: para escolher prioritariamente o valor fornecido pela fonte relacionada ao atributo *sref*.
8. *fusionAttrib* - sequência de pares (atributo, estratégia) para declarar estratégias diferentes para atributos específicos. Ou seja, caso haja uma inconsistência em um atributo, ela é primeiramente resolvida com a estratégia declarada especificamente para o atributo. Caso não haja uma estratégia específica, a estratégia default é utilizada.
9. *type* - Indica ao Exhibit qual importador usar.
10. *rel* - Indica ao Exhibit que esta é uma tag link que ele deve considerar.

A ferramenta Exhibit requer que cada item de uma base de dados possua o campo identificador *label*. Este campo é obrigatório pois através dele o Exhibit realiza a fusão de itens. Todo item que possua o mesmo valor para este campo é tratado como um só. Ou seja, é considerado que eles referem-se ao mesmo objeto do mundo real e portanto devem ser apresentados como um único objeto no novo serviço. Para utilizar o *Mashup Importer* o desenvolvedor do mashup deve indicar quais campos serão tratados como identificadores através dos atributos *hrefLabel*, para a primeira fonte, e *srefLabel*, para a segunda.

O mapeamento de campos é realizado pelo desenvolvedor informando os campos a serem mapeados nos atributos *fromProp* e *toProp*, separados por vírgula. O primeiro campo descrito em *fromProp* será mapeado com o primeiro campo descrito em *toProp*, e assim por diante. A Figura 5 mostra o mapeamento do campo *artist* da primeira fonte para o campo *headliner* da segunda fonte e o mapeamento do campo *category* da primeira fonte para o campo *genre* da segunda fonte.

O atributo *type* deve sempre possuir o valor *application/mashup*. É através desse atributo que a ferramenta sabe qual importador deve usar. O atributo *rel* deve sempre possuir o valor *exhibit/data*, para indicar ao Exhibit que a tag possui dados que ele deve carregar.

Com o uso do *Mashup Importer* o problema relatado na introdução deste artigo é solucionado. O importador realiza sua tarefa através da renomeação de nomes de campos. Os campos *called* (Figura 1(a)) e *name* (Figura 1(b)) recebem o nome *label*. Dessa forma os itens são tratados como um único dado, que é comportamento padrão do Exhibit. Para o mapeamento de campos o importador renomeia os campos descritos no atributo *fromProp* utilizando os nomes de campos descritos no atributo *toProp*. Sendo assim o campo *artist* (Figura 1(a)) é renomeado para *headliner* e o campo *category* é renomeado para *genre*. Quando os campos mapeados possuem o mesmo valor, um dos valores é desprezado para que não haja duplicatas no resultado final. Caso contrário, as estratégias de fusão são aplicadas. No caso dos campos *category* (Figura 1(a)) e *genre* (Figura 1(b)) o resultado final é uma concatenação dos valores de ambos, seguindo a estratégia definida como default. Similarmente, conflitos nos valores dos atributos *price* e *date* escolhem o valor fornecido pela Fonte 2, enquanto os valores mantidos para os atributos *desc* e *location* são provenientes da Fonte 1. O resultado deste processo está ilustrado na Figura 1(c).

Observe que para a apresentação do resultado da fusão em um navegador Web, o trecho de código apresentado na Figura 5 pode ser inserido no código apresentado na Figura 3, substituindo a Linha 4. Como o resultado da fusão é gerado no momento da exibição do arquivo html, caso haja a necessidade de alterar ou adicionar estratégias de fusão, o usuário pode simplesmente alterar este arquivo para que nas futuras exibições as novas estratégias sejam consideradas.

## 5. Trabalhos Relacionados

Um exemplo de serviço criado apenas com o uso de dados ou serviços providos por terceiros é o StereoMap[Duszczak et al. 2010]. Ele é um mashup que permite a busca de eventos musicais por localidade, fazendo uso dos serviços Last.fm<sup>6</sup>, Google Maps<sup>7</sup>, Twitter<sup>8</sup> e Wikipedia<sup>9</sup>. Nele o usuário, ao informar uma localidade, tem como retorno os eventos musicais da cidade de uma forma visual com o uso de marcadores em um mapa. O usuário pode acessar informações do artista via Wikipedia e comunicar ou convidar seus amigos através do Twitter. O StereoMap foi criado usando a linguagem de programação Javascript.

O conceito de mashup auxilia a composição de serviços e seu entendimento. Em [Abiteboul et al. 2008] é apresentado um modelo formal para mashups baseado em mashlets, componente básico do modelo. Um mashlet pode importar dados de determinada fonte, importar outro mashlet, usar serviços web externos e impor padrões de interação entre seus componentes. O modelo é hierárquico no sentido que um mashlet pode incorporar outro mashlet, que por sua vez incorpora outros mashlets, e assim por diante, recursivamente. Há sistemas que proveem soluções avançadas para a integração de dados, como Potter's Wheel [Raman and Hellerstein 2001], que dá suporte ao processo de limpeza de dados e integração de esquemas, mas que não foi proposto para a construção de mashups especificamente. O trabalho que mais se aproxima aos propósitos do Mashup Importer é o sistema Karma [Tuchinda et al. 2011], que tem como objetivo não apenas a construção de mashups através de exemplos, mas também a integração e resolução de conflitos.

## 6. Conclusão

Este artigo apresenta uma extensão da ferramenta de construção de mashups Exhibit com a funcionalidade de fusão de dados. Esta funcionalidade foi obtida com o desenvolvimento de um novo importador na ferramenta. O importador possibilita a declaração de atributos correspondentes em duas fontes de dados e estratégias para a resolução de valores de atributos, caso elas existam. Web mashups são uma tendência já estabelecida na Internet. Entretanto, pode ser concluído através deste estudo, que a tecnologia ainda é um campo de investigação não completamente explorado. Isso vai de encontro com a proposta do Mashup Importer, que facilita a fusão de dados na ferramenta Exhibit, tornando-a mais robusta para geração de web mashups. O novo módulo soluciona o problema introduzido na introdução do artigo. Este trabalho pode ser estendido de diversas formas em trabalhos futuros, dentre as quais podem ser citadas:

---

<sup>6</sup><http://www.last.fm/>

<sup>7</sup><http://maps.google.com/>

<sup>8</sup><http://twitter.com/>

<sup>9</sup><http://www.wikipedia.org/>

- Inclusão de suporte a outros formatos de entrada como o XML, que é um padrão de troca de conteúdo na Web;
- Possibilitar a fusão de dados de mais que duas fontes de dados e utilização de novas políticas para fusão, tais como o dado mais recente ou o valor fornecido pela maioria das fontes.

## Referências

- Abiteboul, S., Greenspan, O., and Milo, T. (2008). Modeling the mashup space. In *Proceeding of the 10th ACM workshop on Web information and data management*, WIDM '08, pages 87–94, New York, NY, USA. ACM.
- Bianchini, D., De Antonellis, V., and Melchiori, M. (2010). Semantic-driven mashup design. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '10, pages 247–254, New York, NY, USA. ACM.
- Bleiholder, J. and Naumann, F. (2008). Data fusion. *ACM Computing Survey*, 41(1):1–41.
- Duszczak, J., Zambom, L., Batista, O., Ferreira, L., Ibrahim, I., and Hara, C. (2010). Stereomap: Um mashup para busca de eventos musicais. In *VI Escola Regional de Banco de Dados*, ERBD '2010.
- Raman, V. and Hellerstein, J. M. (2001). Potter's wheel: An interactive data cleaning system. In *Proc. of the 27th VLDB Conference*.
- Tuchinda, R., Knoblock, C. A., and Szekely, P. (2011). Building mashups by demonstration. *ACM Transactions on the Web*, 5(3).
- Zang, N. and Rosson, M. B. (2009). Playing with information: How end users think about and integrate dynamic data. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 85–92.

# Análise espaço-temporal de mensagens do Twitter

Renata de J. Silva<sup>1</sup>, Luis Otavio Alvares<sup>1</sup>

<sup>1</sup>Depto. Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)  
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brasil

{renatadej.silva,alvares}@inf.ufsc.br

**Abstract.** *We live in an era in which access to the Internet becomes increasingly common. Social networks such as Twitter microblog, have recorded a significant increase of posted messages. Some of these messages have the geographical coordinates of the location where they were issued. This paper proposes the analysis of these posts considering the spatio-temporal aspects in order to obtain knowledge about users of Twitter. For this, we propose changes in the Weka data mining toolkit to obtain better results on twitter data analysis. Experiments were performed with real data obtaining good results.*

**Resumo.** *Estamos vivenciando uma era em que o acesso à internet torna-se cada vez mais frequente. As redes sociais, como o microblog Twitter, tem registrado um aumento significativo de mensagens postadas. Parte destas mensagens possui as coordenadas geográficas do local de onde foram emitidas. Este artigo propõe a análise destas mensagens considerando os aspectos espaço-temporais de modo a obter conhecimento sobre os usuários do Twitter. Para isto, propõe adaptações na ferramenta de data mining Weka de forma a obter melhores resultados. Experimentos foram realizados com dados reais, com bons resultados.*

## 1. Introdução

Com a popularização das redes sociais na internet, a disseminação e o acesso à informação tornou-se muito mais ágil. Uma dessas redes, o Twitter<sup>1</sup>, na verdade mais considerado um microblog do que uma rede social, tem características particulares: suas mensagens são limitadas a 140 caracteres e usualmente são postadas de dispositivos móveis como celulares e *smartphones*; a maioria das mensagens reflete onde o usuário está, ou o que ele está fazendo ou sentindo naquele momento; para receber as postagens de um usuário (ser um seguidor) não há necessidade de concordância deste usuário.

Com mais de 500 milhões de usuários [UOL Tecnologia 2012] e 500 milhões de mensagens por dia [Olhar Digital UOL 2012], o Twitter é uma fonte impressionante de informações. Entretanto, analisar milhões de dados publicados diariamente no Twitter é muito trabalhoso e inviável manualmente. Uma alternativa é aplicar técnicas de mineração de dados. Alguns estudos já abordam este problema, mas muito pouco existe que considere os aspectos espacial e temporal simultaneamente.

---

<sup>1</sup> <http://twitter.com>. Último acesso em março 2013.

Este trabalho tem o foco em mineração de dados utilizando a base de dados do Twitter, com *tweets* – mensagens publicadas no Twitter – georreferenciados. São apresentadas adaptações na ferramenta Weka para que as análises de dados espaço-temporais dos *tweets* possam se tornar mais eficazes e eficientes. Mais especificamente, é abordada a técnica de formação de agrupamentos com o algoritmo DBSCAN [Ester et al 2006], cuja saída é incrementada com uma visualização na forma de mapas e a indicação das palavras mais frequentes nas mensagens de cada cluster, de modo a se ter uma ideia geral do conteúdo das mensagens.

O restante do artigo está organizado como segue: a seção 2 apresenta alguns trabalhos relacionados; a seção 3 apresenta o que é a ferramenta Weka; a seção 4 apresenta o que foi adaptado nesta ferramenta a fim de melhorar a capacidade de resposta às análises; na seção 5 são apresentados alguns experimentos realizados; e por fim a seção 6 expõe a conclusão e trabalhos futuros.

## 2. Trabalhos Relacionados

As redes sociais na internet são relativamente recentes e o volume de seus dados tem crescido exponencialmente nos últimos anos. A descoberta de conhecimento neste novo tipo de dado tem suscitado muito interesse e vários trabalhos tem abordado o tema. Entretanto, trabalhos considerando os aspectos espaço-temporais das mensagens postadas são bem menos numerosos. Por exemplo, no Twitter, a localização geográfica do local de postagem das mensagens passou a ser disponibilizada apenas em 2010. Alguns trabalhos que abordam a descoberta de conhecimento espaço-temporal em dados do Twitter são mencionados a seguir.

Como os usuários usam bastante o Twitter para informar a seus seguidores o que estão fazendo no momento, esta rede tem características de tempo-real. Sakaki em [Sakaki et al 2010] usou esta característica para a detecção de eventos naturais como terremotos e tufões, usando as mensagens do Twitter como sensores, analisando as palavras das mensagens.

Um sistema para a descoberta de atividades sociais fora do padrão é proposto em [Lee et al 2011]. É utilizado o algoritmo K-means. Cada grupo formado é analisado considerando comportamentos de agregação (usuários que estavam em outros locais e agora estão neste) e dispersão (usuários que estavam neste local e agora estão em outros). Um pico nos dados de agregação é um indício de um evento social. Outro trabalho nesta área, mas que refina o processo com uma análise visual interativa foi proposto recentemente [Chae et al 2012]. O artigo [Lee 2012] vai mais além, pois preve a possível evolução e impacto dos eventos detectados.

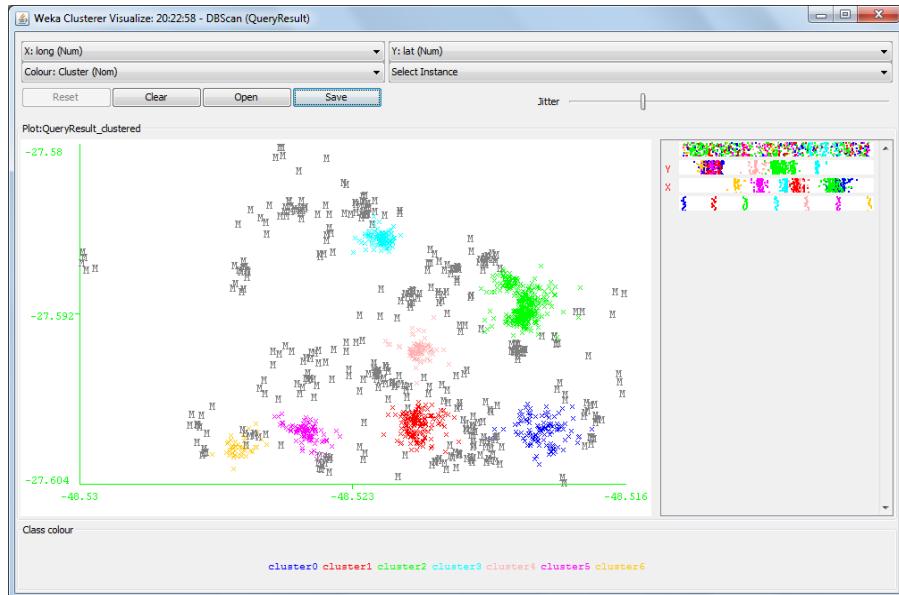
Com o presente trabalho, aplicando técnica de mineração de dados, foram detectadas regiões de grande concentração de *tweets*. Além disto, para obter conhecimento sobre o que os usuários do Twitter estão fazendo nestas regiões densas, foi utilizado um sistema de busca em texto para capturar as palavras mais frequentes.

## 3. A ferramenta Weka

Para a mineração e análise dos dados, utilizou-se o Weka [Witten & Frank 2005]. O Weka é uma ferramenta criada na Universidade de Waikato, Nova Zelândia, de código aberto, desenvolvido na linguagem de programação Java e muito utilizada nos meios acadêmicos. Esta ferramenta possui uma coleção de algoritmos para execução das tarefas de mineração de dados.

A técnica utilizada neste trabalho foi a de Agrupamento (*Clustering*) e o algoritmo aplicado foi o DBSCAN [Ester et al 2006], que é um algoritmo baseado em densidade, isto é, as regiões densas formam os *clusters*. Para ser considerada densa, uma região deve ter um número mínimo de pontos (parâmetro “*minPoints*”) dentro de um círculo (parâmetro “*epsilon*”, raio do círculo).

Na ferramenta Weka, após a execução do algoritmo DBSCAN, é possível visualizar os resultados como mostra o exemplo da Figura 1.



**Figura 1. Visualização do DBSCAN na ferramenta Weka**

Neste exemplo, é possível identificar os *clusters* que o algoritmo formou, neste caso 7. Os *clusters* são identificados pelas diferentes cores e, pode-se visualizar a posição de cada *cluster*, pois no eixo X foi plotado o atributo longitude e no eixo Y a latitude do ponto em que cada mensagem foi postada. Os pontos (*tweets*) que não pertencem a nenhum *cluster* aparecem na cor cinza e são considerados “ruído” ou *noise* pelo algoritmo.

O algoritmo DBSCAN, na ferramenta Weka, não possui recursos para trabalhar com dados geográficos. Desta maneira, é possível notar que seria difícil analisar este tipo de dado, pois não há informações geográficas, ou seja, não se tem como saber em que parte de uma cidade ou país está cada *cluster*. Para melhorar as análises, foram realizados melhoramentos no Weka, descritos na próxima seção.

#### 4. Adaptações na Ferramenta Weka

Para que os resultados pudessem ser analisados de maneira mais ágil, sem que fosse necessário grande esforço humano, a ferramenta Weka foi adaptada. Para isto, foram desenvolvidos 2 recursos novos na ferramenta: (i) geração de um mapa onde cada marcador representa o centróide (centro de gravidade) de um *cluster*; (ii) com o clique de mouse em um marcador, podem ser visualizadas as palavras mais frequentes do *cluster* correspondente.

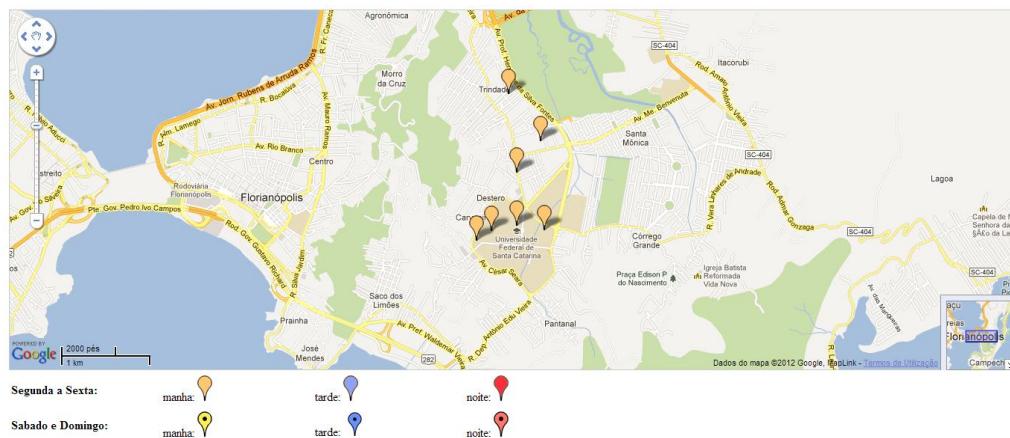
## **4.1. Geração de mapa com a API Google Maps**

Para facilitar a análise de dados geográficos, optou-se por implementar a geração de um mapa real. Foi adicionado um método responsável por esta ação que é automaticamente executado durante a execução do algoritmo DBSCAN do Weka.

Para o desenvolvimento da geração do mapa, foi utilizada a API do Google Maps. Para a inserção de múltiplos pontos com ícones personalizados, foi utilizado como base o *script* do site *Link Nacional* [Link Nacional 2011]. Com isso, foi possível indicar latitude, longitude, ícone/marcador e descrição para cada *cluster*.

No mapa desenvolvido, cada marcador representa o centróide de um *cluster*. Isto foi feito porque plotar todos os pontos de um *cluster* iria poluir muito o mapa e, com isso, dificultaria a análise. Além disso, o conjunto dos pontos de um *cluster* já pode ser visualizado na interface padrão do Weka, se houver necessidade de se conhecer melhor a distribuição dos pontos do *cluster*.

O resultado da geração do mapa é um arquivo HTML. A Figura 2 é um exemplo de como os *clusters* são visualizados na interface que foi desenvolvida neste trabalho. A interface mostra os centróides dos *clusters* gerados, representados pelos marcadores, que também identificam o período da mensagem (manhã, tarde ou noite) conforme a sua cor, e se foram postados em dias de semana ou nos fins de semana. No exemplo da Figura 2, todos os *clusters* são de mensagens postadas no período da manhã nos dias de semana.



**Figura 2. Visualização do mapa gerado**

Conforme pode ser observado na Figura 2, o mapa desenvolvido facilita a análise de dados geográficos, pois é possível identificar onde cada *cluster* está situado no espaço, ou seja, é possível visualizar sobre qual local o *cluster* está localizado e também verificar nomes de ruas e bairros, a existência de rios, morros, etc.

No mapa gerado, os recursos do Google Maps podem ser utilizados (por exemplo, utilizar o *zoom*) e, além disso, foi implementada uma legenda com informações dos marcadores.

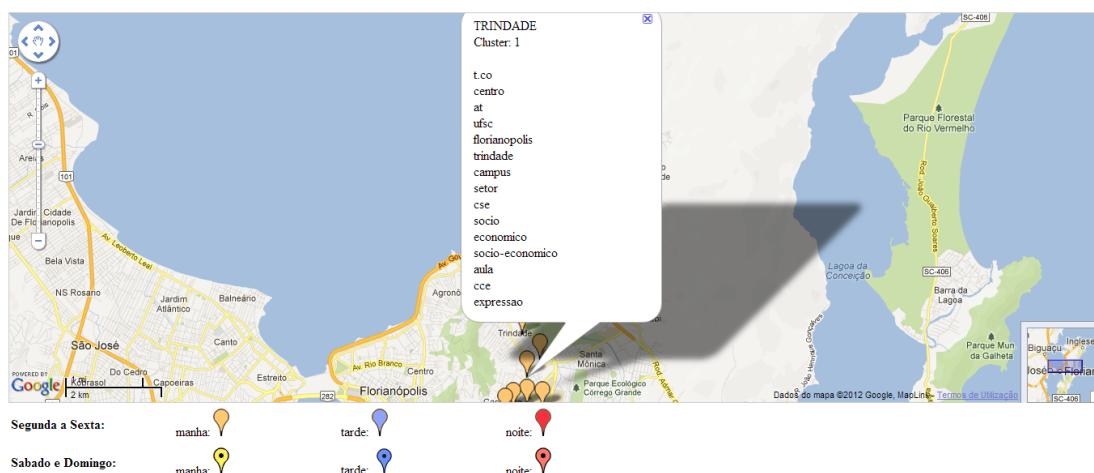
#### **4.2. Obtenção de Palavras Frequentes**

Para conhecer melhor o que os usuários do Twitter estão fazendo, foi implementada uma funcionalidade que captura as palavras mais frequentes das mensagens de cada agrupamento formado. Para isto, foi utilizada a biblioteca de tratamento de texto Tsearch2 [Bartunov & Sigaev 2012], que é uma extensão do PostgreSQL, desenvolvida

na Universidade de Moscou. Optou-se por adaptar esta biblioteca em vez de desenvolver a funcionalidade, pois é uma tarefa complexa e que deve ser computacionalmente eficiente.

Antes de aplicar a função do Tsearch2, para a análise do texto do *tweet* em si, decidiu-se eliminar a acentuação ortográfica, para que a busca por texto encontrasse mais ocorrências de uma mesma palavra.

O Tsearch2 possui diversas opções para tratamento de texto, como eliminação de *stopwords*, *stemming*, etc. Para este estudo não foi realizado *stemming*, de modo que, por exemplo, os termos “casa” e “casarão” são considerados distintos. Foi utilizado o conjunto de *stopwords* (palavras ignoradas pelo sistema) referente à língua portuguesa, acrescentado de outras palavras observadas no decorrer do trabalho como irrelevantes para o estudo, como por exemplo, as letras isoladas e expressões como *4square*.



**Figura 3. Visualização da interface com as palavras mais frequentes de um cluster**

A Figura 3 apresenta a interface desenvolvida para a visualização das palavras mais frequentes nos *tweets* de um *cluster*. Basta clicar sobre um marcador para que a lista das palavras mais frequentes no cluster representado pelo marcador seja apresentada.

## 5. Experimentos Realizados

Para avaliar a eficácia das extensões realizadas foram realizados experimentos com *tweets* postados na cidade de Florianópolis, no período de abril a novembro de 2011, e contendo as coordenadas geográficas do local em que foram postados. Os *tweets* com as coordenadas geográficas corresponderam a aproximadamente 10% dos *tweets* emitidos. O SGBD utilizado foi o PostgreSQL. A escolha deste SGBD foi feita por este permitir a manipulação de dados geográficos por meio da extensão PostGIS, que segue o padrão OGC [OGC 2008].

As informações mais relevantes contidas na base de dados são: latitude e longitude (ambas do tipo “double”), data/hora de postagem da mensagem (tipo “timestamp”) e texto do *tweet* propriamente dito (tipo “text”).

Como havia a intenção de realizar a análise dos *tweets* por bairro de Florianópolis, uma primeira preparação dos dados foi a determinação do bairro em que os *tweets* foram postados. Para isto, inicialmente, foi criada na tabela *tweets* uma

coluna de tipo “geometry”, necessária para a utilização das funções espaciais do PostGIS. Assim, para cada *tweet* foi gerado um tipo geométrico “ponto”, por meio da função “ST\_MakePoint” do PostGIS, aplicada aos campos latitude e longitude.

Em seguida, foi utilizado um arquivo *shapefile* disponível no site do Instituto Brasileiro de Geografia e Estatística (IBGE) para a obtenção dos limites dos bairros de Florianópolis. Desta maneira, foi empregada a função “ST\_Contains” no PostGIS para o cruzamento da tabela de bairros e tabela de *tweets* para, enfim, popular a coluna com a informação do bairro em que o *tweet* foi postado.

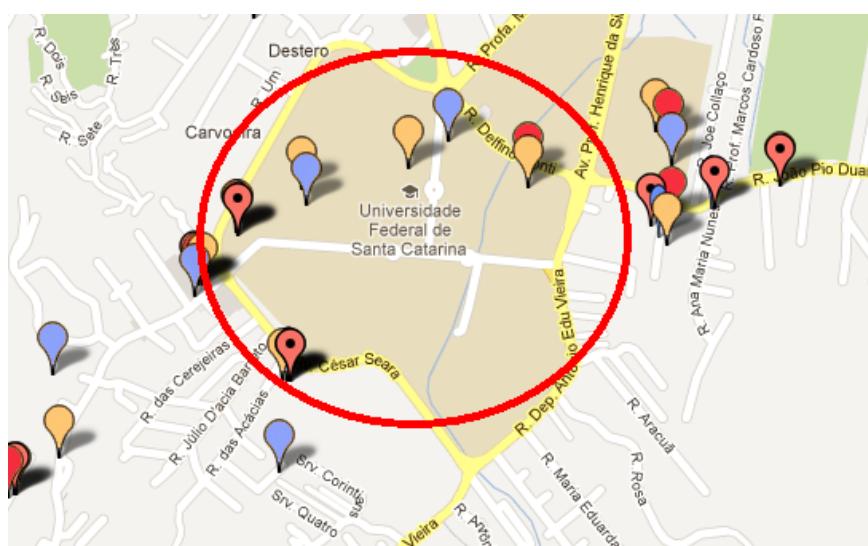
Entre os 35 bairros de Florianópolis, 15 foram desconsiderados para a pesquisa em função do pequeno número de postagens realizadas. O total de registros (*tweets*) analisados foi de 152.552.

O conjunto de dados foi filtrado por 3 atributos: bairro, período do dia (manhã, tarde ou noite) e dia da semana (segunda a sexta-feira ou sábado e domingo), totalizando 6 consultas por bairro.

O algoritmo DBSCAN foi executado, inicialmente, de maneira padrão para os 20 bairros de estudo (minPoints = 2,5% dos *tweets* do bairro, epsilon = 0,045). Isto quer dizer que, para cada bairro, o algoritmo DBSCAN foi executado 6 vezes, totalizando 120 consultas no banco de dados aplicadas ao software Weka. Para que o estudo não se tornasse cansativo e para evitar o trabalho manual, o código da ferramenta Weka foi adaptado para automatizar estas execuções das consultas.

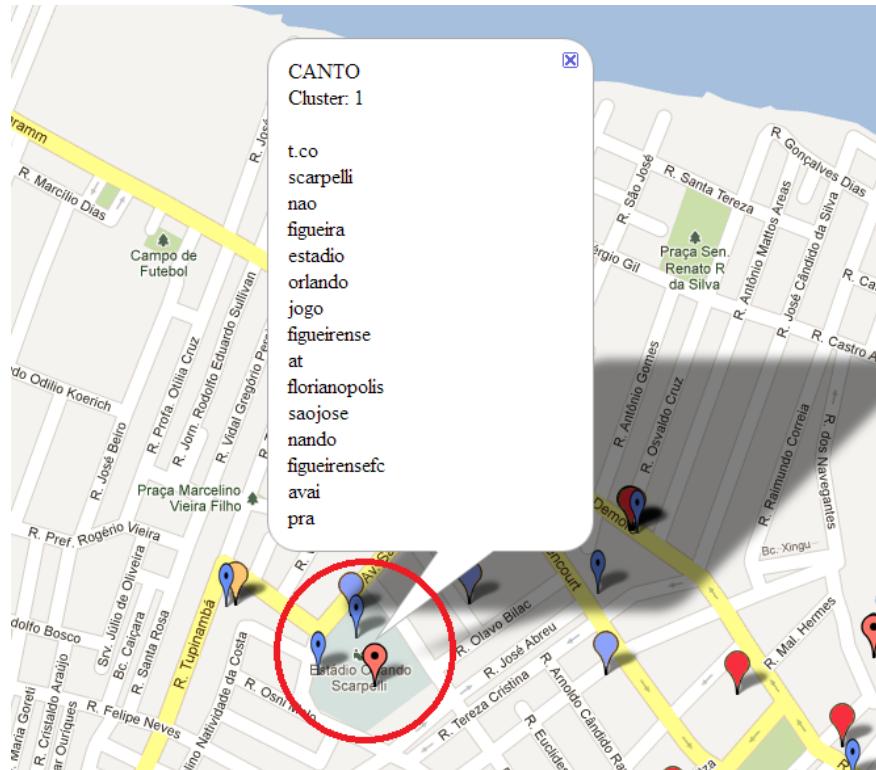
Como o algoritmo utilizado neste trabalho foi o DBSCAN, e o mesmo tem por característica gerar grupos em regiões densas, existe uma grande possibilidade de os pontos centrais gerados por *cluster* estejam sobre, ou muito próximos, a locais atrativos. Por exemplo, universidades, restaurantes, bares, shoppings centers, estádios de futebol, centros comerciais, empresas, entre outros. Isto pode ser percebido ao visualizar os marcadores plotados pelo Weka na interface desenvolvida neste trabalho.

Exemplos dos resultados obtidos com essa análise são mostrados nas Figuras 4 e 5. A Figura 4 apresenta os centróides de *clusters* no bairro Trindade. Pode-se observar que muitos *clusters* estão no campus da UFSC durante os dias de semana (marcado pelo círculo) e que nas noites de finais de semana muitos clusters são formados no entorno da UFSC (marcadores com ponto preto), que é uma região de muitos bares.



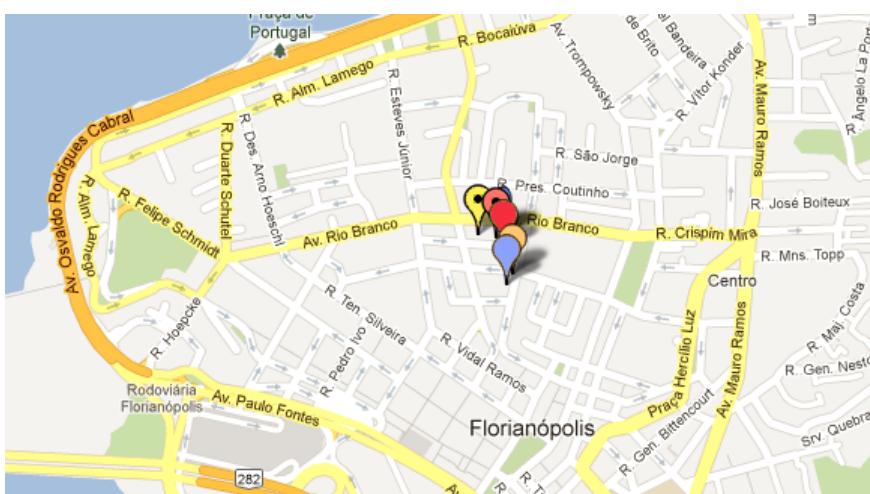
**Figura 4. Clusters formados na UFSC e entorno**

A Figura 5 apresenta *clusters* formados no estádio de futebol Orlando Scarpelli (marcado com o círculo) nas tardes e noites de finais de semana, o que deve corresponder a jogos sábados à noite e domingos à tarde. Além disso, esta figura apresenta as palavras mais frequentes encontradas em um dos *clusters*. Pode-se notar que estas palavras estão relacionadas a futebol.



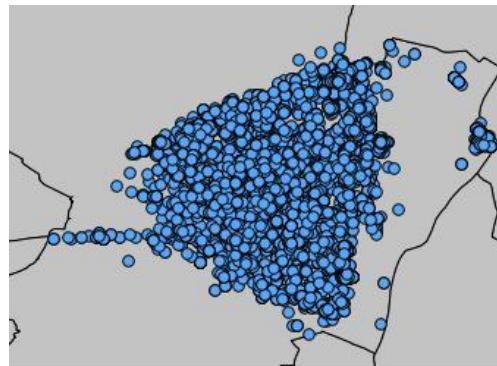
**Figura 5. Clusters formados no estádio de futebol Orlando Scarpelli**

Se para os demais bairros os parâmetros utilizados foram razoáveis, para o bairro Centro, a maioria das consultas gerou somente um *cluster* situado no meio deste bairro (Figura 6). Os centróides ficaram aproximadamente no meio do bairro porque os dados eram muito numerosos e geograficamente homogêneos. A Figura 7 apresenta os *tweets* plotados no mapa, visualizado pela ferramenta Quantum GIS (<<http://www.qgis.org>>). Estes dados são somente do bairro Centro no período da tarde no intervalo de segunda a sexta-feira, totalizando 11.315 registros.



**Figura 6. Clusters formados no bairro Centro**

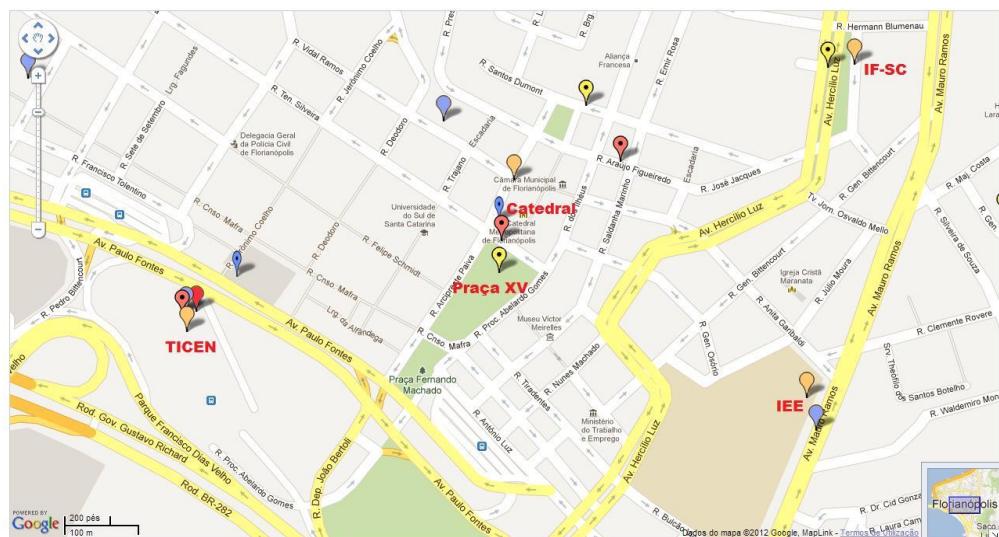
Para que o algoritmo DBSCAN possa gerar mais *clusters*, neste caso, é necessário diminuir o valor dos parâmetros “epsilon” e “minPoints”. Por conseguinte, no bairro Centro, o algoritmo DBSCAN foi executado com diferentes valores de atributos, “epsilon” e “minPoints”, até se tornar possível a descoberta de locais de interesse. Dois destes experimentos são detalhados na sequencia.



**Figura 7. Visualização dos tweets do Centro através da ferramenta QuantumGIS**

Para o experimento 1, foram utilizados os parâmetros: (i) minPoints = 1,25%; (ii) Epsilon = 0,011. Em relação às análises dos demais bairros, o número mínimo de pontos utilizado foi reduzido pela metade e o epsilon representou a quarta parte do valor utilizado nos experimentos com os outros bairros.

Com os parâmetros do experimento 1, foi possível identificar locais de interesse como: (i) Terminal de ônibus urbanos (TICEN) – períodos manhã e tarde nos dias de semana e noite tanto de dias de semana quanto de fins de semana; (ii) Instituto Estadual de Educação (IEE) – manhã e tarde de dias de semana; (iii) Praça XV de Novembro – manhã de fim de semana; (iv) Catedral Metropolitana de Florianópolis – tarde e noite de fim de semana; (v) Beiramar Shopping – manhã e tarde de dias de semana e fins de semana; (vi) Boate El Divino – tarde e noite de fins de semana; (vii) Boate 1007 – manhã e noite de fins de semana; (viii) Mercado Público – tarde de fins de semana; (ix) Morro da Cruz – manhã de fins de semana; (x) Instituto Federal de Santa Catarina (IF-SC) – tarde de dias de semana; (xi) Centro executivo localizado na Avenida Mauro Ramos – manhã de dias de semana, etc.



**Figura 8. Resultado parcial da execução do experimento 1**

A Figura 8 apresenta um *zoom* em parte do Centro com o resultado da execução do experimento 1. É possível identificar alguns dos locais citados, como o TICEN, a Catedral, o IF-SC, o IEE e a Praça XV.

Para o experimento 2, os parâmetros foram reduzidos radicalmente, com o objetivo de detectar um maior número de *clusters* que poderiam ser pequenos, em locais específicos. Foram utilizados os valores minPoints = 45 e epsilon = 0,0005.

Em relação à análise anterior (o experimento 1) foi observado, entre outros: (i) na Praça XV de Novembro (que foi considerada uma região densa no experimento anterior), por possuir uma área relativamente grande, não foi identificada como uma região densa, justamente por os *tweets* estarem mais distantes entre si neste local; (ii) alguns locais não detectados com análises anteriores puderam ser encontrados nesta análise, como a Universidade do Sul de Santa Catarina (UNISUL) e Terminal Rodoviário Rita Maria. Alguns clusters em locais residenciais também foram encontrados.

No mapa apresentado na Figura 9, gerado pela execução do experimento 2, é possível identificar alguns dos locais citados, como o Terminal Rita Maria e a UNISUL.



**Figura 9. Resultado parcial da execução do experimento 2**

## 6. Conclusão e Trabalhos Futuros

Mineração de dados espaço-temporais, com foco em detecção de agrupamentos, em redes sociais não é um assunto muito explorado. Esta pesquisa buscou conhecer o comportamento dos usuários do Twitter – especificamente na cidade de Florianópolis. De acordo com o conhecimento extraído, podem-se tirar conclusões do interesse da população e, analisar o que estão fazendo em determinados locais e horários. Por exemplo, donos de empresas, tendo acesso a estas informações, podem analisar a satisfação de colaboradores e/ou clientes. Este tipo de pesquisa poderá contribuir, por exemplo, com pesquisas de marketing, e consequentemente, aumentar a segurança dos resultados.

Para atender a proposta deste artigo, o código da ferramenta Weka foi adaptado. Isto tornou o *software* uma excelente ferramenta também para visualização. Desta maneira, o resultado encontrado pelo algoritmo DBSCAN pode ser melhor analisado.

Na implementação atual, pode ocorrer de um *cluster* ser formado apenas por *tweets* de um único usuário. Como trabalho futuro pode ser interessante tratar os dados

para desconsiderar SPAMs, ou impedir a formação de um *cluster* se ele não contiver um número mínimo de usuários distintos.

Também se pretende permitir ações de usuário nos mapas gerados, como limpar *clusters* já populados no mapa e aplicar filtros para visualizar somente *clusters* de interesse. Permitir também mais flexibilidade ao usuário, adicionando componentes na interface gráfica com este fim.

## Referencias

- Chae, J. Thom, D ; Bosch, H. ; Jang, Y. ; Maciejewski, R. ; Ebert, David S. ; Ertl, T. (2012) "Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition". IEEE Conference on Visual Analytics Science and Technology (VAST). p 143-152.
- Bartunov; Sigaev (2012). "Tsearch2 - full text extension for PostgreSQL". <http://www.sai.msu.su/~megera/postgres/gist/tsearch/V2/>. Acessado em 30 de dezembro de 2012.
- Ester, M.; Kriegel, H.-P.; Sander, J. and Xu, X. (1966) A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, and U. M. Fayyad, editors, Second International Conference on Knowledge Discovery and Data Mining, AAAI Press. p. 226-231.
- Lee, C-H. (2012) Unsupervised and supervised learning to evaluate event relatedness based on content mining from social-media streams. Expert Syst. Appl. 39(18), p 13338-13356 .
- Lee, C-H.; Yang, H.C.; Wen, W-S.; Weng, C-H. (2012) Learning to Explore Spatio-temporal Impacts for Event Evaluation on Social Media. ISNN (2), p 316-325.
- Link Nacional. (2011). "Script de Múltiplos Pontos", <http://www.linknacional.com.br/criar-site/2011/01/google-maps-api-multiplos-pontos-no-mapa-openinfowindowhtml>. Acessado em 30 de dezembro de 2012.
- OGC (2008) OpenGIS Standards and Specifications: Topic 5, Features. <http://portal.opengeospatial.org/modules/admin/licenseagreement.php?suppressHeaders=0&accesslicense>
- Olhar Digital UOL. (2012) "Twitter gera meio bilhão de mensagens por dia", [http://olhardigital.uol.com.br/jovem/redes\\_sociais/noticias/twitter-gera-meio-bilhao-de-tuites-por-dia](http://olhardigital.uol.com.br/jovem/redes_sociais/noticias/twitter-gera-meio-bilhao-de-tuites-por-dia). Acessado em 30 de dezembro de 2012.
- Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes Twitter users: realtime event detection by social sensors. In Proceedings of the 19th international Conference on World Wide Web - WWW '10. ACM, New York, NY, p 851-860.
- UOL Tecnologia. (2012) "Twitter passa dos 500 milhões de usuários, mas números mostram queda de microblog no Brasil", <http://tecnologia.uol.com.br/noticias/redacao/2012/07/31/twitter-passa-dos-500-milhoes-de-usuarios-mas-numeros-mostram-queda-de-microblog-no-brasil.htm>, Julho. Acessado em 30 de dezembro de 2012.
- Witten, I. and Frank, E. (2005) "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco.

# Classificação de documentos do Exército Brasileiro utilizando o algoritmo de *Naive Bayes* e técnicas de Seleção de Sentenças

Sander P. Pivetta<sup>1</sup>, Sergio L. S. Mergen<sup>1</sup>

<sup>1</sup>Campus Alegrete - Universidade Federal do Pampa (UNIPAMPA)  
Caixa Postal 810 – 97.546-550 – Alegrete – RS – Brazil

sanderpivotta@gmail.com, sergiomergen@unipampa.edu.br

**Abstract.** One of the needs of the Brazilian Army is the automated classification of documents called Boletins Internos (BIs), which must be grouped in order to produce summarized reports about the military. In this paper, we propose a solution based on the Naive Bayes classifier. To archive this goal, there is a need to select the text sentences that are related to each military, so that only those sentences are used during the training. In this sense, we propose two sentence selection heuristics that choose text blocks that appear close to the military name. The experiments show the benefits of using the Bayes classifier along with the proposed sentence selection techniques.

**Resumo.** Uma das necessidades do Exército Brasileiro é a classificação automatizada de documentos chamados Boletins Internos (BI), que devem ser agrupados a fim de gerar relatórios sumarizados a respeito de militares. Neste trabalho, propõe-se uma solução baseada no classificador Bayesiano. Além disso, é necessário identificar as sentenças que são relativas a cada militar, de modo que apenas elas sejam usadas durante o treinamento do classificador. Nesse sentido, o trabalho propõe duas heurísticas de seleção de sentenças que escolhem trechos de texto que apareçam próximas ao nome de cada militar. Os experimentos mostram os benefícios do uso do classificador bayesiano aliado às técnicas propostas de seleção de sentenças.

## 1. Introdução

A popularização do uso dos computadores teve como consequência a existência de uma maior quantidade de documentos digitais. Documentos que antes eram publicados em papel, agora passam a ser representados como sequências de bits em formatos compreendidos por computadores. Com essa mudança, tarefas que costumavam ser feitas manualmente podem ser auxiliadas por meio de abordagens computacionais automatizadas. Uma dessas tarefas envolve a classificação da informação. A classificação visa separar documentos de acordo com algum critério, o que facilita a tomada de decisões sobre os dados agrupados.

Várias organizações possuem a necessidade de classificar documentos. O Exército Brasileiro é um exemplo destas organizações, onde documentos chamados de Boletins Internos (BI) devem ser agrupados a fim de gerar relatórios sumarizados a respeito de militares. Os BIs são documentos confeccionados periodicamente que contém informações relacionadas às atividades realizadas pela instituição e pelos seus integrantes [Exército 2002]. A partir dos BIs são gerados documentos chamados de Folha de

Alterações, existindo um exemplar para cada militar, relatando o histórico referente as atividades por ele desempenhada e sobre a sua vida pessoal [Exército 2001].

Conforme as normas vigentes, encontradas em [Exército 2001], nem todos BIs possuem informações consideradas relevantes para a elaboração das Folhas de Alterações. Dessa forma, dado um militar, é preciso realizar pesquisas sobre todos os BIs produzidos durante o período de um semestre, buscando informações relativas ao militar, para analisar se estas devem ser usadas na produção das respectivas Folhas de Alterações.

Visando agilizar esta atividade, necessita-se encontrar uma forma de realizar a separação automática dos BI possuidores de informações relevantes para cada militar. Neste artigo, propõe-se que esta separação seja realizada com o emprego do aprendizado de máquina. Uma vez que dispõe-se de informações já classificadas em semestres anteriores, torna-se oportuno o emprego do Aprendizado Supervisionado [Mitchell 1997]. Mais especificamente, a tarefa será realizada através do classificador *Naive Bayes* [Mitchell 1997].

Além disso, outro problema pesquisado envolve escolher, para cada documento, quais trechos serão utilizados para realizar a tarefa de classificação. Como os BI são compostos por um conjunto de pequenas informações, referentes a assuntos e pessoas distintas, torna-se necessário identificar quais sentenças são relativas a cada militar. Conforme será demonstrado, a escolha equivocada das informações pode distorcer o treinamento do classificador, levando-o a fazer uma separação menos precisa.

O artigo está organizado da seguinte forma: na seção 2 são mencionados trabalhos que realizam a classificação textual usando o método de *Bayes* e algumas técnicas de seleção de sentenças. Na seção 3 é apresentado o método de classificação proposto. As técnicas utilizadas para realizar a seleção de sentenças são vistas na seção 4. Os experimentos realizados são descritos na seção 5. Já na seção 6 são tecidas as considerações finais.

## 2. Trabalhos Relacionados

Diversos algoritmos de aprendizado supervisionado podem ser utilizados na classificação de documentos textuais. Dentre eles, um que possui bom desempenho é o classificador *Naive Bayes*. Ele é uma algoritmo baseado no *Teorema de Bayes* que propõem uma maneira de calcular a probabilidade da ocorrência de um evento baseando-se nas probabilidades obtidas com a análise dos eventos anteriores. Um motivo de seu sucesso está na forma em que ele trata cada informação, pois estas são consideradas independentes entre si, o que diminui o espaço de busca usado para encontrar uma solução [Mitchell 1997].

O desempenho dessa técnica na classificação de texto é analisada em [Koga 2011], que compara o classificador com outros métodos de classificação existentes. O objetivo do experimento descrito envolveu a classificação automática do sujeito das frases. Para o treinamento foi utilizado um conjunto de atributos morfológicos e estruturais extraídos de frases pré-processadas. Os algoritmos foram implementados dentro do software “WEKA”<sup>1</sup>. Os resultados obtidos demonstram um melhor desempenho do classificador *Naive Bayes*, o que foi justificado pelo fato de que a maioria das informações analisadas não possuírem dependência entre si.

---

<sup>1</sup>[www.cs.waikato.ac.nz/ml/weka](http://www.cs.waikato.ac.nz/ml/weka)

Um exemplo clássico de classificação de textos em que se usa o classificador *Bayesiano* é a análise de mensagens do tipo *spam* [Silva and Vieira 2007]. Normalmente existem mensagens eletrônicas previamente categorizadas como *spans*, o que permite com que abordagens supervisionadas de classificação sejam utilizadas. No trabalho descrito em [Rabelo et al. 2011], foi verificado que em 85% dos *emails* analisados a resposta retornada foi correta, porém, na medida que o conteúdo destas mensagens ficava mais denso, a precisão da aplicação diminuía.

A seleção das sentenças a serem treinadas pelo classificador também merece destaque, uma vez que apenas algumas informações contidas em um documento podem estar relacionados ao assunto alvo da classificação. Nessa linha, o trabalho de [Goldstein et al. 1999] demonstra que a remoção de *stop words* pode melhorar a classificação, assim como a seleção de sentenças com poucas palavras. Em [McDonald and Chen 2002] desenvolve a ferramenta TXTRACTOR para realizar a sumarização de textos, realizando a separação de informações através da segmentação das informações. Para realizar uma melhor sintetização, ele separa as informações contidas nos documentos em grupos conforme as semelhanças apresentadas, facilitando o seu processamento.

É importante destacar que a seleção de sentenças tem diversas aplicações além de servir a uma etapa de pré-processamento em uma tarefa de classificação. Por exemplo, [Wang et al. 2012] procura encontrar sentenças distintas que caracterizam um tópico específico, através da análise de um *corpus* de texto. Também vale a pena mencionar que até mesmo a seleção das sentenças pode utilizar os algoritmos de aprendizado de máquina, como apresentado no trabalho de [Metzler and Kanungo 2008], que objetiva selecionar sentenças para realizar a sumarização dos documentos extraídos da *Web*.

### 3. Método de Classificação Proposto

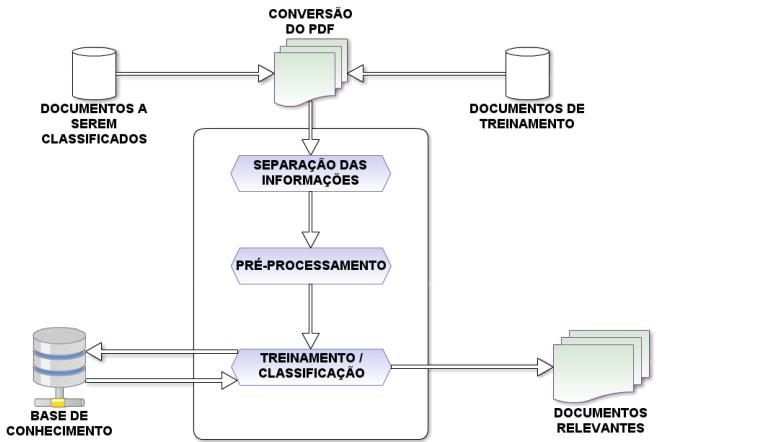
O objetivo principal deste trabalho envolve selecionar os documentos portadores de informações importantes para compor as Folhas de Alterações de cada militar. Dentre os algoritmos de aprendizado supervisionado existentes, escolheu-se o *Naive Bayes*, devido ao bom desempenho quando aplicado na classificação de documentos textuais, conforme destacado na seção anterior.

O classificador *bayesiano* divide o processamento em duas fases: o **Treinamento** e a **Classificação**. As etapas que devem ser executadas nessas duas fases, assim como as informações necessárias, são ilustradas na Figura 1. A descrição de cada um dos componentes da Figura encontra-se a seguir.

*Base de Treinamento:* Contém trechos extraídos dos BIs, sendo que os trechos já foram classificados como “relevantes” e “não relevantes”. São utilizados pelo classificador para descobrir a probabilidade das evidências estarem associadas às classes de interesse. Não são realizadas distinções com relação aos indivíduos e aos trechos que se referem.

*Documentos a Classificar:* Contém BIs que devem ser classificados como “relevantes” ou “não relevantes”. Os BIs já são previamente separados em bases menores, relacionados ao semestre em que foram confeccionados.

*Conversão do PDF:* Como os BIs encontram-se no formato PDF, é necessário realizar uma conversão para um formato textual que possa ser compreendido pelas etapas



**Figura 1. Treinamento e Classificação do algoritmo *Naive Bayes***

posteriores. Para isto foi utilizada uma biblioteca *open source* para java chamada *PDF-Box*<sup>2</sup>, que possibilita a manipulação e criação de arquivos PDF.

*Separação das Informações:* Após a leitura dos arquivos, as informações passam por uma pré-separação. Primeiramente são selecionados os arquivos que possuem alguma referência à pessoa analisada, onde são selecionados aqueles que possuem o ‘nome completo’ do militar (ex. Sander Pes Pivetta) ou a ‘graduação mais o nome de guerra’ (ex. 3º Sgt Sander). Estas informações são selecionadas devido os militares serem referenciados desta forma. Além disso, é necessário dividir o documento em trechos menores, cuja informação esteja relacionada ao militar em questão. A próxima seção descreve como essa divisão é feita.

*Pré-processamento:* Devido a língua portuguesa possuir uma grande variação morfológica, com o acréscimo de prefixos, sufixos, variação de tempos verbais, singular e plural, existe uma grande diversidade de palavras com sentidos semelhantes, o que pode interferir no processo de classificação. Na busca por diminuir a gama de palavras que irá compor a base de conhecimento, são utilizadas a técnica de *stemming* [Rezende 2005] para realizar uma normalização linguística das palavras, com a remoção das variações morfológicas descritas acima, e a técnica de remoção de *stop words* [Rigo et al. 2007], para a exclusão das palavras consideradas inúteis (preposições, artigos, numerais, pronomes e algumas palavras de contexto especificamente militar).

*Treinamento:* Para todos os trechos escolhidos, esta etapa calcula a probabilidade de ocorrência dos eventos dentro das classes analisadas (relevante e não relevante). No caso em questão, cada palavra do vocabulário presente nos trechos escolhidos corresponde a um evento.

*Base de Conhecimento:* É o resultado obtido com o treinamento, onde são armazenadas as probabilidade de cada palavra do vocabulário estar associada a classe dos “relevantes” e dos “não relevantes”.

*Classificação:* Nesta etapa, é calculada a probabilidade de um trecho pertencer a cada uma das classes. Para encontrar este valor, é usado o *Teorema de Bayes*, em que a

<sup>2</sup><http://pdfbox.apache.org>

probabilidade é calculada com base em evidências coletadas anteriormente (armazenadas na base de conhecimento). Se em pelo menos um dos trechos escolhidos do documento, a probabilidade *bayesiana* de ele ser relevante for maior, o documento é considerado relevante para o militar em questão.

*Documentos Relevantes:* São os BI retornados como relevantes pela fase de classificação, para cada militar.

#### 4. Seleção de Sentenças

De todas as etapas realizadas pelo algoritmo *Naive Bayes*, a forma como os trechos são selecionados, tanto na fase de classificação como na de treinamento, ganha destaque devido a sua influência na obtenção dos resultados. Uma inapropriada seleção das informações acarreta na utilização de dados errados para o cálculo da probabilidade e uma categorização errada dos Boletins Internos.

Neste trabalho são propostas duas técnicas para seleção dos trechos, chamadas de “Janela Fixa” e “Janela Deslizante”. Em ambas, o ponto de partida são os pontos no texto onde o nome do militar (pivô) aparece. Dado um pivô, cada técnica utiliza regras diferentes para selecionar o texto que está relacionado ao militar, como será descrito a seguir.

**Janela Fixa:** Esta técnica considera como informações importantes aquelas que encontram-se mais próximas ao nome pesquisado. Para isso, a partir do pivô, é realizada a seleção dos  $\kappa$  caracteres anteriores e posteriores ao nome. Caso a seleção selecione apenas parte de uma palavra, toda a palavra é considerada como parte integrante da sentença.

**Janela Deslizante:** Esta técnica leva em consideração que a informação importante possa estar afastada do nome pesquisado, principalmente quando ele estiver contido em uma lista de nomes.

Em primeiro lugar, deve-se verificar se o pivô encontra-se dentro de uma linha válida. Uma linha é assim considerada se o número de palavras válidas da linha for igual ou superior a  $\lambda$ . Para ser válida, a palavra deve possuir mais do que  $\mu$  caracteres.

A obtenção da linha analisa o texto anterior ao pivô até encontrar um símbolo de término de parágrafo e o posterior até encontrar outro símbolo de término de parágrafo. Caso esta seleção possua  $\lambda > 6$  ela é considerada válida e selecionada, caso contrário, ela é descartada e realizam-se verificações sobre os parágrafos anteriores a ele, até que a condição de validação do texto seja satisfeita, sendo o texto selecionado. Nos casos onde a sentença analisada não ser válida, são analisados os parágrafos anteriores pelo motivo das sentenças referentes ao militar estarem no mesmo parágrafo ou em parágrafos anteriores.

Para exemplificar as técnicas de seleção, considere as Figuras 2 e 3, que selecionam o texto tendo “Sander Pes Pivetta” como pivô. Na Figura 2 foi utilizada a “Janela Fixa” com  $\kappa = 150$ . Ou seja, foram selecionados os 300 caracteres mais próximos ao nome. Já na Figura 3 foi utilizada a “Janela Deslizante”, com  $\lambda = 6$  e  $\mu = 3$ . A escolha destes valores fundamentou-se em testes realizados, onde os resultados obtidos com o uso destes valores foram os mais eficientes.

Já as Figuras 4 e 5 mostram um outro exemplo em que o nome do militar está em uma tabela, juntamente com o nome de outras pessoas. Nesse caso, a informação

Deu entrada na Seção de Transporte Administrativo do Quartel a parte Nr 083 - Furriel, do Cmt Esqd C Ap, solicitando 01 (uma) passagem de ida de Alegrete-RS para Porto Alegre-RS e 01 (uma) passagem de volta de Porto Alegre-RS para Alegrete-RS, de acordo com o inciso V do artigo 28 do Dec nº 4.307, de 18 Jul 02, para o 3º Sgt Mnt Com **Sander** Pes Pivetta, em virtude de realização de consulta médica especializada. A partida e o retorno estão previstos para os dias 13 Jul 11 e 15 Jul 11, respectivamente (solução à nota Nr 040-STA, de 06 Jul 11);

Deu entrada na Seção de Transporte Administrativo do Quartel a parte Nr 083 - Furriel, do Cmt Esqd C Ap, solicitando 01 (uma) passagem de ida de Alegrete-RS para Porto Alegre-RS e 01 (uma) passagem de volta de Porto Alegre-RS para Alegrete-RS, de acordo com o inciso V do artigo 28 do Dec nº 4.307, de 18 Jul 02, para o 3º Sgt Mnt Com **Sander** Pes Pivetta, em virtude de realização de consulta médica especializada. A partida e o retorno estão previstos para os dias 13 Jul 11 e 15 Jul 11, respectivamente (solução à nota Nr 040-STA, de 06 Jul 11);

**Figura 2. Janela Fixa**

referentes a esses militares encontra-se no parágrafo que aparece antes da tabela. Na Figura 4 foi utilizada a “Janela Fixa” com  $\kappa = 150$ . Como pode-se ver, a técnica seleciona praticamente todas as informações contidas na tabela e ignora a sentença anterior a ela, a qual possui a informação pertinente. Já na Figura 5 foi utilizada a “Janela Deslizante”, com  $\lambda = 6$  e  $\mu = 3$ . Observa-se que, nesse caso, como a linha onde o nome do militar ocorre não é considerada válida, a janela de texto deslizou para cima até o encontro de uma linha válida.

Os militares abaixo realizaram, entre os dias 06, 07, 13 e 14 de outubro de 2011, a 2ª Chamada do 2º TAF / 2011 e obtiveram os seguintes resultados:

NOME	CORRIDA	MENÇÃO
Individuo Número Um	2800	R
Individuo Número Dois	2800	E
Individuo Número Três	2800	E
Sander Pes Pivetta	2650	MB
Individuo Número Quatro	2600	B
Individuo Número Cinco	3150	E
Individuo Número Seis	2900	MB
Individuo Número Sete	3000	B
Individuo Número Oito	3000	MB
Individuo Número Nove	3000	B

**Figura 4. Janela Fixa**

Os militares abaixo realizaram, entre os dias 06, 07, 13 e 14 de outubro de 2011, a 2ª Chamada do 2º TAF / 2011 e obtiveram os seguintes resultados:

NOME	CORRIDA	MENÇÃO
Individuo Número Um	2800	R
Individuo Número Dois	2800	E
Individuo Número Três	2800	E
Sander Pes Pivetta	2650	MB
Individuo Número Quatro	2600	B
Individuo Número Cinco	3150	E
Individuo Número Seis	2900	MB
Individuo Número Sete	3000	B
Individuo Número Oito	3000	MB
Individuo Número Nove	3000	B

**Figura 5. Janela Deslizante**

## 5. Resultados Obtidos

Nesta seção é avaliada a qualidade dos métodos propostos para a classificação de BIs disponibilizados pelo Exército Brasileiro, usando as técnicas de seleção de sentenças “Janela Fixa” e “Janela Deslizante”. Para efeitos de comparação, também é verificada a qualidade de outros dois métodos básicos, chamados de “Pesquisa Nominal”(que tem a finalidade de comparar os resultados obtidos com o emprego da seleção de sentenças) e “Documento Inteiro”(que tem a finalidade de apresentar a necessidade de realizar uma correta seleção das informações). A descrição de cada método empregado é apresentada a seguir:

**Pesquisa Nominal:** Esse método realiza a categorização dos BIs sem a aplicação

do aprendizado de máquina. Dado um militar, são considerados relevantes todos os documentos onde o seu nome ocorre.

**Documento Inteiro:** Esse método realiza o treinamento dos BIs sem utilizar a seleção de sentenças. Ou seja, caso um documento contenha informações relevantes a respeito de um militar, todas as palavras do documento também são consideradas como eventos e são associados a classe de documentos relevantes.

**Janela Fixa:** Esse método realiza o treinamento dos boletins internos utilizando a técnica de seleção de sentenças “Janela Fixa”, com  $\kappa = 150$ . Ou seja, caso um documento contenha informações relevantes a respeito de um militar, apenas as palavras que pertencem a um trecho, selecionado por essa técnica, são consideradas eventos associados a classe de documentos relevantes.

**Janela deslizante:** Esse método realiza o treinamento dos BIs utilizando a técnica de seleção de sentenças “Janela Deslizante”, com  $\lambda = 6$  e  $\mu = 3$ . Ou seja, caso um documento contenha informações relevantes a respeito de um militar, apenas as palavras que pertencem ao trecho selecionado por essa técnica são consideradas eventos, sendo associados a classe de documentos relevantes.

Na etapa de treinamento foram usados 214 BIs confeccionados durante um período de dois semestres, para 64 militares selecionados aleatoriamente. A marcação dos documentos em “relevantes” e “não relevantes” foi realizada manualmente, com o auxílio das Folhas de Alterações destes militares. Cada Folha de Alteração possui a identificação dos BIs usados para a sua confecção, o que facilitou o processo de marcação.

A base de treinamento gerada depende do método de classificação usado. No método “Documento Inteiro”, a lista de relevantes e não relevantes é composta por BIs inteiros. Por exemplo, se um BI tiver sido usado para confeccionar uma folha de alterações, todo o BI é incorporado à base de documentos relevantes. Já nos métodos “Janela Fixa” e “Janela Deslizante”, a lista de relevantes e não relevantes é composta por trechos dos BIs. Por exemplo, se um BI tiver sido usado para confeccionar uma folha de alterações, um método de seleção de sentenças é usados para extrair os trechos do documento onde o nome do militar ocorre. Os trechos são incorporados a base de documentos relevantes.

O indicador de desempenho utilizado nos experimentos é o *F-Measure*, que fornece uma medida balanceada dos escores de precisão e cobertura. Valores próximos a zero indicam que tanto a precisão quanto a cobertura foram pobres, enquanto valores próximos a 100 indicam que tanto a precisão quanto a cobertura obtiveram resultados satisfatórios. A precisão é calculada em função do número de BIs classificados como “relevantes” que realmente são. Já a cobertura é calculada em função do número de BIs relevantes que assim foram classificados.

A etapa de testes utilizou BIs e folhas de alterações de um semestre específico que não foi utilizado durante o treinamento. Os métodos de classificação foram encarregados de classificar 113 BIs para um conjunto de 27 militares selecionados aleatoriamente.

Ressalta-se que dentre os BIs empregados no treinamento e na classificação, aproximadamente 12% possuem pelo menos uma informação relevante. Os outros não possuíam nenhuma sentença relevante.

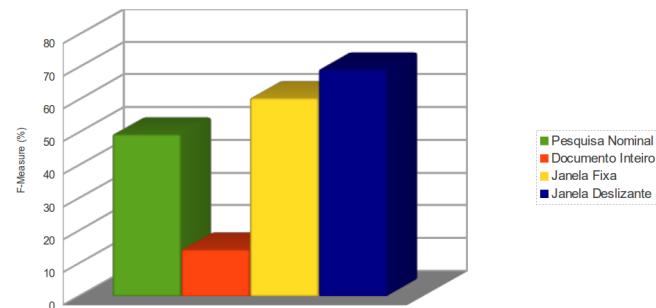
A Tabela 1 apresenta os valores de precisão, cobertura e *F-Measure* encontrados

para cada um dos métodos de classificação empregados. A Figura 6 compara os valores de *F-Measure* alcançados pela técnica de seleção.

Método	Precisão	Cobertura	Medida F
Pesquisa Nominal	33%	100%	49,6%
Documento Inteiro	11%	21%	13,7%
Janela Fixa	57%	65%	60,7%
Janela Deslizante	76%	64%	69,5%

**Tabela 1. Comparativo entre os resultados**

Observe que a “Pesquisa Nominal” atingiu uma *F-Measure* próxima a 50%. Apesar de encontrar todos os documentos relevantes, a precisão é baixa, ou seja, muitos dos documentos retornados não possuem relação para a confecção das folhas de alterações de militares específicos. Já os métodos de classificação *bayeasiana* baseados em técnicas de seleção de sentença obtiveram um desempenho superior. Dentre os dois, o método da “Janela Deslizante” se saiu melhor, atingindo uma *F-Measure* igual a 69,5%. Esse resultado reflete o fato de que, em muitos casos, o nome do militar aparece dentro de uma lista, sendo que nessas situações o método de “Janela Deslizante” consegue selecionar melhor o trecho significativo para o militar.

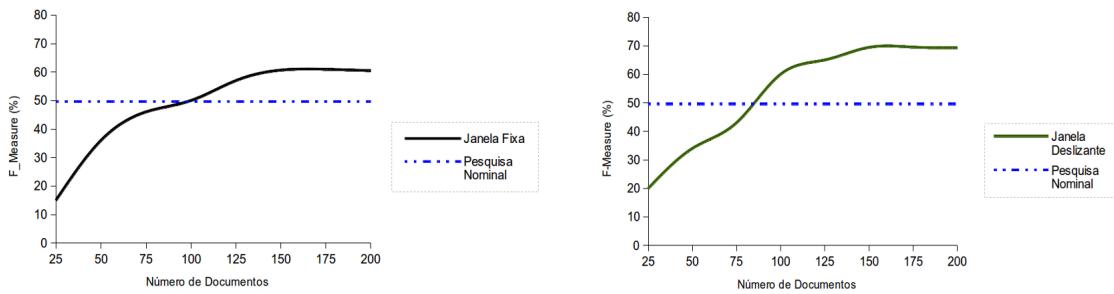


**Figura 6. Comparativo entre os resultados**

Dos quatro métodos, o “Documento Inteiro” obteve o pior desempenho, perdendo inclusive para o método “Pesquisa Nominal” que não utiliza algoritmos de aprendizado de máquina. Isso ocorre porque o treinamento leva em consideração muitos eventos (ocorrência de palavras) que não possuem relação nenhuma com os militares usados durante o treinamento.

Para realizar uma boa classificação, é necessário que o treinamento utilize uma base de conhecimento de tamanho adequado. Bases com pouca informação podem não dispor de evidências suficientes para realizar a classificação da forma correta, levando a uma situação conhecida como *underfitting*. Já bases com muita informação podem se especializar nos dados de treinamento e falhar quando novos dados precisarem ser classificados, levando a uma situação conhecida como *overfitting*.

Nesse sentido, o próximo experimento tem o intuito de verificar como o tamanho da base de treinamento afeta a classificação. A Figura 5 apresenta os resultados obtidos. Os gráficos medem a *F-Measure* obtida quando se usa bases de treinamento de tamanho variável.



**Figura 7. Variação dos resultados utilizando seleção Janela Fixa**

**Figura 8. Variação dos resultados utilizando seleção Janela Deslizante**

O gráfico mostra que o desempenho de ambos os classificadores é pior ao que usa “Pesquisa nominal” quando a base de treinamento dispõe de menos do que 75 BIs, o que caracteriza o *underfitting*. A partir de 150 BIs, o desempenho cai, mas se mantém superior a “Pesquisa Nominal”. Isso sugere que ambas técnicas de seleção de sentenças são capazes de reduzir o ruído na fase de treinamento, o que reflete em uma taxa de precisão e cobertura razoavelmente altas mesmo quando se usa uma quantidade elevada de documentos para treinar o classificador.

A leve queda percebida no desempenho indica que ainda existe um certo ruído na base de treinamento, mas que não chega a gerar o problema de *overfitting*. A explicação para isso pode derivar do fato de que os BIs são produzidos por pessoas diferentes, que escrevem de modo particular, usando um vocabulário próprio. Assim, é possível que os BIs de teste tenham sido produzidos por pessoas diferentes das que produziram os BIs de treinamento, e algumas evidências relevantes não tenham sido devidamente identificadas pelo classificador.

## 6. Conclusão

Este artigo apresentou uma aplicação de classificação de documentos que emprega o algoritmo de aprendizado de máquina supervisionado *Naive Bayes*. O objetivo da aplicação é selecionar os Boletins Internos que devem compor as Folhas de Alterações de militares do Exército Brasileiro. Para auxiliar no treinamento, também foram propostas duas técnicas de seleção de sentença, chamadas de “Janela Fixa” e “Janela Deslizante”. Essas técnicas tem a função de delimitar as palavras dos BIs que são usadas tanto na etapa de treinamento quanto na etapa de classificação.

Para validar a proposta, os métodos de classificação apresentados foram empregados na classificação de um conjunto de Boletins Internos. Analisando os resultados, verifica-se que o algoritmo de *Naive Bayes*, combinado com uma técnica de seleção de sentenças, consegue realizar uma classificação satisfatória dos documentos, comprovando que a atividade mais influente no resultado é a forma como as informações são selecionadas.

Como trabalhos futuros, pretende-se analisar o desempenho das técnicas de “Janela Fixa” e “Janela Deslizante” quando são utilizados valores diferentes para  $\kappa$ ,  $\lambda$  e  $\mu$ . Além disso, pretende-se analisar a possibilidade de descobrir os melhores parâmetros

automaticamente através de algoritmos de aprendizado de máquina baseado em redes neurais.

Outra possibilidade de trabalho futuro envolve estender o classificador *bayesiano* utilizado para que as evidências coletadas englobem a frequência de conjuntos de palavras subjacentes em vez da frequência de palavras individuais. Essa estratégia pode mostrar-se útil caso existam sequências de palavras que costumem aparecer mais frequentemente em documentos de uma certa classe (relevante ou não relevante). Além de implementar essa versão estendida, pretende-se descobrir se existe um tamanho adequado para a sequências de palavras que maximize a *F-Measure*.

## Referências

- Exército (2001). *Boletim do Exército 02*. Secretaria Geral do Exército, Brasília.
- Exército (2002). *Separata ao Boletim do Exército Número 08: Instruções Gerais para a Correspondência, as Publicações e os Atos Administrativos no Âmbito do Exército (IG 10-42)*. Gabinete do Comandante do Exército, Brasília.
- Goldstein, J., Kantrowitz, M., Mittal, V., and Carbonell, J. (1999). Summarizing text documents: sentence selection and evaluation metrics. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, New York. ACM.
- Koga, M. L. (2011). Classificadores Bayesianos: Aplicados a análise sintática da língua portuguesa. In *Escola Politécnica da Universidade de São Paulo*, São Paulo.
- McDonald, D. and Chen, H. (2002). Using sentence-selection heuristics to rank text segments in txtractor. In *Joint Conference on Digital Libraries - JCDL*, New York.
- Metzler, D. and Kanungo, T. (2008). Machine learned sentence selection strategies for query-biased summarization. In *SIGIR Learning to Rank Workshop*.
- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math.
- Rabelo, J. P., Filho, M. A., and Oliveira, T. (2011). Mineração de Textos Através do Algoritmo de Classificação. In *Instituto de Matemática. Universidade Federal da Bahia (UFBA)*, Salvador.
- Rezende, S. O. (2005). *Sistemas Inteligentes. Fundamentos e Aplicação*. Editora Manole Ltda, Barueri.
- Rigo, S. J., Oliveira, J. P. M., and Barbieri, C. (2007). Classificação de Textos Baseada em Ontologias de Domínio. In *Anais do XXXVII Congresso da Sociedade Brasileira de Computação - V Workshop em Tecnologia da Informação e da Linguagem Humana*, Rio de Janeiro.
- Silva, C. F. and Vieira, R. (2007). Categorização de Textos da Língua Portuguesa com Árvores de Decisão, SVM e Informações Linguísticas. In *Anais do XXVII Congresso da Sociedade Brasileira de Computação. V Workshop de Tecnologia da Informação e da Linguagem Humana*, Rio de Janeiro.
- Wang, D., Zhu, S., Li, T., and Gong, Y. (2012). Comparative document summarization via discriminative sentence selection. *ACM Transactions on Knowledge Discovery from Data*.

# **Modelo de Banco de Dados Colunar: Características, Aplicações e Exemplos de Sistemas**

**Bruno Eduardo Soares, Clodis Boscarioli**

Centro de Ciências Exatas e Tecnológicas – Universidade Estadual do Oeste do Paraná  
(UNIOESTE)

Av. Universitária, 2069 – Bairro Faculdade – 85819-110 – Cascavel – PR – Brasil

[besoares90@gmail.com](mailto:besoares90@gmail.com), [clodis.boscarioli@unioeste.br](mailto:clodis.boscarioli@unioeste.br)

*Abstract. This paper describes some aspects of the NoSQL databases, in particular the columnar approach, presenting its main characteristics and advantages in relation to the linear storage method, analyzing its architecture and concepts applied on this storage model. The main advantages of columnar models are in compression, materialization and it helps the analysis of iteration blocks. This model is a tendency in different applications that require high availability.*

**Resumo.** Este artigo discute alguns aspectos dos bancos de dados no modelo NoSQL, em específico na abordagem colunar, apresentando suas principais características e vantagens em relação ao método de armazenamento em linhas, analisando sua arquitetura e conceitos aplicados neste modelo de armazenamento. As principais vantagens do modelo colunar estão na compressão, na materialização, o que ajuda na análise de blocos de iteração. Este modelo é uma tendência em diferentes domínios de aplicação que exijam alta disponibilidade.

## **1. Introdução**

O armazenamento de informações sempre foi um fator fundamental ao se trabalhar com dados de forma digital. Os SGBDs relacionais vêm sendo amplamente utilizados desde sua concepção por serem de fácil manipulação e possuírem recursos que garantem a integridade dos dados, a exemplo das restrições de chave. No entanto, haja vista o crescimento e diversidade de informações geradas a todo tempo, o modelo de armazenamento em linhas nem sempre é o mais recomendado, principalmente, quando grandes quantidades de dados devem ser tratadas.

Os bancos de dados NoSQL podem ser vistos como alternativa para trabalhar com esses casos, por possuírem arquitetura diferenciada e seguirem conceitos diferentes, de forma a facilitar o tratamento com alta escalabilidade de dados. Grandes empresas, por exemplo, a Google, o Facebook e o Twitter adotaram a utilização de bancos de dados NoSQL, mais especificamente de modelo colunar, devido a grande demanda por consultas, vinculado ao elevado número de informações que manipulam.

Esse artigo traz uma discussão das principais características do modelo colunar e suas diferenças em relação ao modo de armazenamento do modelo relacional, apresentando sistemas que implementam esse modelo, e está estruturado da seguinte

forma: A Seção 2 introduz o conceito NoSQL, apresentando definições, características e modelos de bancos de dados que nele se enquadram. A Seção 3 descreve o modelo colunar de banco de dados, bem como sua arquitetura e as vantagens que podem ser obtidas sobre o modelo relacional. Por fim, a Seção 4 traz considerações e perspectivas do modelo colunar de banco de dados.

## 2. NoSQL

Marcus (2011) define o termo NoSQL como “*um sistema que apresenta uma interface de consulta que não é apenas SQL*”. Em Han *et al.* (2011) é afirmado que os bancos de dados NoSQL surgiram para satisfazer necessidades como: (i) Armazenamento de grandes volumes de dados de forma eficiente e requerimentos de acesso; (ii) Alta escalabilidade e disponibilidade; e, (iii) Menor custo operacional e de gestão.

A utilização de bancos de dados NoSQL se dá, principalmente, devido ao aumento gradativo de informações a serem armazenadas, no qual o desempenho é prejudicado e o tempo de resposta se torna um fator preocupante. Pritchett (2008) afirma que quando um banco de dados cresce além de sua capacidade em um único nó (servidor) é necessário optar por escalabilidade horizontal (paralelismo) ou vertical (reforçar o servidor).

NoSQL não é apenas mais um modelo de banco de dados, mas sim, um termo que define uma classe de modelos, sendo, conforme mostra literatura, os mais comuns e aplicados [Hecht e Jablonski, 2011]:

- Orientado a chaves: A estrutura desse modelo é como em uma tabela *Hash*, ou seja, há diversas chaves na tabela, cada qual referenciando um valor (por valor entende-se um tipo de dado). São exemplos de SGBD que suportam esse modelo: RIAK [Basho, 2012] e MemcacheDB [MemcacheDB, 2009].
- Orientado a colunas: Também chamado de modelo colunar, utiliza-se de tabelas para representação de entidades, e os dados são gravados em disco, agrupados por colunas, o que reduz o tempo de leitura e escrita em disco [Matei, 2010], [Abadi, Madden e Hachem, 2008], [Scalzo, 2010].
- Orientado a documentos: Similar ao modelo orientado a chaves, podendo gerar uma chave secundária para indexar seu valor. Exemplos de SGBD que suportam esse modelo o MongoDB [MongoDB, 2012] e o CouchDB [Apache, 2012b].
- Baseados em grafos: Os dados são armazenados em nós de um grafo cujas arestas representam o tipo de associação entre esses nós. São exemplos dessa abordagem o Neo4j [Neo4j, 2012], o GraphDB [GraphDB, 2012] e o InfoGrid [InfoGrid, 2012].

Uma das características chave de NoSQL é a habilidade de realizar escalonamento horizontal (particionamento do banco de dados) [Cattel, 2010], pois trabalham com dados denormalizados, e se tornam uma alternativa para substituição dos bancos relacionais em situações em que o desempenho é afetado pela escalabilidade. O modelo colunar vem sendo amplamente aplicado na substituição da metodologia de armazenamento em linhas, devido a ambos tratarem do mesmo tipo de dados,

trabalharem bem com SQL [Mcknight, 2011] e pela sua superioridade em desempenho com grandes quantidades de dados para certas aplicações.

Além de desempenho, há que considerar o Modelo CAP (*Consistency, Availability, Partition Tolerance* – Consistência, Disponibilidade e Tolerância à Partição) descrito em Gilbert e Lynch (2002), que versa que apenas dois dos três itens abaixo podem ser satisfeitos concorrentemente em um modelo de banco de dados:

- Consistência: percepção de que um conjunto de operações ocorreu de uma só vez;
- Disponibilidade: cada operação deve terminar em uma resposta destinada;
- Tolerância à partição: operações serão completadas, mesmo se componentes individuais estiverem indisponíveis.

Seguindo a ideia do Modelo CAP, como a flexibilidade em escalonamento horizontal é promovida pelo modelo NoSQL, é necessário escolher entre consistência ou disponibilidade como segundo fator. O modelo relacional segue o conceito ACID (Atomicidade, Consistência, Isolamento e Durabilidade). No entanto, Pritchett (2008) questiona que se este garante consistência para bancos de dados particionados, então a disponibilidade é deixada em segundo plano (pelo Modelo CAP). Em contraposição a isso, o autor propôs o Modelo BASE (*Basically Available, Soft State, Eventual Consistency*), sugerindo que o ACID é pessimista e força a consistência no final de cada operação, enquanto o BASE é otimista e aceita que a consistência no banco de dados estará em um estado de fluxo, ou seja, não ocorrerá no mesmo instante, gerando uma “fila” de consistência de posterior execução.

A disponibilidade do Modelo BASE é garantida tolerando falhas parciais no sistema, sem que o sistema todo falhe. Por exemplo, se um banco de dados está particionado em cinco nós e um deles falha, apenas os clientes que acessam aquele nó serão prejudicados, pois o sistema como todo não cessará seu funcionamento.

A consistência pode ser “relaxada” permitindo que a persistência no banco de dados não seja efetivada em tempo real (ou seja, logo depois de realizada uma operação sobre o banco). Pelo ACID, quando uma operação é realizada no SGBD (a exemplo *insert, update e delete*), ela só será finalizada se houver a certeza de que a persistência dos dados foi realizada no mesmo momento. Já no BASE isso não se confirma. Para garantir a disponibilidade, algumas etapas são dissociadas à operação requisitada, sendo executadas posteriormente. O cliente realiza uma operação no banco de dados e, não necessariamente, a persistência será efetivada naquele instante.

Para Pritchett (2008), o Modelo BASE pode elevar o sistema a níveis de escalabilidade que não podem ser obtidos com ACID. No entanto, algumas aplicações necessitam que a consistência seja precisamente empregada. Nenhuma aplicação bancária poderá por em risco operações de saque, depósito, transferência, etc. O projetista do banco de dados deverá estar ciente de que se utilizar o Teorema BASE estará ganhando disponibilidade em troca de consistência, o que pode afetar os usuários da aplicação referente ao banco de dados.

A escolha entre ACID ou BASE dependerá do tipo de aplicação com que se irá trabalhar. A utilização do Teorema BASE não é padrão nos SGBD NoSQL. Alguns

seguem o ACID, outros aplicam conceitos referentes ao BASE, e há também os que permitem o administrador de banco de dados optar entre um ou outro, como é o caso do Cassandra [Apache, 2012a].

### **3. O Modelo Colunar de Banco de Dados**

De acordo com Abadi, Boncz e Harizopoulos (2009), o modelo colunar de armazenamento mantém cada coluna do banco de dados separadamente, guardando contiguamente os valores de atributos pertencendo à mesma coluna (Figura 1(b)), de forma densa e comprimida. Essa forma de armazenamento pode beneficiar a leitura dos dados, porém, comprometendo a escrita em disco.

Orientado a Linhas	Orientado a Colunas
Joao   2432.00   1988   Rio de Janeiro	Joao   Maria   Pedro   Jorge
Maria   2511.00   1986   São Paulo	2432.00   2511.00   3500.00   4200.00
Pedro   3500.00   1976   Mato Grosso	1988   1986   1976   1930
Jorge   4200.00   1930   Paraná	Rio de Janeiro   São Paulo   Mato Grosso   Paraná

**Figura 1.** Forma de armazenamento em (a) linhas e (b) colunas

As principais vantagens que se pode obter com a arquitetura de armazenamento dos bancos de dados de modelo colunar em relação aos de modelo relacional estão vinculadas à compressão, materialização e bloco de iteração, abaixo descritos.

### 3.1. Compressão

Existem diversos métodos de compressão que podem ser utilizados em bancos de dados, dentre os quais estão os apresentados por Ziv e Lempel (1977), um algoritmo universal para compressão de dados sequenciais; Cormak (1985), um método de compressão para operações relacionais; Westmann et al. (2000), que discutem quatro algoritmos simples de compressão, porém bastante interessantes, sendo eles compressão numérica, compressão de *strings*, compressão baseada em dicionário e compressão de valores nulos; e, Zukowski et al. (2006), um algoritmo de compressão desenvolvido para a escalabilidade de processadores modernos, utilizando a memória RAM como cache.

No entanto, nem todos os métodos são aplicáveis para todas as arquiteturas presentes nos bancos de dados, principalmente em se tratando de bancos de dados colunares. O ganho no desempenho da compressão depende não só apenas das propriedades dos dados, mas também da forma com que o processador de consultas manipula os atributos [Abadi, Madden e Ferreira, 2006].

No caso do processador de consultas, é necessário que este contenha um mecanismo que permita manipular os dados da forma em que foram comprimidos, ou mesmo que possa realizar uma descompressão para recuperar as informações no formato original, para daí estar apto a manipulá-las. Caso o processador de consultas realize uma descompressão antes de operar sobre os dados, o ganho obtido será medido não só pela redução de espaço em disco, mas também pelo tempo que levará para realizar essa

operação. Métodos que gastam tempo demais em descompressão podem não ser interessantes, em particular quando o processamento sobre os dados é muito afetado.

A forma como os dados são armazenados em disco também influencia na qualidade da compressão. Bancos de dados de modelo colunar possuem vantagens sobre bancos de dados de modelo relacional nessa questão. Abadi (2008) exemplifica esse fato sugerindo que, pelo método de armazenamento colunar guardar informações por colunas, elas se tornam mais aptas à compressão, por informações semelhantes serem armazenadas sequencialmente.

De uma maneira geral, métodos de compressão são muito úteis e capazes de incrementar consideravelmente o desempenho de um SGBD. No entanto, alguns métodos de compressão devem ser modificados para adaptar-se bem a certos SGBDs, devido às suas diferenças de arquiteturas. Como dito, dois principais componentes que influenciam no desempenho da compressão são o executor de consultas e a camada de armazenamento do SGBD. O executor de consultas deve estar apto a trabalhar com dados comprimidos ou conseguir descompactá-los para recuperar as informações originais e a camada de armazenamento deve substituir informações comprimidas por referências que possam transmitir a essência dos dados descomprimidos.

A compressão de dados é um fator fundamental ao trabalhar com uma grande quantidade de dados devido à redução de espaço em disco e melhor desempenho. Vários SGBDs colunares aplicam métodos de compressão apresentando bons resultados em redução de espaço em disco. Esse fator se tornou um dos quesitos principais na escolha de um SGBD, principalmente para grandes empresas. Os SGBDs de modelo colunar podem comprimir informações com uma proporção maior que os de modelos relacionais, tornando-se esta uma importante vantagem sobre SGBDs desse modelo.

### 3.2. Materialização

Da mesma forma que existe uma metodologia para armazenar os dados em disco nos bancos de dados, também deve existir outra que recupere as informações armazenadas e as transforme novamente em *tuplas*. Essa operação é chamada de materialização ou reconstrução de *tuplas*. Abadi et al. (2007) apresentam duas formas de materialização:

- *Early Materialization (EM)*: Consiste em adicionar uma coluna a uma *tupla* intermediária de saída, caso a coluna acessada seja requerida posteriormente por algum operador ou esteja incluída na saída da consulta. Essa é a metodologia adotada pelo modelo relacional de banco de dados.
- *Late Materialization (LM)*: Consiste em não adicionar a coluna no mesmo instante em que é requerida. O executor de consultas espera um instante para que, primeiramente, cada predicado seja aplicado à sua respectiva coluna, e uma lista para cada coluna contendo as posições que atenderam ao predicado é gerada. Cada *i*-ésimo valor de cada coluna é comparado e apenas os *i*-ésimos atributos que atenderam a todos os predicados são adicionados à *tupla* de resposta. Essa metodologia é a adotada no modelo colunar de banco de dados.

O modelo colunar de dados se beneficia na utilização da segunda abordagem, enquanto no relacional se obtém melhor desempenho com a primeira. Utilizando a *Late Materialization*, o modelo colunar é apto a reconstruir, em tempo hábil, um número

menor de *tuplas* para retorno do que na outra abordagem, pois os predicados da consulta são verificados antes do retorno da coluna, para que linhas desnecessárias não sejam analisadas.

Considere o exemplo onde se têm a tabela *Aluno* (Figura 2) e deseja-se recuperar o CPF e o RG dos alunos que possuem o número da matrícula maior que 2 e o nome começando com a letra ‘A’. As Figuras 3 e 4 exemplificam os métodos de materialização *EM* e *LM*, respectivamente.

TABELA ALUNO			
Matrícula	CPF	RG	Nome
1	678.Y31.X33-40	4.03X.894 Y	Adriano P. S.
2	6X4.616.Y86-83	2.Y77.26X	Roberto M. G.
3	483.704.8Y5-3X	91.X2Y.53Z 1	Augusto S. F.
4	Y87.828.28X-63	8.456.XY2 Z	Mariana F. B.
5	711.2X2.Y78-28	12.37X.Y23 4	Carlos A. E.
6	1XY.064.415-12	6.472.33X Y	Amanda C.

Figura 2. Exemplo de Instâncias de uma Tabela Aluno

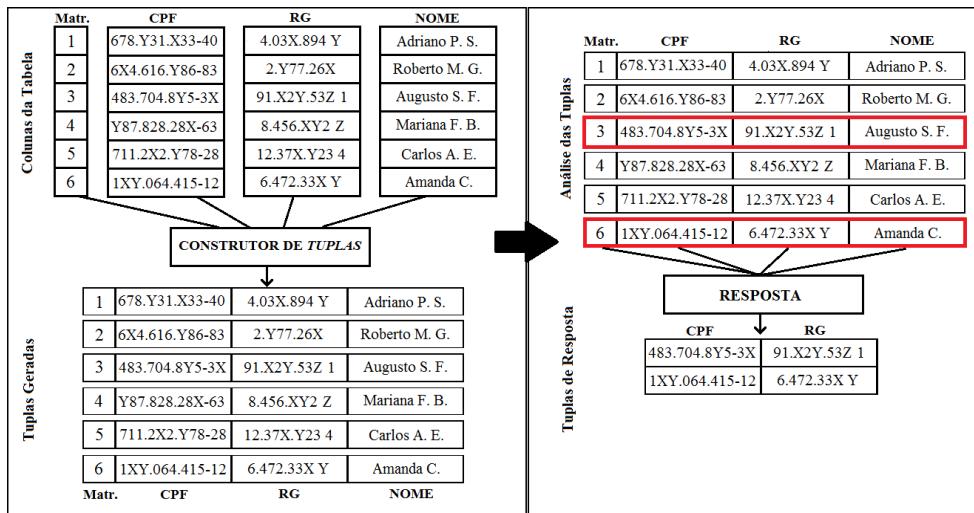


Figura 3. Tuplas geradas sobre o esquema da Figura 2 utilizando *EM*

Pela Figura 3 nota-se que foi necessário gerar uma *tupla* para cada linha da tabela, mesmo para as linhas que não atenderam ao predicado, pois a verificação do predicado é feita após a montagem das *tuplas*. É importante notar que, embora todas as *tuplas* tenham sido montadas nem todas aparecerão na saída. Isso ocorre, pois esse método de materialização primeiramente monta as *tuplas* a partir de cada coluna requisitada na consulta e só depois verifica os predicados, descartando as desnecessárias à consulta (que não atenderam ao predicado).

Na Figura 4 é mostrado que isso não ocorre utilizando a materialização *LM*. Primeiramente, as colunas referentes ao predicado são analisadas, verificando então quais são as posições que o atendem. A partir das posições obtidas é que as *tuplas* são construídas, não necessitando criar *tuplas* desnecessárias.

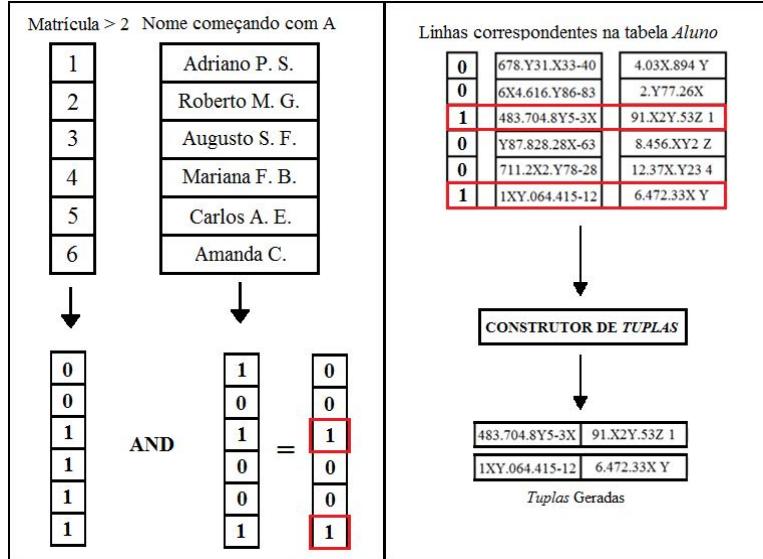


Figura 4. Tuplas geradas sobre o esquema da Figura 2 utilizando LM

### 3.3. Iteração de Bloco

Segundo Zukowski et al. (2005), o executor de consultas de SGBDs relacionais como o MySQL 4.1 [Oracle, 2012] necessitam de tempo demais para interpretar *tuplas* antes de fornecer o resultado de uma consulta. Isso ocorre porque todas as *tuplas* devem ser percorridas e interpretadas uma-a-uma, lendo atributos de forma desnecessária e utilizando um alto número de instruções de CPU para verificar todos os registros.

Os SGBDs de modelo colunar possuem uma estrutura que minimiza este problema. Dado que armazenam em disco os elementos de cada coluna de forma contígua, é possível armazenar todos os valores de uma coluna em um vetor e manda-lo para o executor de consultas com apenas uma instrução de CPU. Dessa forma, o executor de consultas pode operar sobre o vetor contendo todos os elementos da coluna lida de uma só vez, ao invés de requisitar uma instrução para cada *tupla*.

A Tabela 1 traz os principais SGBDs que adotam o modelo colunar. Não há um padrão com relação à linguagem de consultas, interfaces, sistemas operacionais suportados, e outras características técnicas, além da dificuldade em minerar tais características técnicas, que diferem muito em formato e dimensão, nas documentações disponíveis sites dos fabricantes. Por exemplo, MonetDB descreve como características principais ACID, particionamento vertical, permite execução paralelizada, segue o padrão SQL 2003 e possui índice Hash. O Infobright garante ACID e alta escalabilidade, segue o ANSI SQL-92 com algumas extensões do SQL-99. O Cassandra tem controle de concorrência multiversão, opção de escolha entre consistência e disponibilidade e possui uma linguagem de consulta própria, a CQL (*Cassandra Query Language*). Vectorwise suporta ACID, processamento massivo paralelo e alta disponibilidade, com linguagem SQL (não informa o padrão). BigTable faz uso do sistema de arquivo distribuído *Google File System* para garantir confiabilidade e disponibilidade. HBase assegura consistência, persistência e é tolerante à partição. InfiniDB garante ACID, controle de concorrência multiversão e massivo processamento paralelo e SQL - não

informado o padrão. C-Store, tem alta disponibilidade e tolerante a Partição, usa SQL (não diz o padrão) e implementa Bit map como estrutura de índice. Vertica assegura alta disponibilidade e massivo processamento paralelo e SQL - não informa o padrão. Sybase IQ, tem as mesmas características do Vertica, além de estrutura de índice em bits.

**Tabela 1. Exemplos de SGBD no modelo colunar**

SGBD/Licença	Interfaces	Site do Projeto
MonetDB <i>Open Source</i>	SQL, JDBC, ODBC, PHP, Python, RoR, C/C++ e Pearl	<a href="http://www.monetdb.org">http://www.monetdb.org</a>
Infobright <i>Community e Enterprise</i>	JDBC, ODBC, C, C++, C#, dbExpress (Borland Delphi), Eiffel, SmallTalk, Lisp, Pearl, PHP, conector nativo do Java, Python, Ruby, REALbasic, FreeBasic e Tcl	<a href="http://www.infobright.com">http://www.infobright.com</a>
Cassandra <i>Open Source</i>	Java, CQL, JDBC, DB-API 2.0, CLI, (Python), PDO (PHP) e DBI-compatible (Ruby)	<a href="http://cassandra.apache.org/">http://cassandra.apache.org/</a>
Vectorwise <i>Enterprise</i>	SQL, JDBC, ODBC e .NET	<a href="http://www.actian.com/products/vectorwise">http://www.actian.com/products/vectorwise</a>
BigTable Proprietária	C++, Python, Java	Não disponível
HBase <i>Open Source</i>	SQL, JDBC, possíveis por meio da API HBql disponível em: <a href="http://hbql.com">http://hbql.com</a>	<a href="http://hbase.apache.org/">http://hbase.apache.org/</a>
Metakit <i>Open Source</i>	C++, Mk4py (Python) e Mk4tcl (Tcl)	<a href="http://equi4.com/metakit/">http://equi4.com/metakit/</a>
InfiniDB <i>Community e Enterprise.</i>	JDBC, ODBC, ADO.NET, C/C++, PHP, Pearl, Python e Ruby	<a href="http://infinidb.org">http://infinidb.org</a>
C-Store <i>Open Source</i>	C/C++, SQL	<a href="http://db.csail.mit.edu/projects/cstore/">http://db.csail.mit.edu/projects/cstore/</a>
Widebase <i>Open Source</i>	C++, Erland, Go, Haskell, Java, PHP, Python, Ruby e Scala	<a href="http://widebase.github.com/">http://widebase.github.com/</a>
Vertica <i>Enterprise</i>	JDBC, ODBC e ADO.Net	<a href="http://www.vertica.com/">http://www.vertica.com/</a>

#### 4. Conclusões

O modelo colunar possui vantagem em três pontos principais: (i) na compressão, por ter a capacidade de organizar informações semelhantes de forma contígua; (ii) na materialização, por não precisar processar *tuplas* desnecessárias; (iii) no bloco de iteração, por ser capaz de analisar todos os valores de uma coluna com um número menor de instruções de CPU.

Para a recuperação de dados, pode-se dizer que o modelo orientado por linhas é mais eficiente quando muitas colunas de uma tupla são requisitadas e quando a tupla é pequena, tal que o conteúdo possa ser todo recuperado em uma única operação de leitura de disco. O modelo orientado a colunas é mais eficiente quando apenas algumas

colunas de cada tupla precisam ser computadas e também quando se deseja substituir apenas instâncias de algumas colunas em todas as tuplas.

Harizopoulos et al. (2006) e Stonebreaker et al. (2005) discutem a questão de bancos de dados colunares serem otimizados à leitura (*read-optimized*), enquanto bancos de dados orientados a linhas são otimizados à escrita (*write-optimized*), tornando-os uma boa alternativa às aplicações que possuem grande densidade de dados e que são frequentemente requeridos para leitura. *Data warehouses* (DW) são exemplos desse tipo de aplicação, pois possuem imensa quantidade de informações e que são requeridas a todo tempo e para estes, o modelo colunar pode ser de grande valia.

A partir dessas considerações, e, considerando que diferentemente do modelo em linhas, os sistemas não seguem uma padronização base no modelo colunar, torna-se interessante avaliar em que pontos os bancos de dados de modelo colunar são melhor aplicáveis que os de modelo orientado a linhas. Diversos SGBD de modelo colunar possuem suporte a SQL (*Structured Query Language*) e gerenciam informações de forma muito semelhante a outros SGBD de modelo orientado a linhas, podendo assim, haver uma comparação adequada entre os modelos.

## Referências

- ABADI, D. J. *Query Execution in Column-Oriented Database Systems*. Tese de Doutorado. Massachussets Institute of Technology, MA, Fevereiro, 2008.
- ABADI, D. J., BONCZ, P. A., HARIZOPOULOS, S. Column-Oriented Database Systems. In: *Proceedings of the VLDB Endowment*. 2009, v. 2, n. 2, p.1664-1665.
- ABADI, D. J., MADDEN, S. R., HACHEM, N. Column-Stores vs. Row-Stores: How Different Are They Really? In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of data*. New York, NY, USA: ACM, 2008, p. 967-980.
- ABADI, D. J., MYERS, D. S., DEWITT, D. J., MADDEN, S., R. Materialization Strategies in a Column-Oriented DBMS. In: *IEEE 23<sup>rd</sup> International Conference on Data Engineering*. Istanbul, 2007, p. 466-475.
- APACHE SOFTWARE FOUNDATION. *Cassandra*, 2012. <http://cassandra.apache.org/>. Consultado na Internet em: 10/03/2012.
- APACHE SOFTWARE FOUNDATION. *CouchDB*, 2012. <http://couchdb.apache.org/>. Consultado na Internet em: 10/03/2012.
- BASHO. *Riak*, 2012. <http://redis.io/>. Consultado na Internet em: 10/03/2012.
- CATEL, R. Scalable SQL and NoSQL Data Stores. *ACM SIGMOD Record*, New York, NY, EUA, v. 39, n. 4, p. 12-27, Dezembro, 2010.
- CORMACK, G. V. Data Compression in Database System. *Communications of the ACM*, New York, NY, USA, vol. 28, no. 12, p. 1336-1342, Dezembro, 1985.
- GRAPHDB. <http://www.sones.com>. Consultado na Internet em: 20/09/2012.
- GILBERT, S. LYNCH, N. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web-Services. *ACM SIGACT News*, New York, NY, USA, v.33, n. 2, p.51,59, Junho, 2002.

- HAN, J., HAIHONG, E., GUAN LE; JIAN DU. Survey on NoSQL database. *Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on*, 2011. p. 363-366, Dezembro, 2011.
- HARIZOPOULOS, S., LIANG V., ABADI D. J., MADDEN S. Performance Tradeoffs in Read-Optimized Databases. In: *Proceedings of the 32<sup>nd</sup> International Conference on Very Large Data Bases*. Seoul, Korea: VLDB Endowment, 2006, p. 487-498.
- HECHT, R., JABLONSKI, S. NoSQL Evaluation: A Use Case Oriented Survey. In: *Proceedings of the 2011 International Conference on Cloud and Service Computing*, Hong Kong, China, 2011, p. 336-341.
- INFOGRID. <http://infogrid.org/blog/category/nosql/>. Consultado na Internet em: 20/09/2012.
- MARCUS A. The Architecture of Open Source Applications - Elegance, Evolution, and a Fearless Hacks, Capítulo 13: The NoSQL Ecosystem, Editora Kindle, EUA, 2011.
- MATEI, G. Column-Oriented Databases, an Alternative for Analytical Environment. *Database Systems Journal*, Bucharest, Romania, v. 1, n. 2, p. 3-16, Fevereiro, 2010.
- MCKNIGHT, W. *Best Practices in the Use of Columnar Databases*, 2011. [http://www.calpont.com/doc/Calpont\\_Whitepaper-Best-Practices-in\\_the\\_Use\\_of\\_Columnar\\_Databases.pdf](http://www.calpont.com/doc/Calpont_Whitepaper-Best-Practices-in_the_Use_of_Columnar_Databases.pdf). Consultado na Internet em: 16/03/2012.
- MEMCACHEDB. *MemcacheDB*, 2009. <http://memcachedb.org/>. Consultado na Internet em: 10/03/2012.
- MONGODB. <http://www.mongodb.org/>. Consultado na Internet em: 10/03/2012.
- NEO-LJ. <http://neo-lj.org>. Consultado na Internet em: 05/01/2013.
- ORACLE. MySQL – *The World's most popular Open Source Database*, 2012. <http://www.mysql.com/>. Consultado na Internet em: 10/03/2012.
- PRICHETT, D. Base: An Acid Alternative, *Queue – Object-Relacional Mapping*, Nova York, NY, EUA, v. 6, n. 3, p. 50-55, Maio/Junho, 2008.
- SCALZO, B. *Data Warehouse Benchmark: Comparing Calpont InfiniDB® and a Row Based Database*, 2010. <http://www.calpont.com/data-warehouse-benchmark>. Consultado na Internet em: 16/03/2012.
- WESTMANN, T., KOSSMANN, D., HELMER, S., MOERKOTTE, G. The Implementation and Performance of Compressed Databases. *ACM Sigmod Record*, New York, NY, USA, vol. 29, no. 3, p. 55-67, Setembro, 2000.
- ZIV, J., LEMPEL, A. A Universal Algorithm for Sequencial Data Compression. *IEEE Transactions on Information Theory - TIT*, vol. 23, n. 3, p. 337-343, Maio, 1977.
- ZUKOWSKY, M., BONCZ, P. NES, N., HÉMAN, S. MonetDB/X100 – A DBMS In the CPU Cache. *IEEE Data Eng. Bull*, vol. 28, n. 2, p. 17-22, Agosto, 2005.
- ZUKOWSKY, M., HÉMAN, S., NES, N., BONCZ, P. Super-Scalar RAM-CPU Cache Compression. In: *Proceedings of the 22nd International Conference on Data Engineering*, 2006. Washington, DC, USA: IEEE Computer Society, 2006, p. 59.

# **Uma Proposta de *Entity Ranking* Baseada no Uso de Entidades como Objetos de Consulta**

**Thiago C. Krug<sup>1</sup>, Sergio L. S. Mergen<sup>1</sup>**

<sup>1</sup>Campus Alegrete - Universidade Federal do Pampa (UNIPAMPA)  
97.546-550 – Alegrete – RS – Brasil

thiagockrug@gmail.com, sergiomergen@unipampa.edu.br

**Abstract.** This article explores the process of *Entity Ranking* based on relationships, with Wikipedia as a data source. We propose a graph model to represent the relationships between entities. The ranking is calculated based on the number of relationships between entities and their popularity. Furthermore, a graph compression technique is employed which is particularly useful for the type of structure used by Wikipedia. The effectiveness of the ranking is described through the analysis of answers for popular entities queries.

**Resumo.** Este artigo explora o processo de *Entity Ranking* baseado em relacionamentos, tendo a Wikipedia como fonte de dados. É proposto um modelo de grafos para representar os relacionamentos entre as entidades. O ranking é calculado com base no número de relacionamentos entre as entidades e na popularidade que elas possuem. Além disso, é empregada uma técnica de compressão de grafos que é particularmente útil para o tipo de estrutura utilizada pela Wikipedia. A efetividade do ranking é descrita através de análises das respostas obtidas para consultas a entidades populares.

## **1. Introdução**

A área de *Entity Ranking* surgiu recentemente como um campo de pesquisa que visa recuperar entidades nomeadas como respostas a consultas. Exemplos de entidades incluem organizações, pessoas, localidades e datas. Esse problema difere da extração de entidades, onde o foco está na habilidade de identificar as entidades existentes em documentos usando técnicas de processamento de linguagem natural e treinamento a partir de grandes coleções de documentos. Já o processo de *Entity Ranking* foca na recuperação de uma lista ordenada de entidades que possuam relação a uma consulta por palavras-chave.

O interesse por essa área de pesquisa se acentuou a partir da constatação de que uma considerável fração das buscas na *Web* referenciavam entidades nomeadas [Pașca 2007]. Além disso, existem domínios de sites na *Internet* que podem ser considerados grandes repositórios de entidades, o que encoraja a criação de aplicações que utilizem esse tipo de informação. Um dos exemplos mais significantes é a *Wikipedia*.

A forma como os dados estão estruturados na *Wikipedia* a torna um recurso poderoso que pode ser explorado para muitos tipos de análise de dados, como desambiguação de palavras [Mihalcea 2007], recuperação de informação e resposta a consultas [Ahn et al. 2004]. Alguns dos trabalhos de pesquisa que utilizam a *Wikipedia* como fonte de dados trata do problema conhecido como *Entity Ranking*, que envolve encontrar as melhores entidades como resposta a consultas. Ao contrário das técnicas tradicionais

de recuperação de informação na *Web*, onde as respostas são páginas HTML publicadas na *Web*, soluções baseadas em *Entity Ranking* trazem como respostas as entidades publicadas na *Wikipedia*.

Neste artigo, será abordado um problema relacionado ao de *Entity Ranking*, onde o objetivo envolve descobrir entidades que sejam relevantes a uma entidade de consulta, e não a um conjunto de palavras chave. Essa variação do problema original remete a situações em que o usuário já conhece um tópico, e deseja conhecer outros tópicos que estejam relacionados.

Para lidar com essa questão, o artigo propõe uma abordagem que explora a existência de relacionamentos entre as entidades para o cálculo da relevância. O cálculo leva em consideração a intimidade entre entidades relacionadas e a popularidade delas perante todas as demais. Além disso, é apresentado um mecanismo de extração de relacionamentos baseado na *Wikipedia*, onde a estrutura de categorias é usada para identificar os relacionamentos entre as entidades. O artigo também relata como os relacionamentos podem ser descritos em um grafo de entidades, e apresenta uma técnica de compressão que pode ser usada para reduzir o custo de espaço para representação do grafo.

Este artigo está organizado da seguinte forma: A Seção 2 apresenta a abordagem de *ranking* proposta, que compreende a noção de intimidade e popularidade das entidades. A Seção 3 descreve o mecanismo usado para extrair tanto entidades como relacionamentos a partir da *Wikipedia*. Na Seção 4 são descritos experimentos realizados sobre a coleção INEX, que contempla todas as páginas da *Wikipedia* americana de 2007. Os experimentos demonstram como a abordagem proposta ordena os resultados para algumas entidades de consulta, e apresentam a taxa de compressão que foi obtida sobre o grafo de entidades. Os trabalhos relacionados são descritos na Seção 5.

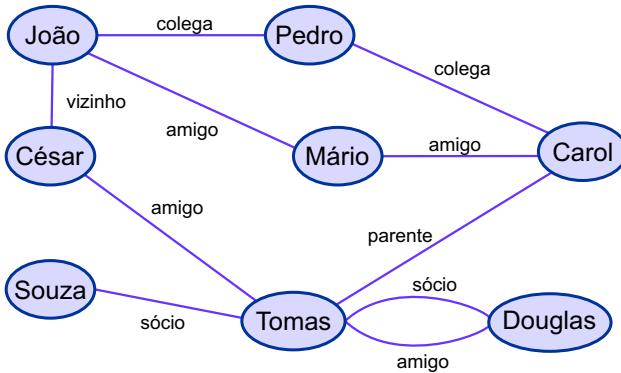
## 2. *Entity Ranking* Baseado em Relacionamentos

O cálculo do *ranking* proposto neste artigo considera a existência de relacionamentos entre as entidades. A partir da análise desses relacionamentos, pretende-se determinar quais dessas entidades são mais relevantes, tanto em relação a entidade especificada na consulta quanto em relação ao conjunto total de entidades mapeadas.

Os relacionamentos entre as entidades podem ser representados por uma estrutura de grafo, onde as entidades são vértices e os relacionamentos entre as entidades são as arestas, conforme a Definição 1.

**Definição 1 (Grafo de Entidades)** *Considere que  $G = \{V, A, T\}$  seja um grafo de entidades orientado e valorado, onde o conjunto de vértices  $V$  é composto pelas entidades  $e$ , enquanto o conjunto de arestas  $A$  é composto por relacionamentos entre essas entidades. Ainda, considere que  $r = (e_i, e_j, tipo) | r \in A$  seja um relacionamento, onde  $e_i \in V$  é uma entidade de origem do relacionamento,  $e_j \in V$  é uma entidade de destino do relacionamento, e  $tipo \in T$  caracteriza o tipo de relacionamento.*

A Figura 1 ilustra um grafo que satisfaçõa essa restrição. O universo de dados capturado no modelo contempla oito entidades, cinco tipos de relacionamento e dez relacionamentos. Como as arestas não são orientadas, o modelo é capaz de representar apenas relacionamentos simétricos. Outros tipos de relacionamentos, apesar de possíveis, não são considerados nesse trabalho.



**Figura 1. Grafo de Exemplo Capturando Relacionamentos entre Oito Entidades**

Dada uma consulta por entidade, o retorno será a lista de entidades relevantes ao objeto de consulta. Neste trabalho, essa lista é determinada conforme a Definição 2.

**Definição 2 (Lista de Resposta)** Considere que  $G = \{V, A, T\}$  seja um grafo de entidades orientado e valorado, e  $e_c \in V$  seja um objeto de consulta. Nesse caso, a lista de resposta será composta por entidades  $e_r \in V$ , onde  $\exists r \in A | (r.e_o = e_c \wedge r.e_d = e_r) \vee (r.e_o = e_r \wedge r.e_d = e_o)$ .

A definição acima parte da intuição de que uma entidade é relevante a uma outra entidade se existirem relacionamentos entre elas. Por exemplo, na Figura 1, dada uma consulta pela entidade "Tomas", a lista de resposta seria composta por quatro entidades: "César", "Souza", "Carol" e "Douglas".

O próximo passo envolve determinar a ordem de exibição das entidades de resposta, de modo que as mais relevantes apareçam primeiro. Neste artigo, considera-se que as entidades mais relevantes a um objeto de consulta sejam aquelas que compartilhem mais relacionamentos com o objeto de consulta. Essa regra procura medir o nível de afinidade entre duas entidades. Quanto mais relacionamentos em comum, maior é a afinidade.

De acordo com esse critério, "Douglas" seria a entidade mais próxima a "Tomas", uma vez que elas compartilham mais relacionamentos (de amizade e sociedade). As demais possibilidade de resposta teriam menos afinidade com "Tomas", já que compartilham com o objeto de consulta apenas um relacionamento cada ("César"/amigo, "Souza"/sociedade e "Carol"/família).

Caso a consulta fosse sobre "João", haveria três entidades de resposta. No entanto, o nível de afinidade com o objeto de consulta é o mesmo. Nesse caso, o critério de desempate passa a ser a popularidade das entidades. O cálculo de popularidade adotado neste artigo é discutido na próxima seção.

## 2.1. Cálculo de Popularidade

O exemplo clássico de *ranking* por popularidade é conhecido como *PageRank*, onde a relevância de páginas Web é usada para ordenar os resultados de buscas por palavra-chave [Brin and Page 1998]. O algoritmo do *PageRank* trata a Web como um grafo, onde as páginas são os vértices e os *links* entre páginas são arestas orientadas. O *ranking* é determinado pela popularidade das páginas, que é calculada de acordo com o número de

*links* que apontam para elas. Além disso, a popularidade de uma página é proporcional a popularidade das páginas que apontam para ela. O cálculo é baseado na probabilidade de que uma página seja acessada aleatoriamente, dado o conjunto total de páginas e seus respectivos *links*. O cálculo aproximado da probabilidade é realizado através de simulações de navegação, onde um usuário fictício percorre as páginas existentes, seguindo os *links* internos aleatoriamente.

A intuição da popularidade é válida quando aplicada ao *ranking* de páginas, mas perde um pouco do sentido se analisada no contexto das entidades conforme modelado neste artigo. Afinal, no grafo *Web* de páginas, os *links* são orientados, e representam "indicações" de que uma página possa ser útil. Naturalmente, quanto mais indicações, mais útil a página deve ser. Já no modelo de entidades proposto, os relacionamentos apresentam associações simétricas, onde a importância das entidades participantes não pode ser representada.

Porém, mesmo partindo de um modelo de grafo semanticamente diferente, um conceito geral de popularidade pode ser explorado para atribuição da importância das entidades. Sendo assim, considera-se que a popularidade seja relativa ao número de arestas das entidades. Ou seja, entidades que possuam mais relacionamentos são mais populares. Além disso, a popularidade de uma entidade pode ser propagada para as entidades relacionadas. Ou seja, entidades que se relacionam com entidades populares serão proporcionalmente mais populares.

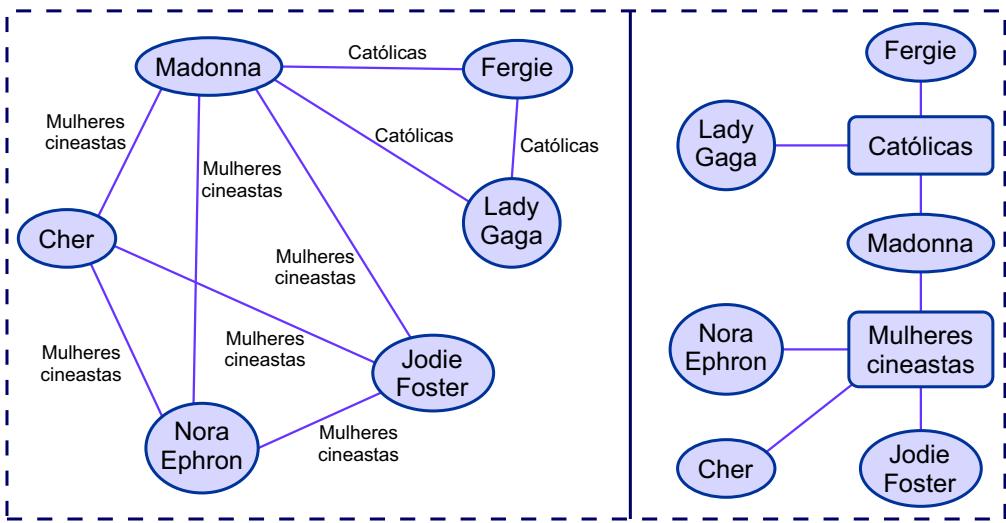
Para ilustrar, considere novamente a Figura 1. Com base nos relacionamentos indicados, "Tomas" seria a entidade mais popular, dado o número de arestas conectadas a ela. As entidades "César", "Mário" e "Pedro" aparecerem empatadas, com dois relacionamentos cada. No entanto, "César" seria considerada mais popular, visto que ela é a única das três que se relaciona com a entidade mais popular.

Com base nessas considerações, propõe-se um algoritmo de propagação de popularidade baseado em um número fixo de iterações. Em cada iteração, a popularidade de uma entidade é calculada como a soma de sua própria popularidade e a popularidade das entidades com quem ela se relaciona. Essa soma leva em consideração as popularidades atribuídas na iteração anterior. É importante destacar que antes da primeira iteração é atribuída uma popularidade igual a todas entidades, para que nenhuma seja favorecida artificialmente. Quanto maior o número de iterações usado, mais distante a popularidade de uma entidade será propagada. Além disso, a popularidade de uma entidade é propagada a um vizinho um número de vezes igual ao número de relacionamentos conectando as entidades. Essa medida visa reforçar o fato de que a propagação deva ser mais acentuada se duas entidades estão mais fortemente relacionadas.

### 3. Extração de Entidades e Relacionamentos

Nesse artigo adota-se uma abordagem simples, tanto para identificação das entidades quanto dos relacionamentos, explorando a estrutura de categorias da *Wikipedia*. De acordo com o *template* de formatação da *Wikipedia*, o rodapé dos artigos possui uma listagem das categorias que estão relacionadas ao tema do artigo em questão. Ao clicar em uma categoria, o usuário é redirecionado a uma página de categoria, que exibe todos os artigos que pertençam a essa categoria.

Como cada artigo da *Wikipedia* versa a respeito de um assunto em particular, pode-



**Figura 2. Grafo de Relacionamentos Modelando o Universo de Dados de Exemplo**

se concluir que cada artigo é uma entidade por si só. Já em uma página de categorias, é possível considerar que os artigos (entidades) ali mencionados possuam relacionamentos entre si, onde o nome do relacionamento vem a ser o nome da categoria. A Figura 2 à esquerda apresenta um grafo gerado a partir da *Wikipedia* de acordo com a abordagem descrita. Para fins de ilustração, apenas uma parte do universo de dados está sendo representada.

Um ponto que vale a pena destacar a partir desse grafo é a presença de cliques, ou seja, um subconjunto de vértices onde cada dois vértices do subconjunto são conectados por uma aresta. Na verdade, verifica-se de um tipo de clique mais específico, onde as arestas possuem o mesmo rótulo. Cada clique neste formato corresponde aos relacionamentos extraídos a partir de uma página de categoria.

Neste artigo, exploramos a existência desse tipo de estrutura para aplicar uma técnica que visa comprimir o grafo. Em suma, a técnica proposta tem por objetivo eliminar os cliques existentes. Dado um clique, o primeiro passo da técnica envolve criar um vértice especial, chamado de vértice de tipo. Em seguida, as arestas que pertenciam ao clique são removidas. Por último, são criadas arestas conectando os vértices que participavam do clique ao novo vértice de tipo.

Pode-se ver que essa técnica aumenta o número de vértices do grafo e reduz o número de arestas. Para compreender esse comportamento, considere o exemplo simples em que o universo de dados compreende  $n$  entidades que partilham uma categoria. Para representar esse fato, o grafo original demandaria  $n$  vértices e um número de arestas equivalente ao intervalo  $n * (n - 1)/2$ . Já o grafo modificado demandaria  $n + 1$  vértices (um vértice extra para representar o tipo) e  $n$  arestas. A Figura 2 à direita serve como constatação dessa afirmação. Nesse exemplo, o grafo adaptado possui dois vértices a mais e duas arestas a menos do que o grafo original.

Em ambos exemplos citados acima, a suposição é que existam poucas categorias compartilhadas pelas entidades. Em um cenário mais realista, como no domínio de sites da *Wikipedia*, existem diversas ilhas de entidades que partilham categorias em comum.

Além do mais, o número de entidades que compartilham uma mesma categoria é maior. A união desses dois fatores faz a redução do número de arestas compensar o aumento do número de vértices. A Seção 4 apresenta experimentos que demonstram a economia de espaço que essa técnica de compressão pode representar.

## 4. Análise Experimental

Os experimentos relatados neste artigo tem propósito duplo. Em primeiro lugar, será feita uma análise do custo em espaço comparando o grafo de relacionamentos puro com o grafo de relacionamentos comprimido. Além disso, serão analisados os resultados da ordenação usando o algoritmo de *Entity Ranking* proposto.

Para ambas as análises é utilizada a coleção INEX 2007<sup>1</sup>, que contem todas as páginas da *Wikipedia* americana no ano de 2007. A coleção é usada como *benchmark* para a realização de experimentos de *Entity Extraction*. Ao todo, 659388 verbetes e 115625 categorias são indexados. Além disso, são disponibilizadas consultas prontas juntamente com a resposta esperada. Como as consultas são compostas por lista de palavras chave e categorias, não se pode aproveitá-las nos experimentos.

Os resultados atingidos são apresentados a seguir.

### 4.1. Análise do Consumo de Memória

Todos verbetes, categorias e relacionamentos entre verbetes foram extraídos a partir do INEX e alimentados em uma base de dados MySQL. Para fins de comparação, duas estruturas de banco foram utilizadas, sendo que uma modela o grafo de relacionamentos puro enquanto a outra modela o grafo comprimido.

O modelo não comprimido ocupa cerca de 10,13 GB de espaço físico, enquanto o modelo comprimido ocupa cerca de 175,48 MB, o que caracteriza uma melhora de 83,08% na economia de espaço em disco. A justificativa para tamanha diferença pode ser compreendida ao analisar-se os dados da coleção. Mais de 26785 categorias são associadas com ao menos 10 verbetes. Além disso, algumas categorias estão associadas a um número de verbetes bastante elevado. Por exemplo, uma categoria chega a possuir associações com 4534 verbetes. Quanto maiores forem os cliques no grafo, maior o ganho em se optar pela versão compacta. Curiosamente, 40024 categorias não estão associadas a nenhum verbete.

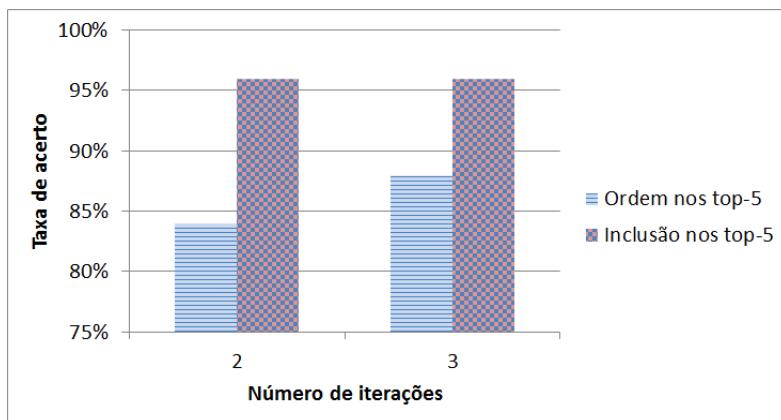
### 4.2. Análise do Algoritmo de Ordenamento

Com o intuito de verificar o funcionamento do algoritmo de ordenamento de entidades, foram criados cinco objetos de consulta, compostos pelas seguintes entidades: "Superman", "IBM", "Madonna", "Friends" e "Counter-Strike".

Para cada objeto de consulta, foram recuperadas as entidades de resposta nas posições 1, 2, 3, 4, 5, 31, 51, 81, 131, 211, nesta mesma ordem. Os cinco primeiros resultados caracterizam entidades bastante relacionadas ao objeto de consulta, sendo que o grau de relacionamento a princípio não difere muito entre eles. Os cinco últimos resultados caracterizam entidades pouco relacionadas ao objeto de consulta, sendo que o grau de relacionamento a princípio cai bastante conforme se usa entidades de ordem inferior.

---

<sup>1</sup><http://www-connex.lip6.fr/denoyer/wikipediaXML/>



**Figura 3. Resultados da Propagação de Popularidade**

Um dos objetivos do experimento é medir se usuários concordam com a forma com que as entidades relacionadas foram ordenadas. Para realizar essa medição, doze usuários responderam a um questionário online<sup>2</sup>. Para cada objeto de consulta, o questionário pedia ao usuário que ordenasse as dez entidades relacionadas de acordo com a relevância. Para que a avaliação não fosse tendenciosa, as entidades foram dispostas em ordem alfabética.

Foram criados dois critérios de avaliação dos resultados, cuja definição e resultados obtidos são apresentados a seguir:

**Inclusão nos top-5** Verifica quantas das cinco entidades melhor ordenadas pelos usuários estão contidas nos top-5. Após realizar a média das respostas de todos os usuários, chegou-se a uma taxa de 63%. Esse resultado é um indicativo de que a estratégia de ordenação adotada é capaz de separar entidades bastante relacionadas das pouco relacionadas, para a maioria dos casos.

**Ordem nos top-5** Verifica quantas das cinco entidades melhor ordenadas pelos usuários estão na mesma ordem no top-5. Após realizar a média das respostas de todos os usuários, chegou-se a uma taxa de 17%. Esse resultado pouco expressivo pode ser interpretado pela dificuldade natural de determinar a relevância em alguns casos. Por exemplo, para o objeto de consulta "Superman", "Flash" e "Batman" são possibilidade de resposta. No entanto, os usuários tiveram dificuldade de determinar qual dessas possibilidades deveria ser melhor ordenada.

Outro experimento realizado analisou o desempenho do algoritmo de ordenação quando a popularidade é propagada a mais níveis de iteração. A Figura 3 apresenta os resultados obtidos, comparando a ordem alcançada usando um nível de propagação com a ordem alcançada usando dois e três níveis de propagação.

Conforme apresentado na Figura 3, a ordem e a inclusão não mudam drasticamente em comparação com a ordem alcançada com um nível de propagação. Isso ocorre principalmente porque o método de ordenação leva em consideração a afinidade em primeiro lugar, e em segundo a popularidade. Ou seja, se o número de relacionamentos for igual, a ordem dos resultados não muda quando se alterar o número de iterações de

<sup>2</sup><http://fluidsurveys.com/surveys/thiago-krug/nivel-de-relacionamento-entre-entidades/>

propagação de popularidade.

Também é possível constatar que existem poucas diferenças usando dois ou três níveis de propagação, o que também pode ser explicado pela forma como o escore de ordenamento é calculado. No entanto, convém destacar que algumas mudanças na ordem interessantes ocorreram quando passou a se usar mais níveis de propagação. É citado como exemplo o objeto de consulta "Superman". Usando um nível de propagação, a entidade relacionada "Kon-El" aparece na 14<sup>a</sup> posição. Seria desejável que essa entidade obtivesse uma relevância maior, visto que se trata de um personagem que pertence a dinastia do "Superman", o que ocorre quando se usa mais níveis de propagação.

## 5. Trabalhos Relacionados

O problema de *Entity Ranking* se assemelha bastante com a proposta deste artigo. Enquanto o *Entity Ranking* tradicional trata de encontrar entidades relacionadas a uma consulta por palavras-chave (e um conjunto opcional de entidades e categorias), a proposta de trabalho do artigo já parte de uma entidade existente, e pretende descobrir as entidades relevantes. Em seguida são discutidos alguns dos trabalhos de *Entity Ranking* existentes.

O trabalho de [Zhu et al. 2007] recupera primeiro todas as entidades que aparecem em páginas onde as palavras chave da consulta também aparecem. A relevância da entidade é computada com base na proximidade entre essa entidade e as palavras chave. Em seguida, as entidades recuperadas são filtradas caso nenhuma de suas categorias coincida com as categorias usadas na consulta.

Em [Kaptein and Kamps 2013], a busca é realizada através de uma série de scores, sendo que um deles compara as categorias da consulta e as categorias pertencentes a cada entidade indexada. A comparação é baseada na distância entre as categorias, calculada através da métrica KL-Divergence. Para o cálculo da divergência, são considerados os termos de todas as entidades que pertencem às categorias comparadas. A relevância de uma entidade é computada em função da menor distância entre qualquer das categorias pertencentes à pesquisa e qualquer das categorias pertencentes à entidade.

Já em [Vercoustre et al. 2008], a distância entre os dois conjuntos de categorias é calculada através de uma função  $\frac{cat(Q) \cap cat(E)}{cat(E)}$  que mede a razão das categorias da consulta e da entidade que intersectam com relação ao total de categorias da entidade. Essa função compartilha da intuição que é usada neste artigo, a de que o compartilhamento de categorias indica relevância entre entidades. No entanto, o artigo dá mais importância ao número de compartilhamentos do que a proporção relativa deles. Além disso, tanto o trabalho de [Vercoustre et al. 2008] como os demais citados nesta seção possuem métricas adicionais, que utilizam as palavras chave da consulta para o cálculo, enquanto a proposta do artigo é completamente baseada no conceito de compartilhamento de categorias.

Como o foco é voltado ao uso de categorias, a possibilidade de compactar grafos que possuam essa característica se torna relevante. Dentro desse contexto, grafos baseados na *Web* possuem características próprias que podem melhorar as taxas de compressão. Em geral, esse tipo de grafos representa relações entre páginas HTML derivadas a partir de *hyperlinks*. Com base em estudos, descobriu-se que a criação desses relacionamentos obedece a alguns padrões. Por exemplo, páginas dentro de um domínio costumam citar a si próprias. Isso permite com que as páginas sejam numeradas de acordo com a ordem

lexicográfica de URL, para que páginas de um mesmo domínio possuam identificadores próximos [Bharat et al. 1998, Blandford et al. 2003].

Outro exemplo considera que muitas páginas incluem *links* para o mesmo conjunto de páginas. A partir dessa observação, [Kumar et al. 1999] propôs uma variação de grafos bipartidos para representar o conjunto de páginas que citam e o conjunto de páginas citadas. Ocorrências desses grafos alimentam uma base de conhecimento, que pode ser usada para acelerar buscas e para processos de mineração de dados.

Seguindo a mesma linha, [Adler and Mitzenmacher 2001] propôs técnicas de similaridade que encontram páginas com listas de adjacência semelhantes. Após descobertas, a lista de adjacência de uma página é substituída por uma referência à página similar, juntamente com operações de edição para representar os pontos em que as duas listas originais divergiam.

Em [Buehrer and Chellapilla 2008], os conjuntos de páginas largamente citadas geram vértices estrela. As páginas que citam esse conjunto de páginas passam a citar esse vértice agrupador, o que reduz o número de arestas do grafo. Esse tipo de compressão é semelhante ao que nós empregamos nesse artigo, no sentido de que vértices especiais são gerados.

Uma distinção importante entre os trabalhos de compressão citados e o apresentado neste artigo é o fato de que as propriedades que possibilitam uma compressão precisam ser descobertas, como os grafos bipartidos ou listas de adjacência semelhantes. Já no contexto explorado no artigo, essas propriedades podem ser automaticamente derivadas durante a extração dos relacionamentos.

## 6. Conclusões

Este artigo explora uma forma diferente de *Entity Ranking*, em que o objeto de consulta é uma entidade, e o objetivo é encontrar a lista de entidades que estão relacionadas à entidade pesquisada. Os resultados obtidos mostram que os conceitos de afinidade e popularidade das entidades são parâmetros úteis na ordenação. O primeiro deles consegue realizar a divisão entre entidades bastante próximas ao objeto de consulta e entidades que são menos próximas. O segundo mostrou-se particularmente interessante em alguns casos específicos, especialmente quando a popularidade era propagada por mais de dois níveis no grafo de relacionamentos. Nesses casos, entidades reconhecidamente mais conhecidas ultrapassaram entidades menos conhecidas.

No entanto, o impacto da popularidade no resultado é reduzido, uma vez que o principal critério de ordenação é o nível de afinidade. Dessa forma, pretende-se estudar outras formas de usar esses dois valores para chegar a um escore final. Além disso, outro problema a ser estudado envolve a extração de relacionamentos entre entidades a partir de texto em linguagem natural. A presença de dados extraídos de outros tipos de fontes de dados pode inclusive levar a criação de um novo critério de ordenação, baseado na importância inferida ao método de extração que gerou o relacionamento. Essa regra partiria da intuição de que duas entidades que apareçam dentro de uma mesma sentença tenham um grau de afinidade diferente do que entidades que partilhem de uma mesma categoria na *Wikipedia*.

Para finalizar, é enfatizado o fato de que a estrutura dos relacionamentos na *Wi-*

*kipedia* permitiu que se alcançasse altos graus de compressão dos dados sem perda de informação. O próximo passo é analisar se a técnica de compressão usada segue sendo útil quando novos critérios de ordenação forem usados. Caso a ordenação empregue características específicas no relacionamento entre entidades (como o nível de afinidade), a técnica de compressão precisaria ser modificada para evitar que essas características específicas sejam perdidas.

## Referências

- Adler, M. and Mitzenmacher, M. (2001). Towards compressing web graphs. In *DCC*, pages 203–212. IEEE Computer Society.
- Ahn, D., Jijkoun, V., Mishne, G., Müller, K., de Rijke, M., and Schlobach., S. (2004). Using wikipedia at the trec qa track. In *Proceedings of TREC 2004*.
- Bharat, K., Broder, A., Henzinger, M. R., Kumar, P., and Venkatasubramanian, S. (1998). The connectivity server: Fast access to linkage information on the Web. In *Proceedings of the 7th International World Wide Web Conference (WWW-7)*, pages 469–477, Brisbane, Australia.
- Blandford, Blelloch, and Kash (2003). Compact representations of separable graphs. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*.
- Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands. Elsevier Science Publishers B. V.
- Buehrer, G. and Chellapilla, K. (2008). A scalable pattern mining approach to web graph compression with communities. In Najork, M., Broder, A. Z., and Chakrabarti, S., editors, *WSDM*, pages 95–106. ACM.
- Kaptein, R. and Kamps, J. (2013). Exploiting the category structure of wikipedia for entity ranking. *Artif. Intell.*, 194:111–129.
- Kumar, S. R., Raghavan, P., Rajagopalan, S., and Tomkins, A. (1999). Extracting large-scale knowledge bases from the web. In *Proceedings of the 25th VLDB Conference*.
- Mihalcea, R. (2007). Using wikipedia for automatic word sense disambiguation. In *Proceedings of NAACL HLT*, volume 2007, pages 196–203.
- Paşca, M. (2007). Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM ’07, pages 683–690, New York, NY, USA. ACM.
- Vercoustre, A.-M., Pehcevski, J., and Thom, J. A. (2008). Focused access to xml documents. chapter Using Wikipedia Categories and Links in Entity Ranking, pages 321–335. Springer-Verlag, Berlin, Heidelberg.
- Zhu, J., Song, D., and Rüger, S. M. (2007). Integrating document features for entity ranking. In Fuhr, N., Kamps, J., Lalmas, M., and Trotman, A., editors, *INEX*, volume 4862 of *Lecture Notes in Computer Science*, pages 336–347. Springer.

# Extração de Nomes de Pessoas Baseado em Etapas de Etiquetamento

Henrico B. Brum<sup>1</sup>, Sergio L. S. Mergen<sup>1</sup>

<sup>1</sup>Campus Alegrete - Universidade Federal do Pampa (UNIPAMPA)  
CEP – 97.546-550 – Alegrete – RS – Brasil

henrico.brum@gmail.com, sergiomergen@unipampa.edu.br

**Abstract.** *The identification of person names inside natural language texts can be used for many goals, such as applications related to the information retrieval and data analysis areas. Given this, our paper explores the entity extraction problem by using unsupervised strategies focused on the recognition of person names. The proposed approach employs a classification process divided in three stages, called Location, Filtering and Expansion. Each stage executes tagging rules that satisfies the purpose of the stage. The experiments show the performance of the rules for the classification of textual documents, analyzing the rules individually and collectively.*

**Resumo.** *A identificação de nomes de pessoas em meio a textos em linguagem natural pode ser usada para diversas finalidades, como por aplicações na área de recuperação de informação e análise de dados. Dessa forma, esse artigo explora a extração de entidades nomeadas por meio de estratégias não supervisionadas com foco no reconhecimento de nomes de pessoas. A abordagem proposta emprega um processo de classificação dividido em três etapas, chamadas de Localização, Filtragem e Expansão. Para cada uma dessas etapas são executadas regras de etiquetamento que atendem os requisitos de cada etapa. Os experimentos demonstram o desempenho das regras utilizadas na classificação de documentos textuais, analisando-as de forma individual e coletiva.*

## 1. Introdução

Um dos temas de pesquisa que tem recebido bastante atenção atualmente envolve entidades nomeadas (ENs). De acordo com [Krishnan and Manning 2006], entidades nomeadas são termos que caracterizam algum objeto, fornecendo indícios que ajudam a identificá-lo de forma inequívoca. Os exemplos mais comuns de entidades nomeadas incluem nomes de pessoas, nomes de organizações, localidades, entre outros.

Diversas aplicações podem fazer uso de entidades nomeadas, principalmente na área de recuperação de informação e análise de dados. Por exemplo, ao reconhecer que uma entidade nomeada é utilizada em uma consulta, os motores de busca podem se valer dessa informação para aprimorar o resultado da consulta. Outro exemplo envolve a descoberta de conhecimento sobre informações textuais. Algoritmos de processamento de linguagem natural podem se valer das entidades nomeadas contidas em um texto para realizar associações entre os componentes desse texto. Mais adiante, técnicas de aprendizado de máquina poderiam utilizar as associações descobertas em textos históricos para prever comportamentos futuros.

Para que tais aplicações sejam possíveis, é necessário identificar os termos que representem entidades nomeadas. No decorrer dos anos, diversas técnicas de reconhecimento de entidades nomeadas foram propostas. Muitas delas exploram evidências presentes nas sentenças em que os termos aparecem para guiar o reconhecimento. Algumas dessas evidências compreendem a detecção de padrões e a análise da vizinhança (termos que costumam cercar entidades nomeadas) usada em conjunto com a análise da estrutura morfossintática da sentença.

O uso de muitas evidências torna difícil estipular uma equação precisa para identificar se um conjunto de termos equivale a uma entidade nomeada. Por isso, costuma-se recorrer a algoritmos supervisionados de aprendizado de máquina, capazes de atribuir pesos para as palavras de modo a melhorar o cálculo que realiza a classificação. Contudo, tais algoritmos requerem bases de treinamento, contendo rótulos pré-marcados identificando entidades nomeadas, para que novas entidades possam ser reconhecidas no futuro.

Nesse artigo, será analisada a possibilidade de que o reconhecimento de ENs possa ser feito através de uma abordagem que não requer bases de treinamento. Para isso, será apresentado um mecanismo de classificação composto por três etapas: Localização, Filtragem e Expansão. Cada etapa aceita regras com um comportamento específico, cujo objetivo é marcar ou desmarcar palavras como nomes de pessoas. Também são propostas regras que podem ser utilizadas dentro de cada etapa.

Para verificar a qualidade do processo de classificação e das regras proposta, são realizados experimentos utilizando o *benchmark HAREM*, um repositório *XML* de textos demarcados de acordo com o seu tipo. Os experimentos demonstram a importância de cada uma das etapas assim como das regras das quais são compostas.

Este artigo está estruturado da seguinte forma: Na seção 2 é apresentado o mecanismo de classificação proposto, as etapas envolvidas e as regras que foram desenvolvidas. Na seção 3 são explicados os experimentos realizados sobre o repositório *HAREM*. Os trabalhos relacionados são descritos na seção 4. Para finalizar, a seção seção 5 traz as conclusões.

## 2. Regras Propostas

O objetivo geral dos algoritmos de extração de ENs é percorrer blocos de texto e inserir marcações sintáticas nas palavras (etiquetar, ou *tag*) de acordo com o seu tipo. Essa demarcação também ocorre em técnicas do tipo *POS Tagger* (*Part of speech tagger*), onde o objetivo é etiquetar as palavras de acordo com a sua classe gramatical. Por esse motivo, algoritmos desse tipo são chamados de etiquetadores.

Como o foco deste trabalho comprehende o reconhecimento de nomes de pessoas, consideramos apenas dois tipos de etiqueta: 'N' (nome) e 'O' (outro). Dessa forma, dada uma sentença qualquer, o objetivo do etiquetamento é rotular as palavras como 'N' ou 'O', conforme ilustrado na Figura 1. Observa-se que a etiqueta 'O' caracteriza qualquer coisa que não seja nome de pessoa, como numeral, título, verbo, advérbio ou até mesmo de uma entidades nomeada que não represente um indivíduo.

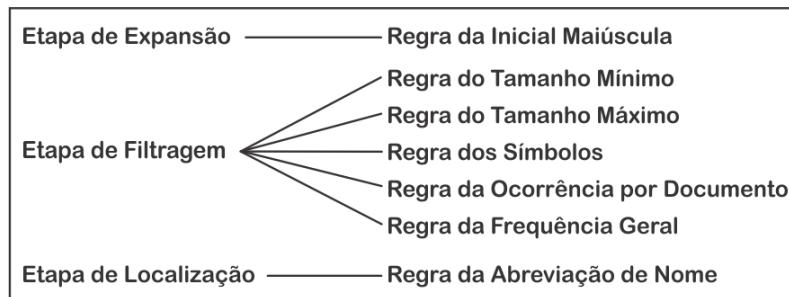
Para atingir esse objetivo é proposto um etiquetador baseado em etapas de execução, conforme apresentado na Figura 2. O etiquetamento é dividido em três etapas: Localização, Filtragem e Expansão. Cada etapa é composta por um conjunto de

**Frase Exemplo:** O carteiro Ricardo B. Souza dirigiu até o bairro Rosário para encontrar seus amigos Marcio Matheus e Soaraia Falcão.

O	carteiro	Ricardo	B.	Souza	dirigiu	até	o	bairro	Rosário	para
0	0	N	N	N	0	0	0	0	0	0
encontrar	seus	amigos	Marcio	Matheus	e	Soaraia	Falcão.			
0	0	0	N	N	0	N	N			

**Figura 1. Exemplo de marcação de palavras em uma sentença**

regras que atendem ao propósito da etapa. De modo geral, a Etapa de Localização tem o propósito de encontrar candidatos a nomes de pessoas, a Etapa de Filtragem tem o propósito de eliminar candidatos erroneamente identificados, e a etapa de expansão tem o objetivo de encontrar novos candidatos a partir dos que sobreviveram à etapa de filtragem.



**Figura 2. Categorias de Regras Propostas**

Para demonstrar as características e problemas enfrentados na implementação das regras de cada uma das etapas, utilizaremos um exemplo (Figura 3) composto por três parágrafos retirados da internet<sup>1</sup>. Os termos sublinhados nos textos referem-se a entidades nomeadas que deveriam ser detectadas como nomes de pessoas.

## 2.1. Etapa de Localização

A etapa de Localização tem o objetivo de encontrar o máximo possível de nomes de pessoas no texto. Para isso, devem ser projetadas regras de localização que tenham uma alta cobertura, mesmo que sejam associadas a uma precisão baixa. Nesse momento não existe a preocupação com excesso de informação irrelevante ou inválida, uma vez que a etapa de filtragem se encarregará de eliminar os falsos positivos encontrados.

Para fins de etiquetamento, considera-se que inicialmente todas palavras são marcadas com o rótulo 'O'. A medida que as regras descubram candidatos, o rótulo das palavras vai sendo adaptado.

Neste artigo, consideramos uma única regra de localização, baseada na observação de que nomes de pessoas, em geral, são escritos com a primeira letra maiúscula. A definição da regra encontra-se abaixo:

<sup>1</sup>Parágrafos retirados dos artigos referentes a Hebe Camargo, Quentin Tarantino e Jodie Foster presentes no domínio <http://pt.wikipedia.org.br>

<p>Sua família mudou-se para a capital, São Paulo em 1943, quando Hebe tinha 14 anos de idade. Fêgo já na capital passou integrar a Orquestra da Rádio Difusora, onde ele regeu a orquestra da emissora de rádio e sempre levava consigo Hebe Camargo. Ela iniciou como cantora na rádio Tupi aos 15 anos de idade se apresentando no programa Clube Papai Noel. Ao gravar um CD em homenagem a Carmen Miranda ela ficou conhecida como “estrelinha do samba” e posteriormente como “A estrela de São Paulo”. Em 1950 ela lançou sua primeira música cantada, “Oh! José” juntamente com “Quem Foi que Disse” em um compacto de 78 rotações.</p>	<p><u>Tarantino</u> tem um grupo de atores que freqüentemente participam de seus filmes, incluindo <u>Tim Roth</u> (<u>Reservoir Dogs</u>, <u>Pulp Fiction</u>, <u>Four Rooms</u>), <u>Harvey Keitel</u> (<u>Reservoir Dogs</u>, <u>Pulp Fiction</u>, <u>From Dusk Till Dawn</u>), <u>Uma Thurman</u> (<u>Pulp Fiction</u>, <u>Kill Bill: Vol.1</u>, <u>Kill Bill: Vol.2</u>), <u>Michael Madsen</u> (<u>Reservoir Dogs</u>, <u>Kill Bill: Vol.1</u>, <u>Kill Bill: Vol.2</u>, <u>Sin City</u>), <u>Steve Buscemi</u> (<u>Reservoir Dogs</u>, <u>Pulp Fiction</u>, <u>Four Rooms</u>, <u>Sin City</u>, <u>Grindhouse</u>) e <u>Samuel L. Jackson</u> (<u>Pulp Fiction</u>, <u>Jackie Brown</u>, <u>Kill Bill Vol.2</u>).</p>	<p><u>Jodie</u> alcançou fama mundial com o sucesso do filme e com uma indicação ao Oscar de melhor atriz (coadjuvante/secundária). Poucos anos depois, a tentativa de assassinato do presidente dos Estados Unidos <u>Ronald Reagan</u>, baleado por um psicopata chamado <u>John Hinckley</u>, lhe causaria um grave conflito emocional e psicológico, com a revelação feita por <u>Hinckley</u> de que o ato visava chamar a atenção de <u>Jodie</u>, por quem era platonicamente apaixonado e a quem seguia de longe há meses no campus da Universidade de Yale, onde ela estudava, e que havia assistido <u>Taxi Driver</u> por mais de quarenta vezes apenas para vê-la na tela.</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Figura 3. Exemplo de parágrafos de textos distintos**

**Regra da Inicial Maiúscula:** Todas as palavras que possuam a primeira letra em maiúscula recebem a marcação ‘N’ .

Para exemplificar a regra, considere a Figura 4. Como pode ser observado, todos os nomes de pessoas foram corretamente marcados. Porém, muitas outras palavras que não se referiam a pessoas também foram marcadas como tal, como siglas ou substantivos que iniciavam sentenças. Esse comportamento está em consonância com o propósito da etapa de localização, uma vez que a regra conseguiu uma alta taxa de cobertura.

<p>Sua família mudou-se para a capital, São Paulo em 1943, quando Hebe tinha 14 anos de idade. Fêgo já na capital passou integrar a Orquestra da Rádio Difusora, onde ele regeu a orquestra da emissora de rádio e sempre levava consigo Hebe Camargo. Ela iniciou como cantora na rádio Tupi aos 15 anos de idade se apresentando no programa Clube Papai Noel. Ao gravar um CD em homenagem a Carmen Miranda ela ficou conhecida como “estrelinha do samba” e posteriormente como “A estrela de São Paulo”. Em 1950 ela lançou sua primeira música cantada, “Oh! José” juntamente com “Quem Foi que Disse” em um compacto de 78 rotações.</p>	<p><u>Tarantino</u> tem um grupo de atores que freqüentemente participam de seus filmes, incluindo <u>Tim Roth</u> (<u>Reservoir Dogs</u>, <u>Pulp Fiction</u>, <u>Four Rooms</u>), <u>Harvey Keitel</u> (<u>Reservoir Dogs</u>, <u>Pulp Fiction</u>, <u>From Dusk Till Dawn</u>), <u>Uma Thurman</u> (<u>Pulp Fiction</u>, <u>Kill Bill: Vol.1</u>, <u>Kill Bill: Vol.2</u>), <u>Michael Madsen</u> (<u>Reservoir Dogs</u>, <u>Kill Bill: Vol.1</u>, <u>Kill Bill: Vol.2</u>, <u>Sin City</u>), <u>Steve Buscemi</u> (<u>Reservoir Dogs</u>, <u>Pulp Fiction</u>, <u>Four Rooms</u>, <u>Sin City</u>, <u>Grindhouse</u>) e <u>Samuel L. Jackson</u> (<u>Pulp Fiction</u>, <u>Jackie Brown</u>, <u>Kill Bill Vol.2</u>).</p>	<p><u>Jodie</u> alcançou fama mundial com o sucesso do filme e com uma indicação ao Oscar de melhor atriz (coadjuvante/secundária). Poucos anos depois, a tentativa de assassinato do presidente dos Estados Unidos <u>Ronald Reagan</u>, baleado por um psicopata chamado <u>John Hinckley</u>, lhe causaria um grave conflito emocional e psicológico, com a revelação feita por <u>Hinckley</u> de que o ato visava chamar a atenção de <u>Jodie</u>, por quem era platonicamente apaixonado e a quem seguia de longe há meses no campus da Universidade de Yale, onde ela estudava, e que havia assistido <u>Taxi Driver</u> por mais de quarenta vezes apenas para vê-la na tela.</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Figura 4. Marcação de Palavras na Etapa de Localização**

Apesar de considerar-se apenas uma regra na Etapa de Localização, outras regras poderiam ser concebidas, caso elas contenham indícios fortes de que uma palavra corresponde a nome de pessoa, mesmo sem iniciar em maiúscula. A complementação das

regras de Localização pode ser abordada em trabalhos futuros.

## 2.2. Etapa de Filtragem

Apesar da boa cobertura oferecida pela Etapa de Localização, novas regras são necessárias para eliminar as palavras indevidas. As regras que possuem esse intuito são encontradas na Etapa de Filtragem. Nesta etapa, apenas as palavras previamente etiquetadas como 'N' são processadas. Caso qualquer uma das regras de filtragem for satisfeita, a palavra volta a ser marcada como 'O'.

As regras de filtragem adotadas analisam as palavras e as descartam em relação à condições como tamanho mínimo, máximo, presença de símbolos e numerais na palavra e frequência geral da palavra. A seguir, cada uma das regras é apresentada com mais detalhes:

**Regra do Tamanho Mínimo** Palavras com menos de três caracteres são marcadas com 'O'.

Essa regra parte da observação de que nomes de pessoas normalmente possuem mais do que dois caracteres. Para exemplificar, essa regra poderia ser usada para remover palavras erroneamente marcadas como 'N' pela etapa anterior, como artigos (ex. "A"), conjunções (ex. "Em") e algumas siglas (ex. "CD").

**Regra do Tamanho Máximo** Palavras com mais de 10 caracteres são marcadas com 'O'.

Essa regra parte da observação de que nomes de pessoas normalmente possuem menos do que onze caracteres. Para exemplificar, essa regra poderia ser usada para remover palavras erroneamente marcadas como 'N' pela etapa anterior, como nomes de filmes (ex. *Grindhouse*) e títulos de instituições (ex. *Universidade*).

**Regra dos Símbolos** Palavras que possuam em sua composição símbolos e numerais são marcadas com 'O'.

Essa regra parte da observação de que nomes de pessoas não possuem símbolos especiais. A presença desses símbolos está normalmente associada a termos referentes a e-mails e endereços. A implementação dessa regra ignora hifens, acentos e apóstrofos, pois estes podem ser encontrados em nomes de pessoas. Para exemplificar, essa regra poderia ser usada para remover palavras erroneamente marcadas como 'N' pela etapa anterior, como em "*Kill Bill Vol.2*".

**Regra da Frequência por Documento** Palavras que ocorram com frequência maior do que  $\alpha$  em todos os documentos são marcadas como 'O', sendo  $\alpha$  um valor percentual.

Essa regra parte da observação de que o mesmo nome de pessoa não costuma aparecer com frequência elevada no conjunto total de documentos. Sendo assim, as palavras comuns (de frequência elevada) são descartadas, enquanto as palavras raras (de frequência baixa) são conservadas. Para exemplificar, considere a Figura 4, e suponha que  $\alpha$  seja igual a 50%. Nesse caso, a regra removeria a palavra 'Ela' como nome de pessoa, visto que esta palavra tem uma frequência igual a 66,6% (a palavra aparece no primeiro documento e no terceiro documento, sendo que temos 3 documentos no total).

O ponto de corte  $\alpha$  pode ser definido de forma empírica, conforme demonstrado

na seção de experimentos.

### 2.3. Etapa de Expansão

A Etapa de Expansão prevê o uso de marcações prévias para encontrar novos nomes de pessoas. Assim como na Etapa de Localização, as regras nesta etapa são aditivas. Ou seja, elas tem o objetivo de mudar as marcações de 'O' para 'N'.

Neste artigo será considerada uma única regra de expansão, a regra de Abreviação de Nome:

**Regra de Abreviação de Nome:** Uma palavra é marcada como 'N', se possuir um único caractere maiúsculo (ignorando caracteres de pontuação), e a sucessora possuir a etiqueta 'N'.

Essa regra parte da observação de que caracteres isolados sucedidos por um nome de pessoa provavelmente se referem a uma abreviação de um nome composto. Para exemplificar, através da aplicação desta regra, a palavra 'L.' de 'Samuel L. Jackson', que seria marcada como 'O' pela regra de tamanho mínimo executada durante a etapa anterior, voltaria a ser marcada como 'N' durante a expansão.

## 3. Experimentos e Resultados

A qualidade do processo de classificação de nomes de pessoas proposto neste trabalho é avaliada através do *HAREM*, um repositório de textos marcados usado em experimentos no campo de Processamento de Linguagem Natural<sup>2</sup>. O *HAREM* é composto por um documento *XML* com 129 documentos com marcações de entidades nomeadas distribuídas em categorias. Ao todo são 74.298 palavras, sendo 2.622 representando nomes de pessoas. No documento *XML* os nomes de pessoas são marcados com a categoria "PESSOA" e tipo "INDIVIDUAL".

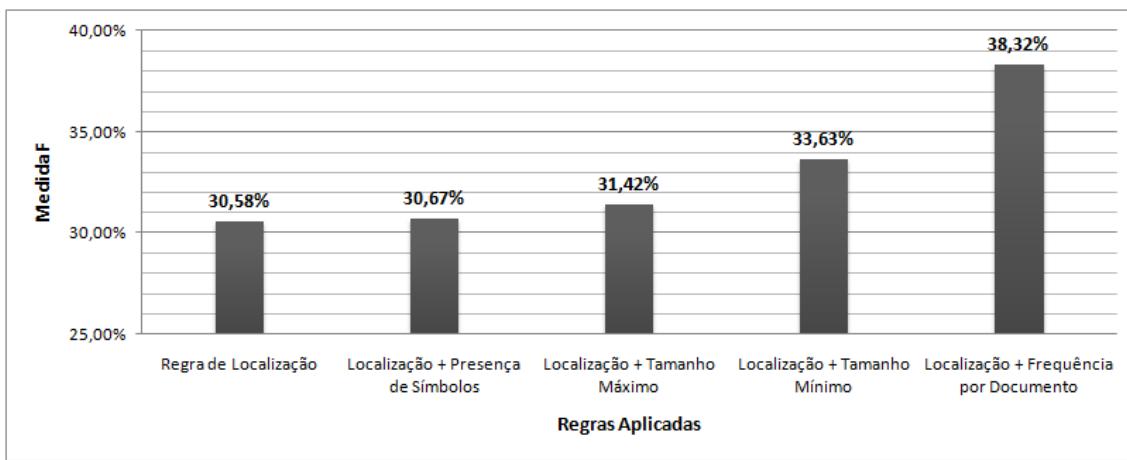
Para fins de avaliação usou-se as métricas de Precisão e Cobertura. A Precisão é dada pelo número de palavras classificadas como nome de pessoas que realmente são nomes de pessoas no *HAREM*. Já a Cobertura é dada pelo número de palavras classificadas como nome de pessoas em função do total de nomes de pessoas existentes no *HAREM*. Também usou-se a medida F, que calcula uma média harmônica entre os valores de precisão e cobertura para chegar a um valor que exprima o impacto conjunto dessas métricas.

Para a Regra da Frequência por Documento, o valor de  $\alpha$  foi determinado com base em uma análise manual de uma amostragem de documentos contidos no *HAREM*. A partir dessa amostragem, procurou-se pelo nome da pessoa (entidade nomeada da categoria "PESSOA" e tipo "INDIVIDUAL") que aparece com maior frequência. O valor de frequência encontrado foi usado como ponto de corte  $\alpha$ .

A Figura 5 apresenta os resultados obtidos com a aplicação das regras de Localização e Filtragem. Em um dos casos avaliados, é avaliada apenas a Etapa de Localização, ou seja, a regra da Inicial Maiúscula. Nos demais casos, a regra da Inicial Maiúscula é aplicada juntamente com alguma técnica da etapa de filtragem.

---

<sup>2</sup>Disponível em '<http://www.linguateca.pt/harem/>'



**Figura 5. Regras de FiltragemAplicadas Individualmente**

Com a regra de Localização isolada foi obtida uma cobertura de 100% na análise dos 129 documentos, o que indica que todos os nomes de pessoas foram corretamente marcados como tal. Porém, a precisão foi de 18%, o que significa que muitas palavras foram erroneamente marcadas como nomes de pessoas. A medida F resultante atingiu 30,58%.

Como a Figura demonstra, as demais regras avaliadas procuram filtrar o resultado obtido pela Etapa de Localização de modo a aumentar a precisão. De modo geral, em todas as regras testadas houve aumento da precisão, tendo como contrapartida uma redução na cobertura. Observa-se que a regra de Frequência por Documento foi a que obteve um melhor desempenho.

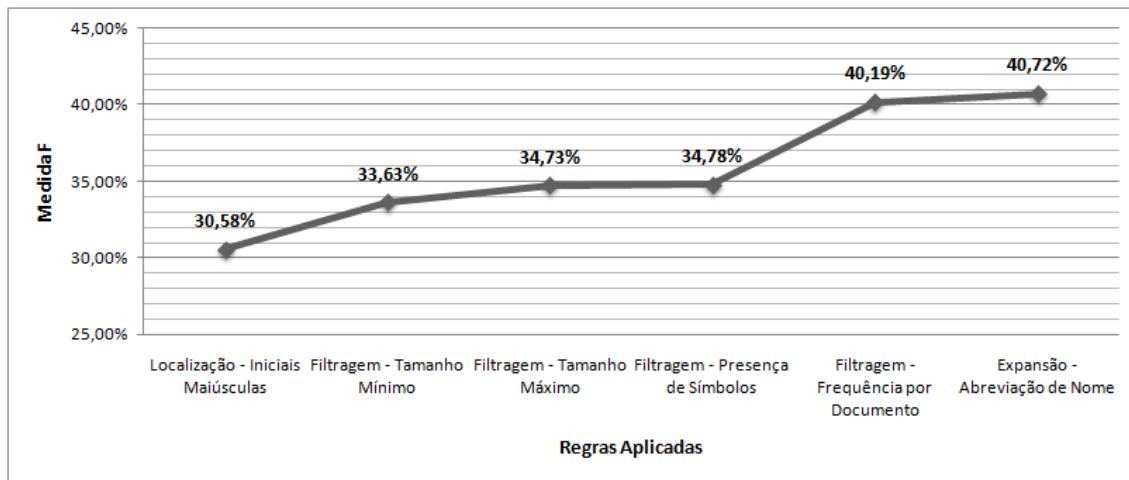
A Tabela 1 exibe alguns exemplos de palavras que perderam a marcação 'N' acertadamente após o uso de regras específicas de Filtragem. Como pode-se ver, as regras são complementares, o que indica que elas possam ser usadas em conjunto para obter uma melhor filtragem.

Regra de Filtragem	Exemplos de Nomes Filtrados
Regra de Tamanho Mínimo	Siglas (como <i>RS</i> e <i>SP</i> ) Exclamações (como <i>Ah</i> ) Títulos (como <i>Sr</i> e <i>Dr</i> )
Regra do Tamanho Máximo	Cidades (cidades (como <i>Teresópolis</i> ) Países (como <i>Grã-Bretanha</i> e <i>Afeganistão</i> )
Regra dos Símbolos	Valores (como <i>R\$2,00</i> ) Fórmulas químicas (como <i>H5NI</i> )

**Tabela 1. Exemplos de Palavras Filtradas**

Visando verificar essa possibilidade, realizou-se outro experimento em que regras foram sucessivamente adicionadas ao processo de classificação, seguindo o fluxo estabelecido pelo processo proposto, em que executam-se em ordem as regras de Localização,

Filtragem e Expansão. A Figura 6 ilustra os resultados obtidos.



**Figura 6. Evolução de Eficiência ao Longo das Regras**

Conforme pode-se observar, o desempenho da classificação aumenta conforme novas regras são adicionadas. É interessante destacar a importância da existência de Regras de Expansão. No caso avaliado, a Regra de Abreviação de Nome conseguiu aumentar a cobertura sem perda na precisão, o que resultou em um valor melhor da Medida F. Ao final do processo, atingiu-se uma Medida F equivalente a 40,72%, o que é um resultado razoavelmente expressivo, considerando o uso de técnicas de mineração não-assistidas.

#### 4. Trabalhos Relacionados

O problema de extração de entidades (ou reconhecimento de entidades) nomeadas contempla tanto a localização das entidades nomeadas quanto uma identificação mais precisa dos tipos específicos dessas entidades as quais esse conjunto se refere. As duas tarefas podem ser realizadas por algoritmos de classificação, sendo que na localização os termos são classificados em duas categorias (positivo ou negativo) enquanto na identificação o número de categorias é maior, e depende da aplicação [Nadeau and Sekine 2007].

Os algoritmos de classificação podem ser supervisionados, não supervisionados e semi-supervisionados. Os algoritmos supervisionados partem de *córpus* textuais pré-marcados e um conjunto de dimensões (*features*), e descobrem regras sobre as dimensões que determinam em que categoria os termos se enquadram [McCallum and Li 2003]. Esse tipo de abordagem é útil para tarefas de classificação em que as regras são complexas demais para serem definidas manualmente, como por exemplo, para identificar inequivocamente o tipo de entidade localizada. No entanto, elas dependem da presença de entidades nomeadas pré-rotuladas(também chamadas de *gazetters*) para funcionar.

Os algoritmos semi-supervisionados não dependem propriamente de *gazetters*, mas partem de algumas sementes (*seeds*) referentes ao tipo de entidade que se deseja buscar. As sementes são padrões textuais que costumam aparecer junto ao tipo de entidade de interesse. Sentenças que contém essas sementes são recuperadas, e a partir da análise dessas sentenças se busca identificar novas entidades e novos padrões de texto relevantes. Os padrões encontrados se transformam em novas sementes, de modo que o processo de busca e análise possa ocorrer sucessivamente, até que um limiar seja atingido.

[Pasca et al. 2006] Esse tipo de abordagem é útil quando se pretende descobrir entidades nomeadas existentes em um conjunto de documentos indexados.

Assim como os algoritmos supervisionados, os não supervisionados também usam regras sobre dimensões para realizar o processo de descoberta. Uma abordagem típica adotada envolve a clusterização, que agrupa termos que possivelmente se referem ao mesmo tipo de entidade nomeada [Cucchiarelli and Velardi 2001]. Além disso, alguns trabalhos mais específicos podem se valer de outras técnicas que não recorram ao aprendizado de máquina. Por exemplo, [Alfonseca and Manandhar 2002] descreve uma técnica que atribui um tipo de entidade nomeada a uma palavra com base no análise do contexto da palavra (as palavras que estão a sua volta). Se o contexto costuma aparecer mais seguidamente associado a um tipo específico de EN, a palavra é considerada como sendo do mesmo tipo.

De modo geral, existem uma série de dimensões (*features*) que podem ser usadas para o reconhecimento e identificação de ENs, tanto em abordagens supervisionadas quanto não supervisionadas. Os exemplos mais comuns envolvem análise de maiúsculas/minúsculas, dígitos para a identificação dos componentes de uma data, análise dos radicais e afixos de palavras [Bick 2004] e padrões summarizadores que atribuem um tipo de dados a palavra [Collins 2002].

Também existem dimensões estatísticas, relacionadas a fatos presentes no *corpus textual* estudado. Por exemplo, palavras que aparecem tanto na forma maiúscula quanto na forma minúscula são consideradas como sendo o mesmo tipo de substantivo que em alguns casos aparece no início de frases [Mikheev 1999]. O conceito de raridade também é utilizado para o reconhecimento de entidades nomeadas. Por exemplo, em [da Silva et al. 2004], palavras compostas não são consideradas entidades nomeadas caso elas possuam um termo longo e raro. Já [Shinyama and Sekine 2004] parte da observação de que textos de notícia de uma mesma época costumam conter a mesma entidade. Isso permite descobrir entidades nomeadas raras, mas que foram destaque em algum período específico no tempo. Curiosamente, os dois últimos trabalhos citados utilizam a frequência de modo inverso ao adotado neste artigo. Ou seja, quanto menos frequente o termo, menores as chances de ele ser uma entidade nomeada. Contudo, é importante destacar que a regra de Frequência usada neste trabalho tem um foco ortogonal, uma vez que o intuito é de filtragem, e não a de descoberta de candidatos.

## 5. Conclusão

Este artigo apresentou um processo para reconhecimento de Entidades Nomeadas do tipo 'nome de pessoa' baseada na aplicação de regras em três etapas distintas: localização, filtragem e expansão. Também foram propostos exemplos de regras que poderiam ser executadas dentro de cada uma das etapas.

Os resultados na classificação dos nomes de pessoas usando o repositório marcado *HAREM* mostraram que as regras conseguem atingir uma cobertura elevada e uma precisão baixa, o que gerou um escore de Medida F intermediário. Isso sugere que novas Regras de Filtragem e Expansão precisam ser concebidas, as primeiras para remover palavras da lista de nomes e a segunda para corrigir eventuais enganos cometidos durante a filtragem.

Além da criação de novas regras, deseja-se também parametrizar algumas das

regras propostas, como as baseadas em tamanhos mínimos e máximos das palavras. Para isso, será verificado através de experimentos a existência de um valor adequado a ser usado como ponto de corte por número de caracteres.

Como análise final, concluímos que as regras testadas atingiram um desempenho satisfatório, considerando a inexistência de uma etapa prévia de treinamento. Para fins de comparação, pretende-se implementar um algoritmo supervisionado de aprendizado de máquina baseada nessas mesmas regras, e verificar em que aspectos as abordagens supervisionada e a não supervisionada se diferenciam.

## Referências

- Alfonseca, E. and Manandhar, S. (2002). An unsupervised method for general named entity recognition and automated concept discovery. In *In: Proceedings of the 1 st International Conference on General WordNet*.
- Bick, E. (2004). A named entity recognizer for danish. In *LREC*. European Language Resources Association.
- Collins, M. (2002). Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 489–496, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Cucchiarelli, A. and Velardi, P. (2001). Unsupervised named entity recognition using syntactic and semantic contextual evidence. *Comput. Linguist.*, 27(1):123–131.
- da Silva, J. F., Kozareva, Z., and Lopes, J. G. P. (2004). Cluster analysis and classification of named entities. In *LREC*. European Language Resources Association.
- Krishnan, V. and Manning, C. D. (2006). An effective two-stage model for exploiting non-local dependencies in named entity recognition. In Calzolari, N., Cardie, C., and Isabelle, P., editors, *ACL*. The Association for Computer Linguistics.
- McCallum, A. and Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Mikheev, A. (1999). A knowledge-free method for capitalized word disambiguation. In Dale, R. and Church, K. W., editors, *ACL*. ACL.
- Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26. Publisher: John Benjamins Publishing Company.
- Pasca, M., Lin, D., Bigham, J., Lifchits, A., and Jain, A. (2006). Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, pages 1400–1405. AAAI Press.
- Shinyama, Y. and Sekine, S. (2004). Named entity discovery using comparable news articles. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

# Métodos Estatísticos para Segmentação de Listas Web

William Marx<sup>1</sup>, Sergio L. S. Mergen<sup>1</sup>

<sup>1</sup>Campus Alegrete - Universidade Federal do Pampa (UNIPAMPA)  
CEP – 97.546-550 – Alegrete – RS – Brasil

william.f.marx@gmail.com, sergiomergen@unipampa.edu.br

**Abstract.** *Data extraction from HTML lists has the goal of transforming a record list into a tabular format, composed by records and columns. This sort of extraction serves many goals, such as feeding a dataspace that associates Web data sources that refer to the same subject. There already exist proposals of mechanisms that perform this extraction. One of them in particular divides the extraction process in three stages (Splitting, Alignment and Refinement), where the first one relies on a preexistent knowledge base. In this paper, we propose extraction rules whose purpose is to replace the initial Splitting stage. Such rules explore the existence of content delimiters in the list records, and they are not dependent on previous information. Experiments show the efficiency of the extraction rules applied over real HTML lists found on the Web, in comparison with the method currently used in the Splitting stage.*

**Resumo.** *A extração de dados a partir de listas HTML tem por finalidade transformar uma lista de registros em um formato tabular composto por registros e colunas. Essa extração serve a diversas finalidades, como alimentar um dataspace que relaciona fontes de dados da Web que tratam de um assunto comum. Já existem mecanismos propostos que realizam essa extração. Um deles em particular divide o processo de extração em três etapas(Splitting, Alignment e Refinement) sendo que a primeira delas depende de uma base de conhecimento preexistente. Neste artigo são propostas regras de extração cujo objetivo é substituir a etapa inicial de Splitting. Tais regras exploram a existência e frequência de delimitadores de conteúdo nas linhas da lista, e independem de informações prévias para funcionar. Os experimentos mostram a eficácia das regras na extração aplicadas em listas HTML reais encontradas na Web, em comparação com o método atualmente usado na etapa de Splitting.*

## 1. Introdução

A produção de dados na *Web* vem crescendo exponencialmente, desde simples páginas HTML até formatos mais sofisticados como *feeds RSS* e *Web Services*. De certa forma, essas informações podem ser vistas como fazendo parte de uma gigantesca base de dados heterogênea e sem uma autoridade central que regula a estrutura dos dados e tampouco as políticas para adição e remoção de informações.

Indo um pouco mais além, é possível criar uma linha imaginária que divida essa gigantesca base de dados em bases menores, para os diferentes tipos de domínios existentes. Desses bases menores surge um conceito que está em voga nos dias atuais: *dataspaces*. Em poucas palavras, um *dataspace* pode ser descrito como um conjunto de fontes de

dados heterogêneas que atendem a um propósito comum [Franklin et al. 2005]. A possibilidade de acessar dados de um *dataspace* aumenta a qualidade das pesquisas, e tem aplicação direta em áreas como extração de conhecimento e recuperação de informação.

Um dos desafios a ser vencido para que *dataspaces* saiam do mundo das idéias e se tornem elementos tangíveis envolve reconhecer a estrutura presente nas fontes de dados, para que mais adiante um processo de integração possa consolidar esses dados em uma base unificada, seja ela virtual ou materializada. Esse problema se torna ainda mais desafiante quando a estrutura é implícita, oculta por detrás de padrões de documentação difíceis de ser identificados.

Esse é o caso por exemplo dos dados publicados nas páginas HTML. Em boa parte dos casos, o conteúdo das páginas HTML são textos escritos em linguagem natural, que por vezes são envoltos em *tags* de marcação, que mais tem o fim de definir a formatação visual do documento do que criar uma estrutura que organize o texto em conceitos semânticos. Ou seja, informações descritas em páginas HTML tendem a ser pobres em estrutura. Ironicamente, esses documentos são a fonte mais rica de informações da Web, visto que a linguagem HTML é o padrão de facto para a publicação de dados na Web.

Dentro da linguagem HTML, existem construtores que, embora voltados primariamente para formatação visual, servem também para estruturar a informação. Um desses construtores são as listas. Verificando blocos de texto contidos em listas, é possível perceber uma divisão inicial que organiza a informação como uma coleção de registros. No entanto, cada registro por si só pode ser estruturado, dividindo o conteúdo em campos que representem informações distintas, mas que estejam associadas.

Dada essas características das listas HTML, surgiram trabalhos acadêmicos que visam transformar os blocos de dados contidos nas listas em tabelas de dados, compostas por linhas e colunas. Um desses trabalhos em especial define processos a serem executados sobre uma lista, para que ao fim seja possível realizar a transformação. No entanto, os processos requerem o uso de bases de conhecimento para auxiliar na identificação das colunas contidas em cada linha.

Dentro deste contexto, este trabalho propõe uma série de regras estatísticas que visam transformar listas em tabelas. De modo geral, as regras se valem da presença e frequência no texto de caracteres especiais que costumam ser usados para realizar a separação de conteúdo. Ainda, as regras independem da existência de bases de conhecimento, o que torna o mecanismo de extração útil em situações onde não existe uma base que possa ser usada ou a base esteja indisponível.

O texto está organizado da seguinte forma: A seção 2 apresenta alguns trabalhos publicados a respeito de extração de dados na Web, inclusive o trabalho que será utilizado como base de comparação nos experimentos. A seção 3 apresenta a abordagem proposta, descrevendo cada uma das regras criadas, e as situações em que elas são úteis. Os resultados dos experimentos são discutidos na seção 4. Por fim, a seção 5 apresenta as considerações finais.

## 2. Trabalhos Relacionados

Existem diversos estudos a respeito de abordagens automatizadas que extraem listas de objetos a partir da *Web* [Arasu and Garcia-Molina 2003, Crescenzi et al. 2001]. O processo típico de extração consiste em três passos: extração de registros, extração de atributos e rotulação.

O primeiro passo envolve identificar os registros presentes no documento em análise. Em alguns tipos de documentos essa identificação é trivial, como em listas HTML, onde as próprias linhas já separam os registros. Já em páginas na *Web*, esses registros seriam segmentos no texto que encapsulam dados referentes a um objeto específico. Entre as técnicas usadas para realizar essa busca pode-se citar o uso de algoritmos de casamento de strings [Liu et al. 2003] e processamento de linguagem natural.

O segundo passo envolve extrair atributos do objeto identificado. Considerando objetos contendo dados de filme, esses atributos poderiam ser 'título' e 'ano', por exemplo. O último passo envolve interpretar os atributos identificados, ou o próprio objeto que os agrupa, afim de rotulá-los com nomes apropriados. Normalmente, as soluções propostas para esse problema são baseadas modelos probabilísticos e aprendizado de máquina [Wang and Lochovsky 2003, Zhu et al. 2006].

Os trabalhos mais próximos ao proposto neste artigo são aqueles que realizam o segundo passo. Existem diversos trabalhos nesta categoria, como aqueles que analisam páginas de resposta de formulários *Web* para realizar a extração de atributos. Por exemplo, em [Zhao et al. 2007] é usado um método estatístico para minerar páginas de resposta na busca de dados que obedeçam *templates* pré-definidos. Já o trabalho de [Zhai and Liu 2005] utiliza informação visual de como os dados seriam renderizados pelo navegador de *Internet* para inferir como esses dados estão estruturados.

Existem também estudos mais específicos, e ainda mais próximos a este trabalho, cujo foco é a extração de atributos de listas HTML. Em [Machanavajjhala et al. 2011], a extração é realizada de forma coletiva. Ou seja, múltiplas listas são extraídas de forma simultânea, explorando a redundância de conteúdo e estrutura contidas nas listas analisadas.

Neste artigo, será dada ênfase ao trabalho proposto por [Elmeleegy et al. 2009], que é capaz de extrair atributos de uma lista por vez. A técnica de extração empregada possui três fases, chamadas de Divisão (*Splitting*), Alinhamento (*Alignment*) e Refinamento (*Refinement*). Na fase de divisão, cada linha é dividida em múltiplos campos. Mais informações sobre essa divisão serão descritas mais adiante.

Na fase de alinhamento é calculado um número esperado de colunas por linha, com base nas divisão inicial feita. As linhas que tenham ficado com menos ou mais colunas do que o esperado são ajustadas.

Na fase de divisão, uma linha é separada em vários campos candidatos, sendo que a cada campo é atribuído um escore. Em seguida é aplicada uma técnica de seleção dos melhores campos, que leva em consideração o escore e a inexistência de sobreposição entre os campos selecionados.

Vale a pena destacar que a geração dos campos candidatos e seus escores é feita de acordo com três escores parciais, que são ponderados e normalizados para a obtenção do escore final:

1. **Escore de Tipo:** Um campo recebe escore máximo se o seu conjunto de palavras corresponder a um tipo de informação reconhecido, como valores numéricos, datas, *urls*, *emails* e telefones.
2. **Escore de Modelo de Linguagem:** Calcula o escore de um campo com base na probabilidade condicional de cada uma das palavra do campo  $w_i$  seguir uma sequência das palavras anteriores do campo  $w_1, \dots, w_{i-1}$ , ou seja, calcula  $P_r(w_i|w_1, \dots, w_{i-1})$ . O escore é calculado com base na probabilidade de que a primeira palavra do campo em questão siga a última palavra do campo candidato anterior.
3. **Escore de Compatibilidade de Tabela:** Compara o conteúdo das listas com um *corpus* de tabelas, afim de identificar quais informações normalmente aparecem como parte de uma mesma coluna.

Apesar de a técnica proposta por [Elmeleegy et al. 2009] ser não supervisionada, durante a fase de divisão os escores parciais de Modelo de Linguagem e de Compatibilidade de Tabela são calculados com base em um *corpus* textual e um *corpus* de tabelas, respectivamente.

O único escore parcial que independe de dados externos é o de Tipo, que procura por padrões textuais dentro das linhas da lista. A dependência da etapa de divisão por bases de conhecimentos é uma limitação da abordagem, que impede que ela seja usada integralmente para separar os atributos de listas sem uso de informações preexistentes.

### 3. Métodos de Divisão Propostos

Com base no exposto na seção anterior, este trabalho propõe novas regras que podem ser usadas para substituir a etapa de divisão proposta em [Elmeleegy et al. 2009]. As regras baseiam-se na noção de que existem caracteres específicos que costumam ser utilizados para fazer a delimitação de conteúdo.

Para exemplificar, considere o trecho “Star Wars - George Lucas”. Nesse trecho, o hífen é usado para separar o nome do diretor do filme que ele dirigiu. Assim como o hífen, existem outros caracteres que servem para separar informações distintas, mas que estejam relacionadas. Neste trabalho, esse conjunto de caracteres é chamado de caracteres de separação.

A determinação de quais caracteres devem compor essa lista está fora do escopo deste artigo. Para simplificar, adota-se um critério de seleção que considera os caracteres não alfanumérico como separadores (ex. “-” e “:”). Esse critério parte da intuição de que textos normalmente são compostos por caracteres alfanuméricos, e que é mais elegante usar como separador os caracteres que não costumam aparecer no corpo do texto.

Observe que nem sempre os caracteres de separação são usados para separar conteúdo. Por exemplo, em “Batman - O Cavaleiro das Trevas, Christopher Nolan (2008)”, o hífen tem uma função de apostrofe especificador que aparece em meio a um título de filme. Já em “Star Wars Episódio I: A Ameaça Fantasma - 1999”, o hífen é usado com a função de separador. Com base nesse exemplo, surge a necessidade de analisar o contexto onde os caracteres de separação ocorrem para determinar sua real função.

Nesta seção são descritos os métodos de segmentação de conteúdo propostos, cujo objetivo é encontrar os caracteres que exercem a função de separação. Os métodos criados

procuram inferir o contexto através de uma análise estatística, determinando a importância de cada caractere de separação com base na frequência com que ele ocorre na lista. A Figura 3.1 é usada para apoiar a explicação. Ela contém exemplos de listas, métodos usados na transformação e a respectiva segmentação, representada na forma de uma tabela com linhas e colunas.

### 3.1. Método Caractere Sempre Presente

Este método busca um caractere especial com maior número de ocorrências que necessariamente está presente em todas as linhas da lista.

Um exemplo de segmentação usando este método pode ser visto lista 'A' da Figura 1, onde a informação foi corretamente segmentada. Este método também foi aplicado nas listas 'B' e 'D' da Figura 1. Como em ambas existe mais de um tipo de caractere separador, o método só conseguiu acertar a primeira coluna.

<b>Lista Original / Métodos Aplicados:</b>	<b>Resultado:</b>															
<b>Lista A / Métodos: 1, 2, 3, 4, 5, 6</b>	<table border="1"> <thead> <tr> <th>Coluna1</th><th>Coluna2</th><th>Coluna3</th></tr> </thead> <tbody> <tr> <td>1973</td><td>Metamorfose Ambulante</td><td>3min50s</td></tr> <tr> <td>1974</td><td>Medo da Chuva</td><td>3min</td></tr> <tr> <td>1976</td><td>Eu Nasci Há 10 Mil Anos Atrás</td><td>4min52s</td></tr> <tr> <td>1987</td><td>Maluco Beleza</td><td>-</td></tr> </tbody> </table>	Coluna1	Coluna2	Coluna3	1973	Metamorfose Ambulante	3min50s	1974	Medo da Chuva	3min	1976	Eu Nasci Há 10 Mil Anos Atrás	4min52s	1987	Maluco Beleza	-
Coluna1	Coluna2	Coluna3														
1973	Metamorfose Ambulante	3min50s														
1974	Medo da Chuva	3min														
1976	Eu Nasci Há 10 Mil Anos Atrás	4min52s														
1987	Maluco Beleza	-														
<b>Lista B / Métodos: 1, 2, 3, 4</b>	<table border="1"> <thead> <tr> <th>Coluna1</th><th>Coluna2</th><th>Coluna3</th></tr> </thead> <tbody> <tr> <td>1973</td><td>Metamorfose Ambulante (3min50s)</td><td>-</td></tr> <tr> <td>1974</td><td>Medo da Chuva (3min)</td><td>-</td></tr> <tr> <td>1976</td><td>Eu Nasci Há 10 Mil Anos Atrás (4min52s)</td><td>-</td></tr> <tr> <td>1987</td><td>Maluco Beleza (3min25s)</td><td>-</td></tr> </tbody> </table>	Coluna1	Coluna2	Coluna3	1973	Metamorfose Ambulante (3min50s)	-	1974	Medo da Chuva (3min)	-	1976	Eu Nasci Há 10 Mil Anos Atrás (4min52s)	-	1987	Maluco Beleza (3min25s)	-
Coluna1	Coluna2	Coluna3														
1973	Metamorfose Ambulante (3min50s)	-														
1974	Medo da Chuva (3min)	-														
1976	Eu Nasci Há 10 Mil Anos Atrás (4min52s)	-														
1987	Maluco Beleza (3min25s)	-														
<b>Lista C / Métodos: 5</b>	<table border="1"> <thead> <tr> <th>Coluna1</th><th>Coluna2</th><th>Coluna3</th></tr> </thead> <tbody> <tr> <td>1973</td><td>Metamorfose Ambulante</td><td>3min50s</td></tr> <tr> <td>1974</td><td>Medo da Chuva</td><td>3min</td></tr> <tr> <td>1976</td><td>Eu Nasci Há 10 Mil Anos Atrás</td><td>4min52s</td></tr> <tr> <td>1987</td><td>Maluco Beleza</td><td>-</td></tr> </tbody> </table>	Coluna1	Coluna2	Coluna3	1973	Metamorfose Ambulante	3min50s	1974	Medo da Chuva	3min	1976	Eu Nasci Há 10 Mil Anos Atrás	4min52s	1987	Maluco Beleza	-
Coluna1	Coluna2	Coluna3														
1973	Metamorfose Ambulante	3min50s														
1974	Medo da Chuva	3min														
1976	Eu Nasci Há 10 Mil Anos Atrás	4min52s														
1987	Maluco Beleza	-														
<b>Lista D / Métodos: 1, 2, 3, 4, 6</b>	<table border="1"> <thead> <tr> <th>Coluna1</th><th>Coluna2</th><th>Coluna3</th></tr> </thead> <tbody> <tr> <td>1973</td><td>Metamorfose Ambulante (3min50s)</td><td>-</td></tr> <tr> <td>1974</td><td>Medo da Chuva (3min)</td><td>-</td></tr> <tr> <td>1976</td><td>Eu Nasci Há 10 Mil Anos Atrás (4min52s)</td><td>-</td></tr> <tr> <td>1987</td><td>Maluco Beleza</td><td>-</td></tr> </tbody> </table>	Coluna1	Coluna2	Coluna3	1973	Metamorfose Ambulante (3min50s)	-	1974	Medo da Chuva (3min)	-	1976	Eu Nasci Há 10 Mil Anos Atrás (4min52s)	-	1987	Maluco Beleza	-
Coluna1	Coluna2	Coluna3														
1973	Metamorfose Ambulante (3min50s)	-														
1974	Medo da Chuva (3min)	-														
1976	Eu Nasci Há 10 Mil Anos Atrás (4min52s)	-														
1987	Maluco Beleza	-														

**Legenda:**

- |                             |                                    |                                           |
|-----------------------------|------------------------------------|-------------------------------------------|
| 1: Caracter Sempre Presente | 2: Melhor Caractere                | 3: Caractere com a Menor Variação         |
| 4: Um Caractere             | 5: Conjunto de Melhores Caracteres | 6: Conjunto de Caracteres Sempre Presente |

**Figura 1. Exemplos de Segmentação Baseados nos Métodos Propostos**

### 3.2. Método Melhor Caractere

Este método busca pelo caractere especial com maior número de ocorrências, sendo que o caractere não necessariamente precisa ocorrer em todas as linhas da lista.

O desempenho do método nos exemplos é igual ao do método anterior. Seu uso se justifica para casos de erros de digitação, onde o caractere de separação não foi incluído em alguma das linhas da tabela.

### **3.3. Método de Um Caractere**

Este método busca pelo caractere especial com maior número de ocorrências em cada linha, e usa-o como separador na linha analisada. Isso é feito em cada linha da lista possibilitando que um caractere diferente seja escolhido em cada linha.

O desempenho do método nos exemplos é igual ao dos métodos anteriores. Seu uso se justifica para casos em que uma mesma lista é composta por tipos diferentes de linhas, situação que é bastante comum em arquivos do tipo *flat file*.

### **3.4. Método Caractere com a Menor Variação**

Este método busca apenas um caractere separador para toda a lista, sendo que é escolhido o que apresentar a menor variação do número de ocorrências em relação a média de ocorrências de todos os caracteres encontrados na lista.

O desempenho do método nos exemplos é igual ao dos métodos anteriores. Seu uso se justifica para casos em que caracteres especiais ocorram em todas as linhas mas que não sejam de fato usados como separadores. A regra parte da intuição que esses caracteres terão uma frequência irregular pelas linhas da lista.

### **3.5. Método Conjunto de Melhores Caracteres**

Este método faz a segmentação utilizando um conjunto de caracteres especiais como delimitadores de informação, sendo que este conjunto é o mesmo para todas as linhas da lista.

Ao contrário dos métodos anteriores, este pode selecionar um número variável de caracteres de separação. Para que um caractere seja selecionado, deve-se verificar se o número de ocorrências dele é maior que a média de ocorrências dos caracteres especiais por linha subtraída do seu desvio padrão.

Por utilizar mais caracteres de separação, o método conseguiu segmentar corretamente as colunas tanto das listas 'A' e 'C', conforme apresentado na Figura 1.

### **3.6. Método Conjunto de Caracteres Sempre Presente**

Este método também faz a segmentação utilizando um conjunto de caracteres especiais como delimitadores de informação, sendo que este conjunto é o mesmo para todas as linhas da lista.

O método busca por todos os caracteres especiais que estão presentes necessariamente em todas as linhas da lista, independentemente do número de ocorrências em cada linha.

Apesar de trabalhar com um conjunto de caracteres, o método não segmentou corretamente a lista 'D' da Figura 1. Isso ocorreu pelo fato desta lista utilizar como separador de conteúdo os caracteres '(' e ')', que não aparecem em todas as linhas.

## **4. Experimentos**

Esta seção apresenta os experimentos realizados para a segmentação de listas encontradas na *Web*. A seguir são apresentados os aspectos relacionados aos experimentos.

## 4.1. Criação dos conjuntos de teste

Para validar este trabalho foram construídos três conjuntos de testes, cada um constituído de 50 listas *Web*. As listas foram coletadas manualmente, seguindo os seguintes critérios:

- As linhas não sejam parte de listas de menu
- As linhas são compostas por textos curtos (não mais do que 500 caracteres por linha)
- As linhas possuem caracteres de separação

Com base nesses critérios, foram coletadas páginas considerando três contextos distintos, conforme descrito abaixo:

**Wikipedia** : Esta coleção possui listas aleatórias recuperadas a partir de páginas da *Wikipedia*. A geração da coleção foi realizada através de um recurso da *Wikipedia* que direciona o usuário a uma página aleatória. Nesta página, foram coletadas todas as listas que satisfizeram os critérios definidos acima. Em seguida, uma nova página aleatória foi acessada, e o processo se repetiu até que 50 listas tivessem sido coletadas.

**Listas 10** : Esta coleção possui listas aleatórias existentes em sites que informam os 10 tópicos mais relevantes a respeito de um determinada categoria. Foram utilizadas aproximadamente 50 páginas do site *listas10.org* durante a coleta. Dentre as categorias coletadas encontram-se Cinema, Músicas e Esportes.

**WT10G** : Esta coleção possui listas aleatórias recuperadas a partir da *Web*. A geração da coleção foi realizada através de um *snapshot* da *Web* de 1997, contida na *corpus* WT10G<sup>1</sup>. As páginas da coleção estão divididas em múltiplas pastas numeradas, sendo cada pasta dividida em múltiplos documentos XML numerados. Cada documento XML também é dividido em blocos numerados, referentes a uma página específica. Navegando aleatoriamente por essa estrutura, foi recuperada uma página a partir de onde foram coletadas as listas que satisfizeram os critérios definidos acima. Em seguida, uma nova página aleatória foi acessada, e o processo se repetiu até que 50 listas tivessem sido coletadas.

## 4.2. Marcação das Listas

As listas foram manualmente marcadas com a segmentação consideradas correta. Essa marcação é utilizada para medir a eficácia dos métodos propostos. Para fazer a segmentação, procurou-se dividir as linhas em colunas que pudessem ser rotuladas de modo simples e intuitivo.

Para exemplificar, a linha “Batman(1987)” seria separada em duas colunas, pois é intuitivo identificar que a primeira coluna refere-se ao nome de um filme enquanto a segunda se refere ao ano do filme. Já a linha “Batman bateu recorde de público (e de arrecadação também)” não seria separada, pois faz mais sentido manter toda a informação em apenas uma coluna, que poderia ser rotulada como “Notícia”.

Outro exemplo envolve a presença de atributos multivvalorados, que foram mantidos em um único segmento. Por exemplo, a linha “Batman: 1989, 1992, 1995, 1997” seria separada em dois segmentos, um para o nome do filme e outro para os anos de

---

<sup>1</sup>[http://ir.dcs.gla.ac.uk/test\\_collections/wt10g.html](http://ir.dcs.gla.ac.uk/test_collections/wt10g.html)

lançamento. Do contrário, seria necessário criar rótulos separados para cada ano, o que não é visto como uma boa prática de modelagem, até porque o número de valores de um atributo multivalorado é dinâmico.

#### 4.3. Forma de Avaliação

A avaliação foi realizada através da medição da precisão na segmentação. Três níveis distintos de precisão foram medidos, conforme descrito abaixo:

**Precisão dos Segmentos** Verifica quantos dos segmentos que deveriam ser separados foram realmente identificados.

**Precisão das Linhas** Verifica quantas linhas foram devidamente segmentadas.

**Precisão das Listas** Verifica quantas listas tiveram todas suas linhas devidamente segmentadas.

A precisão das linhas e listas é mais restrita do que a precisão dos segmentos. Um único segmento não identificado na linha torna essa linha inválida, para fins de medição. Da mesma forma, uma única linha inválida em uma lista torna essa lista inválida.

#### 4.4. Base de Comparação

Além de os métodos serem comparados entre si, eles também são comparados com uma das técnicas utilizadas no trabalho de [Elmeleegy et al. 2009], que segmenta uma lista com base na análise de padrões textuais presentes em cada linha.

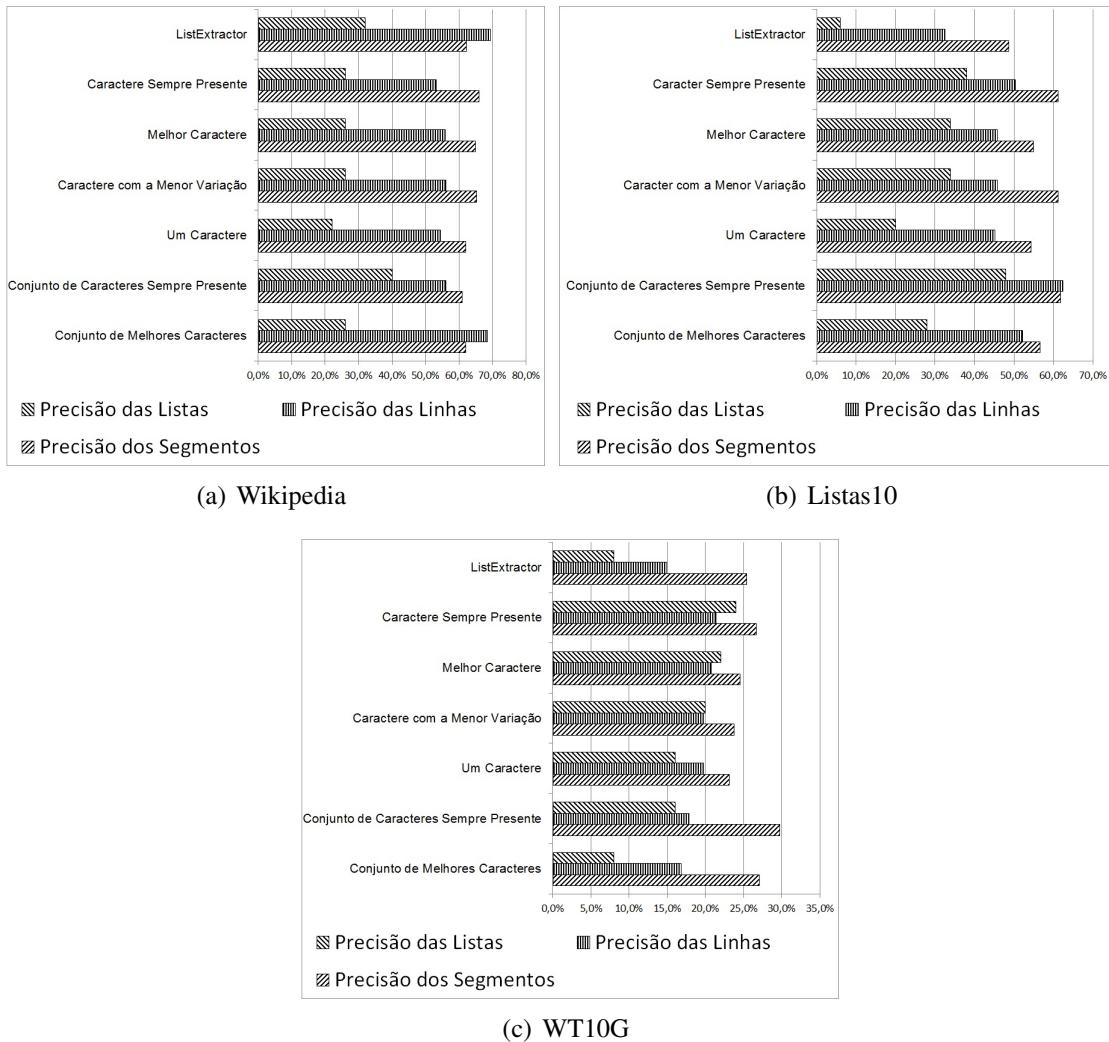
Para possibilitar a avaliação, a técnica foi implementada através de expressões regulares que reconhecem padrões textuais conhecidos, como datas, telefones e *emails*. Conforme descrito em [Elmeleegy et al. 2009], essa é uma das técnicas propostas para compor a etapa de *splitting*. As demais técnicas não foram avaliadas neste artigo uma vez que elas dependem de bases de conhecimentos preexistentes.

#### 4.5. Análise dos Resultados

Os resultados alcançados pelos métodos de segmentação são apresentados na Figura 2. Como se pode ver, todas as técnicas obtiveram desempenho semelhante na coleção 'Wikipedia'. Destaque é dado para o método 'Conjunto de Caracteres Sempre Presente', que se sobressaiu no critério relativo ao número de listas corretamente segmentadas. Esse resultado indica que as listas publicadas em artigos da *Wikipedia* costumam usar caracteres de separação de uma forma bem comportada. O desempenho satisfatório do método 'ListExtractor' sugere que os segmentos das listas costumam usar padrões textuais bem definidos.

Na coleção 'listas 10', as listas também usam caracteres de separação de uma forma bem comportada, sendo que o método 'Conjunto de Caracteres Sempre Presente' mais uma vez se saiu melhor. Nesta coleção, destaca-se o baixo desempenho do método 'ListExtractor'. Uma possível explicação para esse fato é o número insuficiente de expressões regulares que foram implementadas. Por exemplo, valores monetários foram bastante comuns nas listas desta coleção, e nenhuma das expressões usadas conseguiam identificar esse padrão textual.

Já o desempenho geral dos métodos da coleção 'WT10G' foi bastante inferior se comparados às demais coleções. Nenhum dos métodos obteve uma precisão maior do



**Figura 2. Resultados de Precisão Obtidos sobre Três Coleções de Listas Reais da Web**

que 30% na segmentação de listas inteiras. Como os dados não pertencem a um domínio específico, onde padrões de exibição costumam ser adotados, os caracteres usados na separação não seguiam um formato bem comportado. Mesmo assim, percebe-se que o desempenho em todos os métodos propostos no artigo superam a técnica baseado em padrões textuais.

## 5. Conclusões

Este artigo apresentou métodos estatísticos que realizam a segmentação de listas. A segmentação é realizada com base na frequência de ocorrência de um conjunto de caracteres especiais que costumam ser usados para fazer a separação de conteúdo. O objetivo do trabalho foi verificar se os métodos propostos podem ser usados dentro de uma arquitetura de segmentação proposta na literatura([Elmeleegy et al. 2009]), como substituto de um de seus métodos internos que analisa a presença de padrões textuais no texto.

Foram realizados experimentos sobre coleções de listas reais extraídas da *Web*. Os resultados mostram que os métodos propostos têm um desempenho igual ou su-

perior ao método baseado em padrões textuais quando as listas utilizam caracteres de separação. O próximo passo é empregar os métodos propostos dentro da arquitetura de [Elmeleegy et al. 2009] afim de analisar como a segmentação se comporta.

Outra possibilidade de trabalho futuro envolve descobrir dinamicamente qual método é mais eficaz na segmentação de uma lista. Para atingir esse objetivo, pretende-se analisar as evidências exploradas pelas regras, na busca por correlações que indiquem a melhor forma de realizar a segmentação para cada caso específico.

## Referências

- Arasu, A. and Garcia-Molina, H. (2003). Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 337–348, New York, NY, USA. ACM.
- Crescenzi, V., Mecca, G., and Merialdo, P. (2001). Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 109–118, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Elmeleegy, H., Madhavan, J., and Halevy, A. (2009). Harvesting relational tables from lists on the web. *Proceedings of the VLDB Endowment*, 2(1):1078–1089.
- Franklin, M., Halevy, A., and Maier, D. (2005). From databases to dataspaces: a new abstraction for information management. *SIGMOD Rec.*, 34(4):27–33.
- Liu, B., Grossman, R., and Zhai, Y. (2003). Mining data records in web pages. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 601–606, New York, NY, USA. ACM.
- Machanavajjhala, A., Iyer, A. S., Bohannon, P., and Merugu, S. (2011). *Collective extraction from heterogeneous web lists*. ACM Press.
- Wang, J. and Lochovsky, F. H. (2003). Data extraction and label assignment for web databases. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 187–196, New York, NY, USA. ACM.
- Zhai, Y. and Liu, B. (2005). Web data extraction based on partial tree alignment. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 76–85, New York, NY, USA. ACM.
- Zhao, H., Meng, W., and Yu, C. (2007). Mining templates from search result records of search engines. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 884–893, New York, NY, USA. ACM.
- Zhu, J., Nie, Z., Wen, J.-R., Zhang, B., and Ma, W.-Y. (2006). Simultaneous record detection and attribute labeling in web data extraction. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 494–503, New York, NY, USA. ACM.

# **Uma aplicação de Rede Social de Consumo Baseada em uma Arquitetura de Data Warehouse**

**Holisson S. da Cunha<sup>1</sup>, Sergio L. S. Mergen<sup>1</sup>**

<sup>1</sup> Campus Alegrete - Universidade Federal do Pampa (UNIPAMPA)  
Caixa Postal 97546-550 – Alegrete – RS – Brasil

holissonsud@gmail.com, sergiomergen@unipampa.edu.br

**Abstract.** This paper describes a data warehousing architecture targeted at the problem of collecting and analyzing products offers available in the Web. Besides presenting the architecture, we describe the modules that were implemented in order to build an social network application that wraps the access to the data warehouse. One of the implemented modules allows collecting data interactively, by gathering information provided by the users of the social network themselves. Another module that deserves attention uses the k-means classifier to automatically divide the users in communities based on their financial profile. Experiments show how the classifier behaves for dividing users in a fixed number of communities given a controlled testing scenario.

**Resumo.** Esse artigo descreve uma arquitetura de data warehouse voltada para o problema da coleta e análise de dados de ofertas de produtos disponibilizados na Web. Além de apresentar a arquitetura, serão descritos os módulos que foram implementados tendo em vista a criação de uma aplicação de rede social que encapsula o acesso ao data warehouse. Um dos módulos implementados permite a coleta de dados de maneira interativa, a partir de dados de consumo informados pelos usuários da rede social. Outro módulo que merece destaque utiliza o algoritmo de classificação k-means para automaticamente dividir os usuários em comunidades com base no seu perfil de consumo. Os experimentos relatam como o algoritmo se comporta na divisão dos usuários em um número fixo de comunidades dado um cenário de teste controlado.

## **1. Introdução**

Com a popularização da internet e o avanço das tecnologias de informação, o comércio eletrônico tem crescido com rapidez. Apenas em 2011 foram gastos R\$ 18,7 bilhões em compras pela internet, o que permite visualizar o aumento da procura de serviços e produtos pelos usuários.

A qualidade no serviço e o melhor preço são os principais critérios para a definição de uma compra. As diversas possibilidades que o mercado eletrônico oferece tornam essa tarefa cansativa e demorada, o que desmotiva os usuários.

Devido à dificuldade em definir um local que ofereça melhor preço e maior qualidade de serviço, usuários costumam buscar informações que auxiliem a tomada dessa decisão. Algumas possibilidades são: pesquisas em portais de comparação de preço, sites de *reviews* e redes sociais, onde na maioria das vezes, essas informações são cedidas através da experiência de consumo de outros usuários.

Com base nesse cenário, propõe-se a construção de uma aplicação no formato de uma rede social, que incentiva o usuário a publicar dados pessoais referentes a consumo, aliada a tecnologia de *data warehouse*.

Devido a sua flexibilidade e fácil entendimento, a modelagem multidimensional própria dos *data warehouses* permite agregar dados de forma eficaz, de modo que diversas análises possam ser realizadas [Raghu Ramakrishnan 2003]. No caso em questão, os dados de diferentes usuários a respeito de suas compras podem ser agrupados para a geração de análises estatísticas úteis para os usuários, como gráficos de comparação de preço de produtos.

Neste artigo será apresentada a arquitetura de *data warehouse* proposta. Trata-se de uma arquitetura genérica, que possibilita a coleta de dados de ofertas de produtos e a posterior análise dessas ofertas. No entanto, neste artigo será demonstrada uma aplicação específica, em que a arquitetura serviu para o armazenamento e análise de informações referentes a compras realizadas pelos usuários da rede social.

Dois módulos específicos foram desenvolvidos. Um deles realiza a coleta de dados interativamente, a partir de dados de consumo fornecidos pelos próprios usuários da rede social. O segundo módulo utiliza o algoritmo de classificação não supervisionado *k-means* para agrupar usuários em comunidades que possuam perfis de consumo semelhante. Esse tipo de agrupamento permite aproximar usuários que adquiriram produtos semelhantes, o que pode ser útil na tomada de decisão referente a compra de determinado produto.

Este artigo está dividido da seguinte forma: na seção 2 é apresentada a arquitetura de *data warehouse* proposta, o que inclui a modelagem multidimensional utilizada, o módulo responsável pela carga de dados e o módulo que implementa a técnica *K-means*. Na seção 3 são discutidos experimentos a respeito do uso do *K-means* para o agrupamento de usuários em um número fixo de comunidades. Os trabalhos relacionados são descritos na seção 4. Para finalizar, a seção 5 traz as considerações finais.

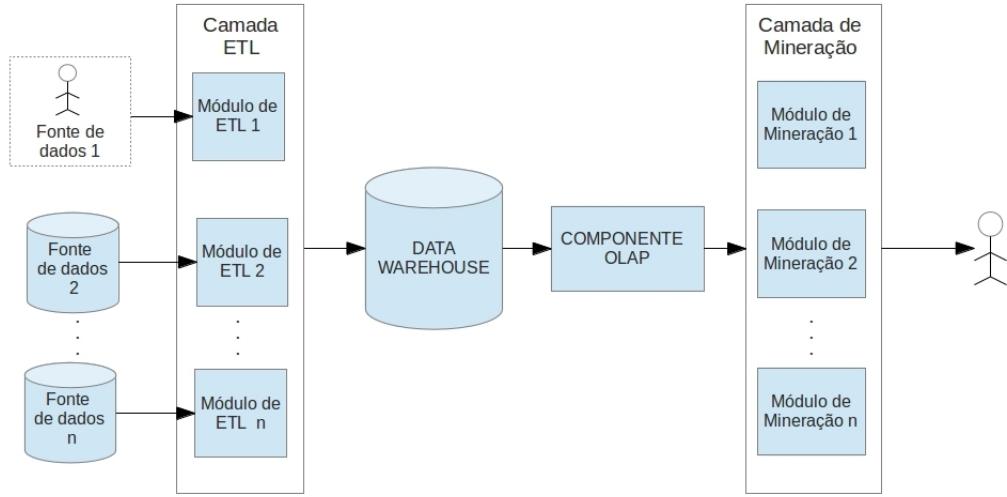
## 2. Arquitetura da Aplicação

A Figura 1 apresenta a arquitetura proposta para esse trabalho. De modo geral, as ofertas de produtos são extraídos das fontes de dados e alimentados em um *data warehouse*, que decompõe os dados em dimensões que caracterizam cada oferta. A partir dos dados, é possível realizar operações OLAP para gerar gráficos mostrando correlações entre os dados, ou para análises de dados mais sofisticadas através de técnicas de mineração de dados.

Nas próximas seções, os componentes da arquitetura são apresentados de forma detalhada. Juntamente com a explicação dos componentes da arquitetura, serão apresentados os módulos específicos que foram implementados tendo em vista uma aplicação de rede social que encapsula o acesso aos dados armazenados.

### 2.1. Fontes de Dados

De modo geral, os dados de interesse são aqueles relacionados a ofertas de produtos, o que inclui informações como categoria e preço, por exemplo. Esses dados encontram-se distribuídos em diversas fontes externas, como bases de dados operacionais, planilhas



**Figura 1. Arquitetura de *Data Warehouse* usada para Coleta e Análise de Dados de Ofertas de Produtos**

eletônicas, documentos XML, formulários Web e sites de *e-commerce*, dentre outros. Neste artigo, destaca-se a possibilidade de recuperar dados de interesse diretamente a partir dos próprios usuários da rede social, conforme ilustrado no retângulo tracejado da Figura 1.

## 2.2. Camada ETL (Extract, Transform and Load )

A carga das fontes de dados para o *data warehouse* é realizada através da camada de ETL. Essa camada é composta por módulos de ETL, sendo que cada módulo se especializa na carga de dados de um tipo específico de fonte de dados.

De modo geral, um módulo de ETL é responsável pelas operações de extração, limpeza e carga. A extração e a carga se encarregam de selecionar os dados da fonte de dados e armazená-la no *data warehouse*, respectivamente. Já a transformação se encarrega do processo de adequação dos dados para que esses possam ser aproveitados. Essa etapa é necessária uma vez que as fontes podem apresentar erros, anomalias e inconsistências. Para solucionar esses problemas, a etapa de transformação utiliza técnicas de limpeza de dados (*data cleaning*), que tem como objetivo detectar e remover anomalias dos dados; e Desambiguidade de dados (*Data Disambiguation*), que busca categorizar corretamente esses dados. Através dessas técnicas é possível alcançar unicidade de informação e obter melhores benefícios de suporte à decisão.

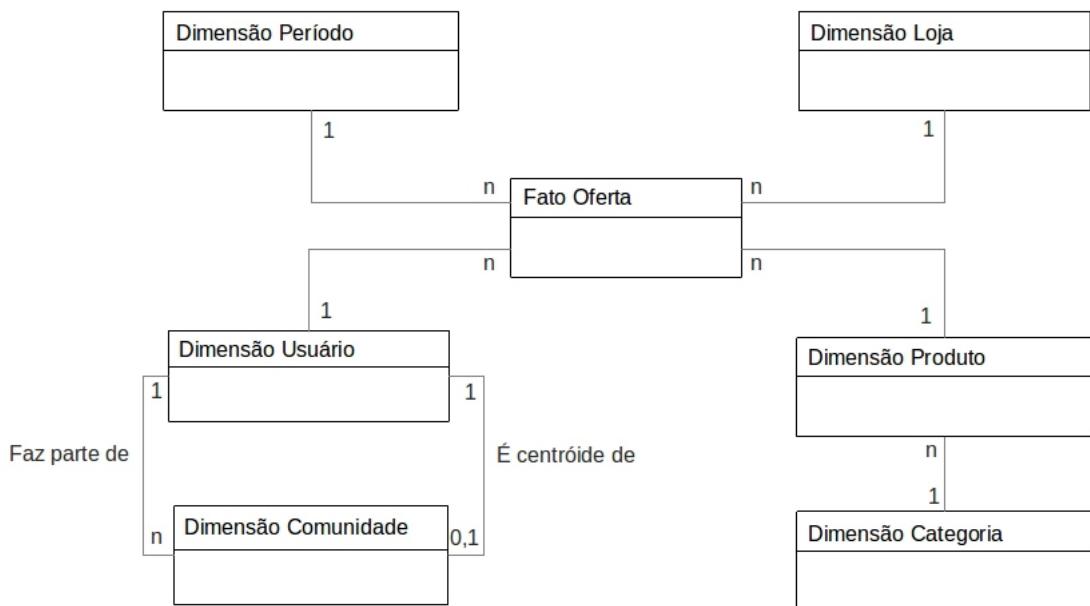
Neste artigo foi projetado um tipo específico de módulo de ETL que recebe dados diretamente de usuários, e não de uma fonte de dados concreta já existente. De acordo com esta abordagem, usuários fornecem suas informações de consumo através de um formulário de registro de compra de produtos, disponível no ambiente de uma rede social.

Para preencher o formulário, os seguintes campos são fornecidos: Produto comprado, Categoria do produto comprado, Período da compra, Loja onde foi feita a compra e Preço do produto comprado. O formulário é disponibilizado como um recurso de um organizador financeiro. Depois de registrar a compra, o usuário pode pesquisar em um calendário o histórico de compras realizadas.

### 2.3. Data Warehouse

O esquema de *data warehouse* utilizado neste trabalho baseia-se no modelo floco de neve, onde as tabelas dimensões principais se relacionam diretamente com a tabela fato, e algumas tabelas dimensão extras se relacionam com as dimensões principais. Essa estrutura tem o propósito de normalizar as tabelas dimensionais, visando reduzir o espaço ocupado por essas tabelas e criar hierarquias entre as mesmas [Machado 2010].

A figura 2 ilustra a modelagem de dados proposta para esse trabalho. O modelo contém uma entidade central (fato), chamada oferta, que armazena os registros de ofertas de consumo fornecidos pelos usuários. A tabela oferta contém o valor unitário de cada produto, e identificadores para as entidades dimensões.



**Figura 2. Modelo Multidimensional Relacionado a Dados de Ofertas de Produtos**

Nas tabelas dimensões são armazenados dados descritivos relacionados às ofertas de produtos, que são o produto ofertado, local onde foi feita a oferta, período da oferta e usuário que adquiriu o produto ofertado. Observe que a dimensão usuário só será preenchida nos casos em que a oferta for na verdade um item já adquirido, sendo que o usuário nesse caso é a pessoa que realizou a compra.

Para fins de normalização, as dimensões categoria e comunidade se relacionam respectivamente com as dimensões produto e usuário. A primeira identifica a categoria do produto ofertado, enquanto a segunda identifica a comunidade a que o usuário pertence. Conforme indicado na figura, existe um segundo relacionamento entre usuário e comunidade, que identifica quem é o usuário central da comunidade. O preenchimento dos dados relacionados a comunidades é visto na seção 2.5.

### 2.4. Componente OLAP

O componente de OLAP (*On-line Analytical Processing*) é responsável pelo acesso aos dados multidimensionais armazenados no *data warehouse*. Diferentemente do modelo

relacional puro, em que as consultas são especificadas através de operadores de seleção, projeção, junção e agrupamento, no modelo multidimensional, onde os dados são dispostos em estruturas dimensionais chamadas de cubos, novos operadores podem ser utilizados. Algumas das operações OLAP mais comuns são *Drill Down*, para diminuir o nível de granularidade de uma dimensão, *Roll up*, para aumentar o nível de granularidade de uma dimensão e *Slice and Dice*, para fatiar o cubo de dados através da seleção em uma das dimensões.

Através das operações OLAP, é possível realizar uma série de análises sobre os dados de consumo fornecidos pelos usuários. Por exemplo, os dados podem ser utilizados para a geração de gráficos de tendência de preço, exibindo as correlações existentes entre locais de venda e período das ofertas. Neste artigo, as operações OLAP são usadas para recuperar os dados que serão utilizados na classificação das comunidades, conforme descrito na próxima seção.

## 2.5. Camada de Mineração de Dados

A camada de mineração de dados utiliza os dados sumarizados armazenados no *data warehouse* para descobrir padrões e correlações entre os fatos automaticamente. Uma das formas de realizar essa análise envolve o uso de algoritmos de aprendizado de máquina, que buscam classificar os fatos em grupos que partilhem de alguma característica em comum.

Neste trabalho, foi implementado um módulo de mineração que divide os usuários em comunidades, agrupando-os de acordo com o seu perfil de consumo. Para isso, foi utilizado o *K-means*, um algoritmo não supervisionado de aprendizado de máquina.

Dado um número de comunidades, o algoritmo mapeia os usuários em um espaço multidimensional e executa um número limitado de iterações. Em cada iteração são escolhidos os usuários centróides de cada comunidade. Os demais usuários são agrupados na comunidade onde a distância euclidiana entre ele e o centróide for menor.

Na primeira iteração, os centróides são escolhidos aleatoriamente. Nas iterações seguintes, o centróide de uma comunidade será o usuário central da comunidade calculada na iteração anterior. O agrupamento termina após um número pré-definido de iterações ou depois que a classificação convergir, ou seja, quando os agrupamentos não modificarem entre uma iteração e a seguinte.

Para o cálculo da distância euclidiana, os usuários são mapeados em um espaço multidimensional composto pelas categorias de produtos existentes no *data warehouse*. Para cada usuário, o vetor de categorias  $\{c_1, c_2, \dots, c_n\}$  é preenchido de acordo com as compras feitas pelo usuário. Ou seja, o valor de  $c_i$  é definido como sendo a soma de todos os produtos comprados que sejam da categoria  $c_i$ . A seleção dos fatos, categorias e usuários é feita através dos operadores de OLAP mencionados na seção anterior.

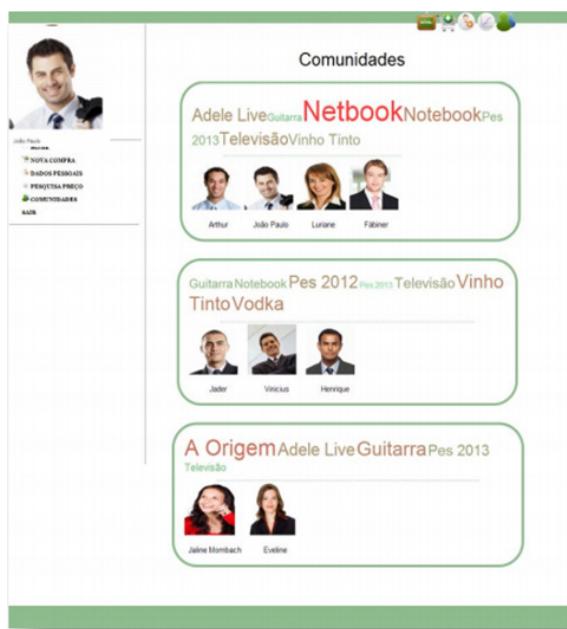
Após o término da execução do algoritmo, o *data warehouse* é atualizado com a atribuição da comunidade de cada usuário, assim como com a atribuição do usuário centróide de cada comunidade, conforme os dois relacionamentos entre as dimensões usuário e comunidade, indicados na Figura 2.

## 2.6. Visualização das Comunidades

O resultado da divisão em comunidades pode ser visualizado através de um recurso fornecido pela rede social. Ao acessar a visualização, um usuário obtém uma lista ordenada de comunidades, sendo que a ordem é relativa a distância euclidiana entre o usuário e os centróides das comunidades. A primeira comunidade será aquela a que o usuário efetivamente pertence. As demais comunidades são aquelas que possuem graus decrescentes de relevância com o usuário. Essa forma de exibição leva em consideração que o perfil de um usuário pode se espalhar por diversos grupos de interesse, e que é interessante que ele tenha acesso à pessoas que pertençam a todas as comunidades definidas.

Como a classificação é não supervisionada, os usuários são divididos em grupos que partilhem características em comum. No entanto, ainda é necessário identificar cada uma dessas comunidades com uma descrição breve e precisa. A forma encontrada de realizar essa tarefa foi através de um gerador de assinaturas. Esse gerador cria uma *tag cloud* (nuvem etiquetada) para as comunidades, contendo o nome de produtos. Os produtos que foram mais comprados pelos membros das comunidades estarão com o tamanho de fonte maior e cor diferenciada, permitindo que o usuário identifique o propósito de cada comunidade.

Através da Figura 3, pode-se visualizar como as comunidades serão apresentadas para usuário. Cada membro de uma comunidade será representado através da imagem do seu avatar, seguido do seu nome definido no cadastro da rede social. É possível ver que o usuário está mais fortemente relacionado a comunidade em que o produto *Netbook* é mais largamente vendido.



**Figura 3. Comunidades geradas com base no perfil de consumo dos usuários.**

## 3. Experimentos

Esta seção apresenta os resultados da avaliação do nível de precisão para a classificação de usuários com base no seu perfil de consumo, utilizando o algoritmo de aprendizado de

máquina *K-means*. Neste trabalho, apresentaremos os resultados obtidos utilizando um número de centróides fixo equivalente a três. Ou seja, os usuários serão agrupados em três comunidades distintas ao final da etapa de classificação.

Para realizar o experimento foram armazenados no *data warehouse* registros fictícios de compras. De modo a comparar o resultado da classificação de forma objetiva, foi usado um cenário de teste controlado, de acordo com os critérios definidos abaixo:

- Foi gerado um conjunto de usuários  $U_1 \leftarrow \{u_1, u_2, u_3, u_4\}$ .
- Foi gerado um conjunto de usuários  $U_2 \leftarrow \{u_5, u_6, u_7, u_8\}$ .
- Foi gerado um conjunto de usuários  $U_3 \leftarrow \{u_9, \dots, u_{20}\}$ .
- Foi gerado um conjunto de categorias  $C_1 \leftarrow \{c_1, c_2, c_3, c_4\}$ .
- Foi gerado um conjunto de categorias  $C_2 \leftarrow \{c_5, c_6, c_7, c_8\}$ .
- Foi gerado um conjunto de categorias  $C_3 \leftarrow \{c_9, \dots, c_{20}\}$ .
- Para cada usuário  $u_i \in U_1$ , foram gerados oito registros de compras, sendo duas compras para cada categoria de  $C_1$
- Para cada usuário  $u_i \in U_2$ , foram gerados oito registros de compras, sendo duas compras para cada categoria de  $C_2$
- Para cada usuário  $u_i \in U_3$ , foram gerados oito registros de compras da categoria  $c_i$

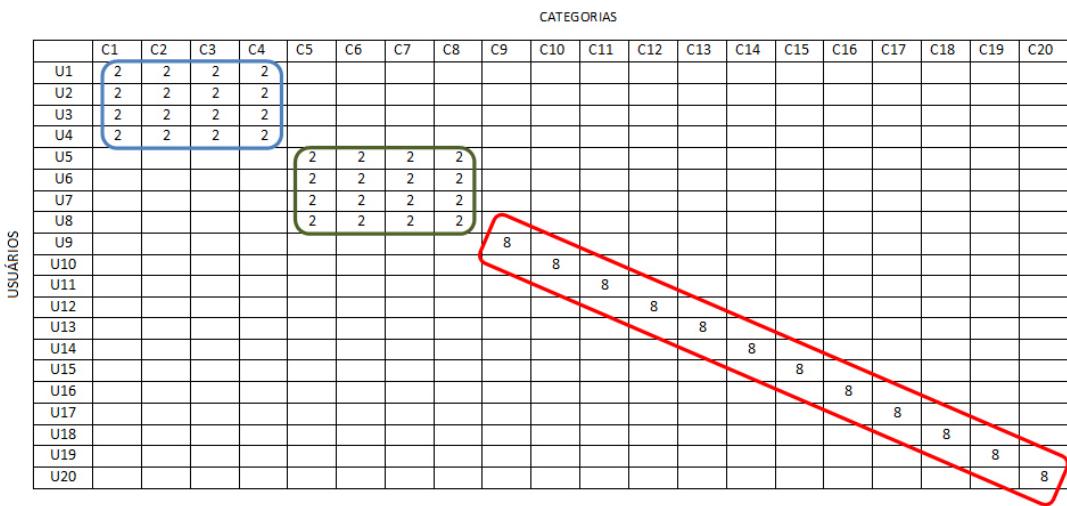
Os critérios acima foram escolhidos de modo que cada um dos conjuntos de usuários apresente comportamento distinto dos demais. Os usuários de  $U_1$  realizaram compras de produtos de quatro categorias, enquanto os usuários de  $U_2$  realizaram compras de produtos de outras quatro categorias. Já os usuários de  $U_3$  são aqueles que compram produtos de categorias que ninguém mais adquiriu, ou seja, o perfil de consumo de cada um desses usuários é distinto de todos os demais.

A Figura 4 apresenta a classificação que foi obtida para essa configuração. Como pode-se ver, o classificador reconheceu que os usuários dos conjuntos  $U_1$  e  $U_2$  possuem perfis de consumo semelhantes, e agrupou todos os demais usuários em um *cluster* a parte. Idealmente esses usuários não deveriam ser considerados próximos de ninguém, dado que seus perfis de consumo são divergentes. No entanto, o uso de apenas três centróides impede que isso aconteça.

Mesmo assim, é interessante ver que o classificador pelo menos não agrupou esses usuários com aqueles que possuem perfil bem definido. Dado que o número de centróides é fixo, pode-se considerar esse resultado satisfatório, uma vez que o terceiro grupo pode ser interpretado como O resultado gerado pode ser interpretado como o conjunto de pessoas que compram produtos de apenas uma categoria.

## 4. Trabalhos Relacionados

No contexto deste trabalho, técnicas de classificação aplicadas sobre dados de produtos e baseadas no algoritmo *K-means* já vem sendo aplicadas para problemas de segmentação de mercado. Em alguns casos, o algoritmo é usado em conjunto com outras técnicas, como mapas auto organizáveis [Kuo et al. 2002] e acasalamento de abelhas (*Honey Bee Mating*) [Fathian and Amiri 2008]. No problema de segmentação de mercado, a partir da análise do comportamento, os clientes são separados em segmentos, que são posteriormente marcados como rentáveis ou não. Nesse caso, o objetivo da classificação é basicamente direcionar campanhas de marketing para segmentos mais rentáveis. Já o nosso



**Figura 4. Análise da Classificação Realizada pelo *k-Means***

trabalho tem a própria aproximação das pessoas como objetivo. Essa distinção faz com que as técnicas partam de informações diferentes para realizar a classificação, como o dia da compra, o número de compras e o valor dos produtos adquiridos. Consequentemente, a própria concepção de que pessoas devem compor uma comunidade tende a diferir.

O trabalho em questão também pode ser analisado como um problema de recomendação, onde o objetivo é indicar pessoas com quem indivíduos específicos gostariam de se relacionar. A literatura está repleta de estudos relacionados a problemas de recomendação baseado no histórico de ações do usuário. Essa área recebe o nome de Filtragem Colaborativa (*Collaborative Filtering*). Em boa parte dos casos, o objetivo envolve a recomendação de produtos, como no site de compras da Amazon [Linden et al. 2003]. A partir das preferências conhecidas de alguns usuários, pretende-se fazer recomendações com base em preferências desconhecidas de outros usuários.

Algumas técnicas de recomendação, chamadas de técnicas baseadas em modelos, usam algoritmos de classificação como o *K-means* para realizar a predição de produtos [Su and Khoshgoftaar 2009]. No entanto, conforme descrito em [Linden et al. 2003], o processo de recomendação *online*(em tempo real) se torna oneroso quando embasada em algoritmos de aprendizado de máquina.

Também existem trabalhos de Filtragem Colaborativa que visam a recomendação de pessoas. Para esses casos, normalmente a recomendação pode ser feita em modo *offline*, o que permite o uso de técnicas menos responsivas, como as baseadas em classificação. As abordagens propostas variam de acordo com o objetivo específico que se pretende atingir. Por exemplo, em [Ungar et al. 1998], pessoas são agrupadas com base em seus interesses por filmes. Como a informação de gosto é esparsa, foi proposta uma forma de classificação que ocorre em duas frentes. Numa delas são agrupados filmes que são preferidos pelo mesmo grupo de pessoas, e na outra são agrupadas pessoas que preferem os mesmos filmes.

Existem também trabalhos de recomendação que atuam diretamente sobre re-

des sociais. Em [Hannon et al. 2010], são gerados relacionamentos entre usuários do *Twitter* com base na análise do conteúdos dos seus Tweets. Já em [Cai et al. 2010], a recomendação de pessoas usa a rede de relacionamentos existentes, explorando o conceito de gosto (de quem uma pessoa gosta) e atratividade (quem gosta de uma pessoa).

Convém destacar que o uso de *data warehouse* com dados coletados a partir da Web já foi assunto de investigação na literatura. Em [Hernández et al. 2011], esse modelo é chamado de *data webhouse*. Conforme descrito no artigo, os dados coletados são oriundos da navegação do usuário dentro de um Web site corporativo, e o *data warehouse* gerado tem como objetivo principal a tomada de decisões estratégicas que visam aprimorar a experiência da navegação, de modo que o usuário encontre a informação que procura mais rapidamente.

O modelo que propomos neste artigo também pode ser definido como um *data webhouse*, uma vez que os dados podem ser coletados diretamente através da interação do usuário com a rede social. No entanto, os dados armazenados podem ser utilizados para um número maior de aplicações, como aquelas voltadas para a segmentação de mercado ou recomendação de pessoas/produtos, conforme descrito nos trabalhos acima apresentados.

## 5. Conclusão

Esse trabalho apresentou uma arquitetura de *data warehouse* para coleta e análise de ofertas de produtos. Para validar a arquitetura, foi desenvolvida uma aplicação que utiliza serviços de uma rede social tanto para a coleta como para a análise de dados. Especificamente, a rede social dá ênfase a dados de consumo registrados pelos seus usuários.

Para demonstrar as análises que os dados armazenados possibilitam, o algoritmo *K-means* foi empregado para realizar a separação dos usuários em comunidades, levando em consideração seu histórico de compras registradas na rede social. Essa separação pode ser usada de diversas formas, como por usuários interessados em produtos específicos e por empresas interessadas em campanhas publicitárias focadas. Por exemplo, compras coletivas podem partir tanto do conjunto de pessoas interessadas em um produto quanto por uma empresa que perceba esse interesse nas comunidades que monitora.

A aplicação de compra coletiva é apenas um exemplo que pode ser explorado. O que vale a pena destacar é que, para que aplicações como essas sejam possíveis, é necessário ter os dados concentrados de uma forma que facilite a análise posterior. Nesse sentido, o artigo propõe uma abordagem prática ao empregar redes sociais tanto para alimentar o *data warehouse* quanto para consumir os dados após análise.

Como trabalhos futuros, pretende-se incorporar novos módulos de ETL à arquitetura, de modo que mais informações de ofertas estejam disponíveis. Os módulos podem ser ativos, acessando fontes de dados na Web em busca das informações, ou passivos, sendo chamados por agentes externos que realizam a carga. Essa opção é interessante para lojas que desejem ter seus produtos a disposição na aplicação.

Além disso, pretende-se aprimorar o módulo de ETL existente, para que os dados dos usuários sejam consolidados antes de alimentarem o *data warehouse*. Um dos métodos a ser estudado envolve usar técnicas de casamento de string para corrigir o nome de um produto, caso outro nome semelhante seja encontrado no *data warehouse*.

Outra possibilidade de trabalho futuro envolve modificar o algoritmo *K-means* utilizado de modo que o número de comunidades seja dinâmico. Conforme apresentado nos experimentos, o uso de um número fixo de comunidades pode agrupar usuários que não possuem nenhuma relação de consumo aparente. Também deseja-se estudar as dimensões que compreendem o *data warehouse* afim de identificar novos tipos de análises que podem ser feitas sobre os dados, o que inclui estender o algoritmo de classificação de usuários usando outras dimensões além das categorias de produtos que foram adquiridos.

## Referências

- Cai, X., Bain, M., Krzywicki, A., Wobcke, W., Kim, Y. S., Compton, P., and Mahidadia, A. (2010). Learning collaborative filtering and its application to people to people recommendation in social networks. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 743–748, Washington, DC, USA. IEEE Computer Society.
- Fathian, M. and Amiri, B. (2008). A honeybee-mating approach for cluster analysis.
- Hannon, J., Bennett, M., and Smyth, B. (2010). Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 199–206, New York, NY, USA. ACM.
- Hernández, P., Glorio, O., Garrigós, I., and Mazón, J.-N. (2011). Towards a model-driven framework for web usage warehouse development. In *Proceedings of the 30th international conference on Advances in conceptual modeling: recent developments and new directions*, ER'11, pages 336–337, Berlin, Heidelberg. Springer-Verlag.
- Kuo, R. J., Ho, L. M., and Hu, C. M. (2002). Integration of self-organizing feature map and k-means algorithm for market segmentation. *Comput. Oper. Res.*, 29(11):1475–1493.
- Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80.
- Machado, F. N. R. (2010). *Tecnologia e Projeto Data Warehouse*. Sao Paulo-SP, 3 edition.
- Raghu Ramakrishnan, J. G. (2003). *Sistemas de Gerenciamento de Banco de Dados*. Sao Paulo-SP, 3 edition.
- Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2.
- Ungar, L., Foster, D., Andre, E., Wars, S., Wars, F. S., Wars, D. S., and Whispers, J. H. (1998). Clustering methods for collaborative filtering. AAAI Press.

# **Uma ferramenta para distribuição e mapeamento de dados Paleogeográficos**

**Joaquim Assunção, Maria Pivel, Paulo Fernandes, Duncan Ruiz**

<sup>1</sup>Pontifícia Universidade Católica do Rio Grande do Sul

Faculdade de Informática - FACIN

Porto Alegre, Brazil (+55)51-3320-3558

{joaquim.assuncao,maria.pivel,paulo.fernandes,duncan}@pucrs.br

**Resumo.** As bacias sedimentares marginais despertam grande interesse, tanto de cientistas da terra, como da indústria petrolífera. Pesquisas nestas áreas geram grandes volumes de dados provenientes de técnicas de geofísica e perfurações. Esses dados se encontram, muitas vezes, esparsos e armazenados de formas gráficas e esquemáticas, o que dificulta a aplicação de técnicas computacionais para análise e descoberta de conhecimento. Além disso, não há informação suficiente para toda a área de interesse. Pensando nisso, criou-se uma ferramenta para coleta, distribuição e mapeamento destes dados. Com isso, criou-se novas oportunidades, pois com o auxílio de algoritmos de mapeamento, novos dados foram estimados de modo a preencher uma área da qual anteriormente não havia informação geológica. Os algoritmos desenvolvidos compõem a ferramenta que serve para extração, transformação e carga de dados (ETL). Estes dados compõem um banco de dados paleogeográficos de grande volume. Esse banco visa agregar informações existentes com informações estimadas para se obter novas informações via processos de descoberta de conhecimento em banco de dados (*Knowledge Discovery in Database, KDD*).

## **1. Introdução**

Ao longo de décadas de pesquisa, cientistas das geociências acumularam quantidades substanciais de dados geológicos. Esses grandes volumes de dados têm potencial para possuírem informações ocultas, úteis para as geociências. Porém, quanto maior a quantidade e a diversidade destes dados, maior a dificuldade de se extrair informação dos mesmos [Miller e Han 2001] [Fayyad et al. 1996].

Dados geológicos que representam longos períodos de tempo, usualmente são gerados e acumulados de maneiras distintas, tanto na representação como na forma e local de armazenamento. De fato, muitos são os meios utilizados para armazenamento destas informações. Dentre eles destacam-se os gráficos gerados por levantamentos sísmicos, onde grande parte das informações obtidas, são unidas e sintetizadas em cartas estratigráficas, que possuem foco na ocorrência de litologias em um local e tempo [Milani et al. 2007].

O trabalho descrito neste artigo tem como principal objetivo obter o máximo proveito dos dados presentes nas cartas estratigráficas, unindo-os com outros dados paleogeográficos. Para isso, foram feitos estudos sobre os geodados em questão, além de estudos

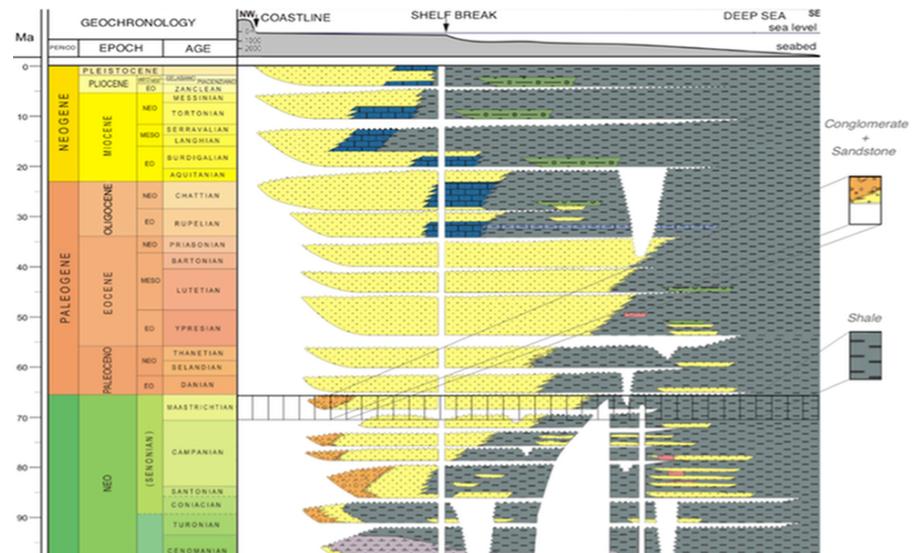
de técnicas para adaptação e mapeamento de dados paleogeográficos. Para atingir este objetivo, este trabalho foi dividido em três etapas: (1) coleta e adaptação dos dados, (2) transformação e estimativa de novos dados, (3) carga e mapeamento dos dados.

O produto final, resultante das três fases, constitui um conjunto de dados no espaço e no tempo. Para isso, na primeira fase, os dados são extraídos das cartas e das demais fontes de interesse, e ambos são adaptados para terem formatos compatíveis entre si. Na segunda fase, os dados são transformados para um padrão numérico, de forma a facilitar sua manipulação e servir como entrada para um algoritmo de estimativa e mapeamento, que na terceira fase realiza o mapeamento dos dados e os insere no banco de dados.

A abordagem descrita acima foi aplicada a dados das bacias sedimentares marginais brasileiras. A seção 2 constitui uma breve descrição do conhecimento necessário para o entendimento da solução. A seção 3 relata a solução criada e os resultados obtidos. Na seção 4 são feitas considerações sobre o que foi criado, a contribuição e os trabalhos futuros.

## 2. Background e dados alvo

Cartas estratigráficas são ferramentas úteis para o estudo de bacias sedimentares. Elas representam graficamente as mudanças geológicas que ocorreram em uma bacia em função do tempo. Além disso, carregam grandes quantidades de informações relativas as litologias presentes naquela bacia [Milani et al. 2007]. A figura 1 mostra um exemplo de carta estratigráfica; neste caso, a carta da Bacia de Santos.



**Figura 1. Simplificação da Carta estratigráfica da Bacia de Santos.** À esquerda, a barra colorida representa a escala de tempo geológico, do Cretáceo até o Recentes (amarelo). À direita, diferentes padrões e cores são usados para representar as litologias que são distribuídas de acordo com a distância da costa (eixo horizontal) e idade geológica (eixo vertical). Espaços em branco representam ausência de depósitos de litologias para uma determinada distância da costa, em uma determinada idade geológica. O grid mostrado sobre a idade Maastrichtiana ilustra uma das áreas de coleta de dados.

Todos os dados extraídos das cartas estratigráficas são divididos por idades geológicas. Cada carta representa um limite entre a linha da costa brasileira até a isóbata

de 3000m. Isto significa que a largura da Bacia é variável. Todavia, o limite marítimo brasileiro é restrito a 200 milhas náuticas; como grande parte das bacias está contida neste limite, cada Idade da carta foi dividida em 37 partes, onde cada parte representa aproximadamente 10km.

Com o objetivo de caracterizar a configuração presente da costa brasileira, além dos dados estratigráficos, a solução desenvolvida também agrega ao banco, dados de batimetria (*i.e.* profundidade do mar) [Smith e Sandwell 1997] e anomalias gravimétricas (*i.e.* desvios do valor teórico da aceleração da gravidade que auxiliam na compreensão da estrutura interna das bacias) [Sandwell e Smith 2009].

### 3. Solução

Para cada tipo de litologia foi atribuído uma representação por potência de dois. Deste modo, várias litologias podem ser representadas com um único valor. Esta abordagem também é prática para a decomposição de valores e, por consequência, obtenção das litologias presentes no local.

Para estimar o valor de um ponto  $x$  no espaço, foi criado um algoritmo que se baseia na distância entre o ponto em questão e os pontos mais próximos de cada limite da bacia em questão. Deste modo o algoritmo cria uma matriz de valores que são mapeados em pontos no espaço. Para isso, foi utilizada a fórmula de Haversine, juntamente com uma variação da mesma para obtenção da curvatura entre dois pontos.

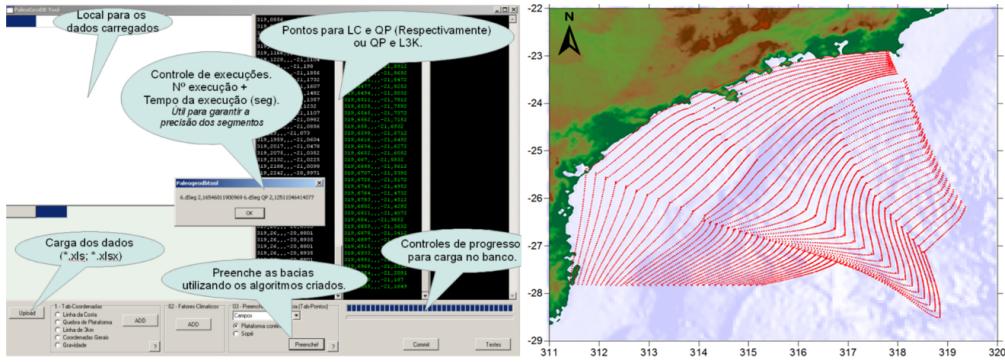
$$d = R.c \quad (1)$$

Onde :

$$\begin{aligned} R &= \text{Raio da Terra} = 6.371\text{km} \\ c &= 2.\text{atan2}(\sqrt{a}, \sqrt{1-a}) \\ a &= \sin^2(\Delta\text{lat}/2) + \cos(\text{lat1}).\cos(\text{lat2}).\sin^2(\Delta\text{lon}/2) \\ \Delta\text{lat} &= \text{lat2} - \text{lat1} \\ \Delta\text{lon} &= \text{lon2} - \text{lon1} \end{aligned}$$

$$\begin{aligned} \Theta &= \text{atan2}(\sin(\Delta\text{lon}).\cos(\text{lat2}), \\ &\quad \cos(\text{lat1}).\sin(\text{lat2}) - \sin(\text{lat1}).\cos(\text{lat2}).\cos(\Delta\text{lon})) \end{aligned} \quad (2)$$

Após a criação do algoritmo, foi desenvolvida uma ferramenta para executá-lo. Mais do que isso, a ferramenta realiza alterações de formatos, importa planilhas Excel, realiza cálculos e aplica os algoritmos criados para estimativas e carrega os dados para o banco. À esquerda na figura 2, é mostrada a interface da ferramenta, juntamente com um breve resumo das suas funções. À direita é mostrado o resultado do mapeamento de dados.



**Figura 2. Ferramenta de ETL, juntamente com o resultado do mapeamento para a bacia de Santos. Cada ponto representa a localização de diferentes dados inseridos no banco e mapeados ao longo da margem continental.**

#### 4. Conclusão

Neste artigo foi relatada a criação de uma ferramenta que incorpora novos algoritmos para distribuição e mapeamento de dados paleogeográficos. Com esta ferramenta é possível atribuir valores para diferentes proporções de depósitos de sedimentos para toda a área de interesse da costa brasileira.

A ferramenta também possibilita a criação de diversos dados que são parte de um grande banco de dados paleogeográficos. Além disso, a ferramenta também pode ser adaptada para mapear outros dados semelhantes em qualquer superfície esférica.

Nossos testes quanto à posição geográfica se mostraram satisfatórios (ver: parte à direita da figura 2), além disso, não foi identificado nenhum conjunto de técnicas na literatura com propósito semelhante.

As informações que os dados gerados representam estão sendo comparadas com dados de levantamentos sísmicos. Para isso, são usadas técnicas de mineração de dados, que não apenas servem para verificação dos dados mapeados, mas principalmente como parte do processo de KDD. Assim, conclui-se que os algoritmos criados, juntamente com a ferramenta, geram e possibilitam a utilização de novos dados e técnicas computacionais para descobrir informações e gerar conhecimento.

#### Referências

- Fayyad, U., Piatetsky-Shapiro, G., e Smyth, P. (1996). Knowledge discovery and data mining: Towards a unifying framework. pages 82–88.
- Milani, E., Rangel, D., Bueno, G., Stica, J., Winter, W., Caixeta, J. e Neto, O. (2007). *Boletim de Geociências da Petrobras 3<sup>a</sup> Ed.*, volume 2. Sól Gráfica.
- Miller, H. e Han, J. (2001). *Geographic Data Mining and Knowledge Discovery*. Research Monographs in Geographic Information Systems Series. Taylor.
- Sandwell, D. e Smith, W. (2009). Global marine gravity from retracked Geosat and ERS-1 altimetry: Ridge Segmentation versus spreading rate. *Journal of Geophysical Research*, 114:18.
- Smith, W. e Sandwell, D. (1997). Global seafloor topography from satellite altimetry and ship depth soundings. *Science Magazine*, 277.

# **Uma Ferramenta MDA para Modelagem de Banco de Dados Relacionais**

**André S. Rosa<sup>1</sup>, Carlos Eduardo Pantoja<sup>1</sup>**

<sup>1</sup>CEFET/RJ - UnED Nova Friburgo, Rio de Janeiro, Brasil

[andre\\_souza.rosa@hotmail.com](mailto:andre_souza.rosa@hotmail.com), [pantoja@cefet-rj.br](mailto:pantoja@cefet-rj.br)

**Abstract.** This paper proposes a Tool for relational databases modeling that uses the Model-Driven Architecture (MDA) approach. The generic meta-model used in this approach enables the designer to choose among different languages and notations. Moreover, it allows the SQL automatic code generation. The tool for the Entity-Relationship notation will be constructed using the Graphical Modeling Framework (GMF). A simple work in progress example is presented the tool's main purpose.

**Resumo** Este artigo propõe uma ferramenta para modelagem de bancos de dados relacionais que usa a abordagem Model-Driven Architecture (MDA). O meta-modelo utilizado na abordagem permite ao projetista escolher entre diferentes linguagens e notações. Além disso, permite a geração do código na linguagem SQL de forma automática. A ferramenta para a notação Entidade-Relacionamento (ER) vai ser construída usando o Graphical Modeling Framework (GMF). Um simples exemplo do andamento do trabalho é apresentado para garantir a finalidade principal da ferramenta.

## **1. Introdução**

A modelagem conceitual é uma descrição concisa dos requisitos de dados do usuário, que inclui detalhes específicos do banco de dados e é usada para garantir que tais requisitos sejam atendidos e que não estejam em conflito entre si [Elmasri et al., 2005]. Existem diversos modelos para modelagem conceitual, voltados para banco de dados relacionais como o Modelo Entidade-Relacionamento (ER) [Chen, 1976], o *Crow'sFoot* [Simsion, 2007] e o diagrama de classes da UML, que apesar de ser um recurso da linguagem de modelagem para sistemas orientados a objetos também pode ser utilizado para modelagem de banco de dados relacionais.

Algumas ferramentas auxiliam o projetista na modelagem conceitual automatizando o projeto de banco de dados, como a *Xcase* [Xcase, 2013], que possui suporte para engenharia reversa e abrange grande parte dos Sistemas Gerenciadores de Banco de Dados (SGBD); o *ER/Studio* [ER/Studio, 2013], que também possui suporte a engenharia reversa e recursos para manutenção da base de dados; e o *brModelo* [Cândido, 2004], que é uma ferramenta *freeware* voltada para o ensino, usa a notação Extended Entity-Relationship (EER), e ainda, faz a conversão do modelo conceitual para o modelo lógico e geração de código para o modelo físico.

Porém, as ferramentas apresentam algumas limitações, como o caso da *Xcase*, que possui o atrelamento a uma notação específica, a *Crow'sFoot*, além de ser uma ferramenta privada. O *ER/Studio* possui suporte a duas notações, a *Crow'sFoot* e a IDEF1X, e a modelagem pode ser direcionada a um grande número de SGBD, porém é

uma ferramenta privada. O brModelo, apesar de ser um *freeware*, tem como foco apenas a notação EER.

Este artigo tem como objetivo propor uma ferramenta, baseada na Arquitetura Orientada por Modelos (*Model-Driven Architecture* - MDA), para modelagem conceitual de banco de dados relacionais. A MDA é uma abordagem de desenvolvimento de *software* dirigida por modelos em diversos níveis de abstração, onde um sistema é modelado usando um modelo independente de plataforma, que será transformado em um modelo específico de plataforma, dado um modelo de plataforma. A utilização de modelos direciona o entendimento, *design*, construção, teste, operação, manutenção e modificação do sistema [Mellor et al., 2005].

A ferramenta utilizará um meta-modelo genérico para as linguagens de modelagens relacionais e o gerador de código de núcleo comum com os padrões ANSI/SQL 93/99/03, proposto por [Rosa et al. 2013], integrado a um ambiente gráfico desenvolvido no *Graphical Modeling Framework* (GMF), permitindo ao projetista utilizar qualquer linguagem de modelagem existente. O GMF disponibiliza a infraestrutura necessária para o desenvolvimento de editores gráficos para modelos e foi construído baseado no *Eclipse Modeling Framework* (EMF) [Steinberg et al. 2008].

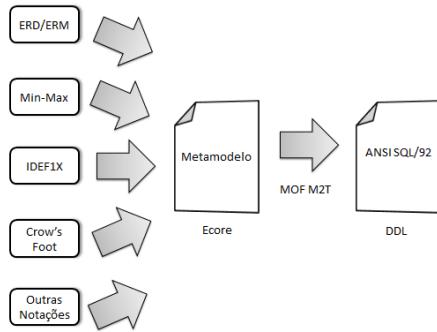
O artigo está estruturado da seguinte forma: na seção 2 será apresentada a ferramenta desenvolvida; na seção 3 serão apresentados os resultados obtidos; na seção 4 serão apresentados alguns trabalhos relacionados; e por fim, na seção 5 a conclusão.

## 2. A Ferramenta MDA

A ferramenta, que consiste em um conjunto de *plug-ins* integrados para o ambiente de desenvolvimento *Eclipse*, utiliza uma metodologia MDA para modelagem conceitual de banco de dados relacionais. A codificação automática ANSI SQL 92/99/03 é gerada a partir de um conjunto de regras de transformações na linguagem *Model To Text* (M2T).

A M2T [OMG, 2008] é uma linguagem utilizada para transformar instâncias de modelos em artefatos de texto a partir de regras de transformações estruturadas na forma de *templates*, onde estes são responsáveis por determinada transformação. A metodologia [Rosa et al., 2013] tem como núcleo comum o meta-modelo genérico para as linguagens de modelagens ER, IDEF1X, *Crow'sFoot* e UML; além de permitir a utilização da notação Min-Max nas instâncias de seus modelos. A metodologia ainda permite a integração de outras linguagens de modelagens, desde que essas sejam aderentes aos conceitos previstos no meta-modelo. A metodologia utilizada pela ferramenta pode ser vista na figura 1.

Para a construção da interface gráfica é necessário ter um modelo de domínio como base, que será o meta-modelo da metodologia adotada. O meta-modelo é representado pelo *Ecore Model (.ecore)*, que é o arquivo utilizado pelo EMF na construção de meta-modelos. Será criado o arquivo contendo o *Domain Generator Model (.gen)*, utilizado para geração do código Java referente a cada elemento presente no modelo de domínio. Também será gerado a partir desse arquivo um projeto de extensão *.edit*, que terá o papel de possibilitar a customização de alguma das partes que precisam ser definidas como imagens e ícones utilizados na representação de recursos. Esse projeto dará origem a parte dos *plug-ins* que conterão as configurações da ferramenta para utilização no *Eclipse*.



**Figura 1. Metodologia adotada [Rosa et al. 2013].**

Os arquivos contendo o *GMF Tool Model* (.gmftool) e o *GMF Graph Def Model* (.gmfgraph) serão criados onde o .gmftool conterá as definições e as customizações da paleta de recursos disponíveis para modelagem. Já o .gmfgraph irá conter a definição gráfica de cada elemento dos recursos para modelagem conceitual. Os arquivos .gen, .gmftool e gmfgraph serão submetidos a uma combinação através do GMF e darão origem ao *Mapping Model* (.gmfmap). O arquivo .gmfmap fará o mapeamento associando os elementos do modelo de domínio às classes; as classes aos elementos gráficos; e esses elementos a suas representações na paleta de recursos.

Será realizada uma transformação sobre o arquivo .gmfmap para gerar o *Diagram Editor Gen Model* (.gmfgen). Através do .gmfgen será possível gerar um projeto executável que conterá as classes necessárias para o funcionamento da interface gráfica e os *plug-ins* utilizados no *Eclipse*. A interface gráfica proposta dará suporte, inicialmente, para modelagens utilizando o modelo ER. O modelo foi escolhido por ser uma linguagem de modelagem amplamente utilizada.

### 3. Resultados Obtidos

Nesta seção são apresentados os resultados parciais obtidos no desenvolvimento da ferramenta gráfica. Na figura 2 há um simples exemplo da notação ER utilizando a interface gráfica desenvolvida, onde ilustra um caso de proprietários e seus respectivos veículos com a cardinalidade (1,N).



**Figura 2 - Exemplo de modelagem conceitual utilizando a ferramenta proposta.**

Os atributos e a cardinalidade não estão sendo graficamente representada nessa versão inicial. Portanto é necessário que estes sejam instanciados no modelo de domínio manualmente. Após a modelagem no ambiente gráfico, o gerador de código ANSI/SQL de Linguagem de Definição de Dados pode ser utilizado para gerar o código do modelo construído nos padrões 92/99/2003.

### 4. Trabalhos Relacionados

A ferramenta proposta com base na abordagem MDA nesse artigo, apesar de ainda estar com o ambiente gráfico em desenvolvimento, será construída tendo em seu núcleo um meta-modelo já existente que reúne características comuns às notações de modelagem

conceitual relacional. Por possuir em seu núcleo esse meta-modelo, a ferramenta estará livre do atrelamento a uma única notação, como é o caso do *Xcase*, que suporta apenas a notação de *Crow'sFoot*, do brModelo, que suporta apenas o Entidade-Relacionamento e do *ER/Studio*, que apesar de ser mais versátil as demais ferramentas, suporta apenas as notações de *Crow'sFoot* e IDEF1X.

## 5. Conclusão

O artigo apresentou uma ferramenta para modelagem conceitual de banco de dados relacionais que utiliza a arquitetura MDA e permite a utilização de diferentes linguagens de modelagem e notações de banco de dados para automatização do projeto de banco de dados. Foi apresentada a parte gráfica da ferramenta para o MER, com um simples exemplo de seu funcionamento, permitindo a geração de codificação automática ANSI SQL 92/99/03 a partir de seu núcleo comum utilizando um conjunto de regras M2T.

A ferramenta oferece flexibilidade ao projetista na escolha da linguagem de modelagem a ser utilizada para a modelagem conceitual do banco de dados, pois independente da linguagem escolhida para o projeto, a ferramenta irá gerar a codificação padrão ANSI SQL 92/99/03, que é compatível com a maioria dos SGBD. Como trabalhos futuros, serão desenvolvidas extensões gráficas e integrações com ferramentas já existentes para as linguagens de modelagem *Crow'sFoot*, IDEF1X e UML; além de permitir a utilização da notação Min-Max.

## Referências

- CÂNDIDO, C. H. (2004). brModelo: Ferramenta de Modelagem Conceitual de Banco de Dados. Dissertação, Centro Universitário UNIVAG.
- CHEN, P. P. (1976). The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, v. 1, n. 1, p. 9–36.
- ELMASRI, R., NAVATHE, S. B. (2005). *Sistemas de banco de dados*. Editora Pearson.
- ER/Studio (2013). <http://www.embarcadero.com/br/products/er-studio>.
- MELLOR, S. J., SCOTT, K., UHL, A. e WEISE, D. (2005). *MDA Destilada: Princípios de Arquitetura Orientada por Modelos*. Ciência Moderna Ltda.
- OMG (Objetc Management Group) (2008). MOFModel To Text Transformation Language (MOFM2T), 1.0. <http://www.omg.org/spec/MOFM2T/1.0>.
- ROSA, A., GONÇALVES, I. and PANTOJA, C. E. (2013). A MDA Approach for Database Modeling. *Lecture Notes on Software Engineering*, v. 1, n. 1, p. 26–30.
- SIMSION, G. (2007). *Data Modeling: Theory and Practice*. Technics Publications Llc.
- STEINBERG, D., BUDINSKY, F., MERKS, E. and PATERNOSTRO, M. (2008). *Emf: Eclipse Modeling Framework*. Pearson Education.
- Xcase Database Design Software (2013). <http://www.xcase.com/>.

# ***DBClassMapper: Uma Ferramenta de Apoio ao Mapeamento e Consultas para o ORM Gendal***

**Diego Magno da Silva, Ronaldo dos Santos Mello**

Depto. de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)  
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brasil.

{diegomagno, ronaldo}@inf.ufsc.br

**Resumo.** Este artigo apresenta a *DBClassMapper*, uma ferramenta de apoio ao mapeamento e consultas ao *ORM* (*Object-Relational Mapping*) *Gendal* cujo principal diferencial é gerar consultas a partir das classes de objetos e poder comparar estas classes com as tabelas relacionais do banco de dados, facilitando a atualização dos mapeamentos. Esta ferramenta foi criada para a *IDE Visual Studio 2010* da Microsoft, a qual utiliza o *.Net* como principal framework de desenvolvimento. A novidade da *DBClassMapper* é que o desenvolvedor pode gerar consultas complexas a partir de poucos cliques e com muito pouco conhecimento de *SQL*.

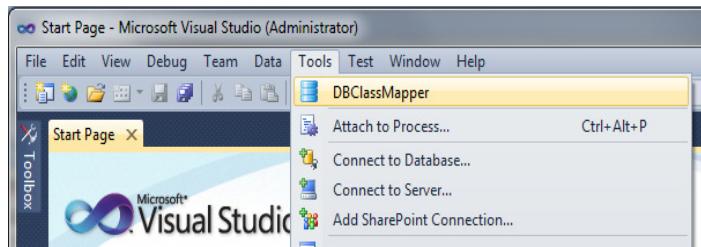
## **1. Introdução**

Em uma aplicação orientada a objetos que utiliza um banco de dados relacional para persistência de seus dados, uma ferramenta de *ORM* é importante para integrar os objetos da aplicação ao banco de dados. A utilização de um *ORM* aumenta a produção de uma equipe, pois reduz o problema da impedância (*impedance mismatch* [Ambler 2003]), uma vez que o programador da aplicação não precisa se preocupar com a construção de comandos na linguagem *SQL* para realizar a definição e manipulação de dados [ORM 2013]. Uma solução neste contexto é o *ORM Gendal* (*Generic Data Access Library*). *Gendal* é um *ORM* para o *framework .Net*, que é muito utilizado pela *IDE Visual Studio*, ambos da Microsoft. Os principais pontos fortes do *Gendal* são a simplicidade de definição de mapeamentos e de consultas complexas.

A *DBClassMapper* é uma ferramenta desenvolvida para o *Visual Studio 2010* com a finalidade de auxiliar a utilização do *Gendal*. Especificamente, *DBClassMapper* provê uma melhoria de usabilidade ao *Gendal*, pois facilita a manipulação da biblioteca ao permitir a construção interativa e simples de mapeamentos e de consultas ao banco de dados. A próxima seção detalha as funcionalidades desta ferramenta.

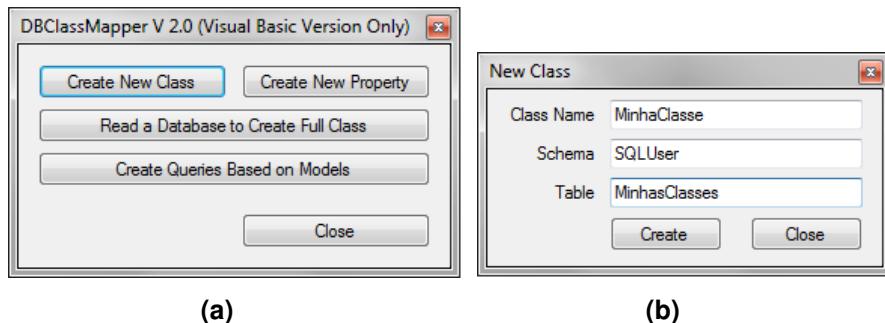
## **2. Ferramenta DBClassMapper: Utilização e Funcionalidades**

A ferramenta *DBClassMapper* foi desenvolvida para o *Visual Studio 2010* da Microsoft em *Visual Basic .Net 4.0* como um *AddIn* (denominação dada aos plug-ins no *Visual Studio*). A sua instalação é muito simples e rápida: basta executar o seu arquivo de instalação no *Windows* (XP ou 7). Quando o *Visual Studio* for aberto posteriormente, a ferramenta já irá aparecer no menu *Tools*, como mostra Figura 1.



**Figura 1.** *DBClassMapper* no menu *Tools* do Visual Studio 2010

A *DBClassMapper* possui várias facilidades integradas em uma única ferramenta. A sua tela de entrada (Figura 2 (a)), apresenta essas funcionalidades.

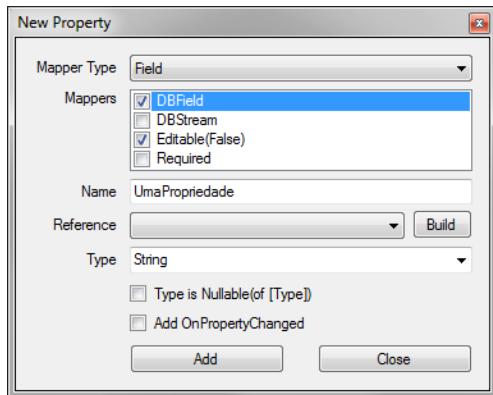


**Figura 2. a)** Tela principal da *DBClassMapper* **b)** Tela *Create New Class*.

A primeira funcionalidade (*Create New Class*), o primeiro botão (superior à esquerda), abre uma tela para auxiliar a criação de classes mapeadas individualmente. A segunda funcionalidade (*Create New Property*) abre uma tela para auxiliar a criação das propriedades mapeadas individualmente. A terceira funcionalidade (botão no centro) abre uma tela que permite comparar um esquema de banco de dados com as classes mapeadas do código fonte do projeto que está ativo no momento. A última funcionalidade (parte inferior) abre uma tela na qual é possível gerar consultas utilizando as classes já mapeadas. Cada uma destas funcionalidades é explicada a seguir.

A Figura 2 (b) mostra a tela para criar um novo mapeamento de classe. Nesta tela basta indicar o nome da classe, o nome da tabela e o esquema do banco de dados já existente que mantém a tabela.

A Figura 3 mostra a tela referente à funcionalidade *Create New Property* que define o mapeamento de uma propriedade de uma classe já existente. Nesta tela se indica o tipo de mapeamento (o qual pode ser *none*, ou seja, não será mapeada, ou tipos padrões como *key*, *alternate key* e *field*). Na sequência, é possível indicar propriedades gerais que um atributo em uma tabela pode ter (a lista varia de acordo com o tipo de mapeamento selecionado), bem como o nome da propriedade. O campo *Reference* indica uma referência a um tipo de objeto, sendo possível selecionar este tipo de objeto no campo *Type* para estabelecer um relacionamento. Por fim, informa-se o tipo de dado e os dois últimos *check boxes* definem, respectivamente, se a propriedade pode ser nula e o último serve exclusivamente para classes de programas que utilizam o padrão de programação *MVVM* (*Model View View-Model*).



**Figura 3. Tela Create New Property**

A Figura 4 mostra a tela para se fazer comparações entre o esquema de um banco de dados com as classes mapeadas do sistema, para fins de criação e atualização das propriedades. No DSN (*Data Source Name*) é indicado um ODBC (*Open Database Connectivity*) já criado previamente no sistema operacional *Windows* para se conectar ao SGBD. O botão *New DSN* permite criar este ODBC. Após indicada a conexão, basta clicar no botão *Read DB*. O usuário é questionado se deseja fazer a comparação entre as tabelas do banco e as suas classes.

Na visualização de tabelas é mostrado em vermelho as tabelas que não tem vínculo com alguma classe; em laranja as que têm vínculo, mas estão diferentes em termos de propriedades mapeadas; e em verde as que estão equivalentes.

Schema	TableName	ClassName	ProjectItemRef
BasTutorial	Person	Person	
Cinema	Film	Film	
Cinema	FilmCategory	FilmCategory	
Cinema	Show	Show	
Cinema	Theater	Theater	
Cinema	TicketItem	TicketItem	
Cinema	TicketOrder	TicketOrder	
BasTutorial	Person	Person	

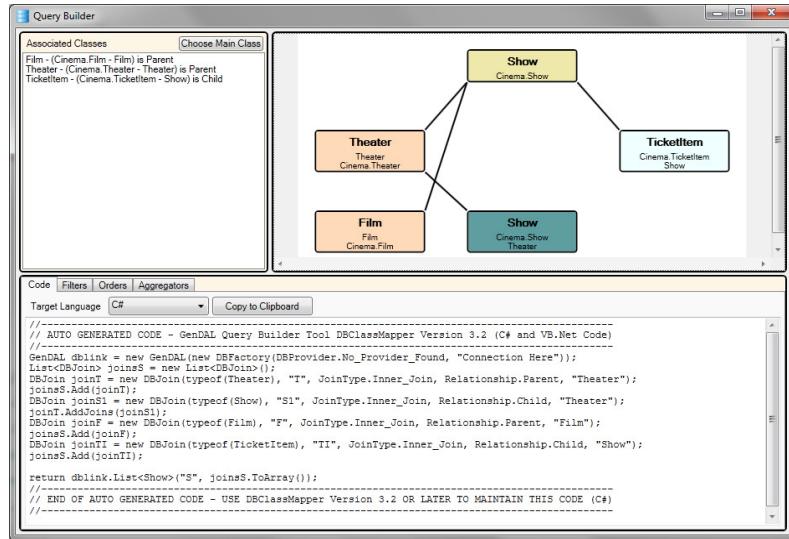
  

Create	Name	TypeName	IsNullable	Range	IsPrimaryKey
<input checked="" type="checkbox"/>	TicketOrder	INTEGER	<input type="checkbox"/>	10	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	ID	VARCHAR	<input type="checkbox"/>	254	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	AdultTickets	INTEGER	<input checked="" type="checkbox"/>	10	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	ChildTickets	INTEGER	<input checked="" type="checkbox"/>	10	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Show	INTEGER	<input checked="" type="checkbox"/>	10	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	childsub	INTEGER	<input type="checkbox"/>	10	<input checked="" type="checkbox"/>

**Figura 4. Tela correspondente à Read a Database to Create Full Class**

A Figura 5 mostra a tela correspondente à funcionalidade *Create Queries Based on Models*, que é um importante diferencial da *DBClassMapper*. Esta tela permite a definição de consultas baseadas nas classes de objetos já mapeados. Ao centro estão todos os nodos, que representam classes, e embaixo ficam as listas de filtros, ordenações e agregações, além do próprio código gerado para a consulta. No lado esquerdo da tela existe a lista de classes que se relacionam com o nodo selecionado, sendo que o primeiro nodo é o primeiro a ser definido a partir da lista de todas as classes mapeadas.

Clicando com o botão direito sob um nodo é possível definir os filtros, ordenações, *alias* e tipo de junção a ser realizado (*inner*, *left* ou *right*).



**Figura 5. Tela Create Queries Based on Models**

Os exemplos apresentados nas Figuras 4 e 5 foram executados em bancos de dados reais contendo dados fictícios no domínio de Cinema. Além disso, a ferramenta está sendo utilizada por uma empresa na área de TI da região e o *feedback* desta utilização foi bastante positivo em termos de grau de satisfação.

### 3. Conclusão

Este artigo apresenta *DBClassMapper*, uma ferramenta gráfica para usuários desenvolvedores de aplicações orientadas a objetos que integra diversas funcionalidades relacionadas ao processo de mapeamento objeto-relacional. Ela foi especificamente desenvolvida para auxiliar o *ORM Gendal* com qualquer SGBD.

A principal contribuição da *DBClassMapper*, se comparada com trabalhos relacionados, como o *Entity Framework* [Entity Framework 2013] e o *Hibernate* [Hibernate 2013], é justamente o fato de disponibilizar diversas funcionalidades de outras já existentes integradas em uma única ferramenta. É a primeira ferramenta para o *ORM Gendal* e uma das poucas (talvez a única) existente para um *ORM* para o *.Net* no *Visual Studio 2010*. Outra inovação importante é o gerador interativo de consultas, funcionalidade não disponível nessas ferramentas similares.

Trabalhos futuros incluem: (i) geração das tabelas e campos do banco de dados a partir das classes, uma vez que, atualmente o esquema do banco deve existir *a priori*; (ii) avaliação de usabilidade; (iii) disponibilidade da ferramenta em outras plataformas.

### Referências

- Ambler, Scott W. *Agile Database Techniques*. 1.ed. Nova Yorque: Wiley & Sons, 2003.
- Entity Framework. <http://msdn.microsoft.com/en-us/data/ef.aspx>. Acesso em: 24/01/2013.
- Hibernate. <http://www.hibernate.org/subprojects/tools.html>. Acesso em: 24/01/2013.
- ORM. <http://pt.wikipedia.org/wiki/ORM>. Acesso em: 24/01/2013.

# **Uma Aplicação baseada em SIG para Análise de Acidentes de Trânsito: Estudo de caso na Rodovia BR-101/ES**

**Wdnei R. Paixão<sup>1</sup>, Karin S. Komati<sup>1</sup>**

<sup>1</sup>Instituto Federal do Espírito Santo (IFES – Campus Serra)  
Rodovia ES-010 - Km 6,5 – Manguinhos 29.173-087 - Serra - ES

[wdneipaixao@gmail.com](mailto:wdneipaixao@gmail.com), [kkomati@ifes.edu.br](mailto:kkomati@ifes.edu.br)

**Abstract.** This work describes an application based on GIS (Geographic Information System), system used for the development of thematic map of traffic accidents occurred during the year 2011 on the stretch of the highway BR-101, located in the state of Espírito Santo (ES). This solution begins with the extraction of accident data provided by DNIT and the inclusion of these data in a PostgreSQL database, then is performed the georeferencing of the highway BR-101 in PostGIS, and ends with the creation of thematic map through Quantum GIS.

**Resumo.** Este trabalho descreve uma aplicação baseada em GIS (Geographic Information System), sistema usado para o desenvolvimento de mapa temático de acidentes de trânsito ocorridos no ano de 2011, no trecho da rodovia BR-101, localizado no Estado do Espírito Santo (ES). Esta solução começa com a extração de dados de acidentes fornecidos pelo DNIT e a inclusão destes dados em um banco de dados em PostgreSQL; a seguir é executado o georreferenciamento da rodovia BR-101 em PostGIS, e por fim cria-se o mapa temático através da Quantum GIS.

## **1. Introdução**

Os acidentes de trânsito representam um sério problema da vida moderna, pois são causadores de perdas e incapacidades físicas, além de gerar altos custos sociais e econômicos. No Brasil, as companhias de seguros indenizaram, em 2010, 51.000 sinistros de morte e 152.000 sinistros de invalidez permanente, que passaram a 58.000 e 240.000 respetivamente em 2011. O custo socioeconômico foi recentemente avaliado em quarenta bilhões de reais por ano [Associação Brasileira de Prevenção dos Acidentes de Trânsito, 2012].

Para entender e diagnosticar esse problema, a utilização de ferramentas computacionais e análise espacial dessas informações são essenciais. Este trabalho foca nas informações dos acidentes de trânsito da rodovia BR-101 no trecho que se localiza no Estado do Espírito Santo (ES) durante o ano de 2011, enfatizando os passos do ciclo de vida do sistema de geoprocessamento, desde a obtenção das informações até a criação dos mapas temáticos. Outro objetivo deste trabalho é utilizar somente softwares livres para que o trabalho descrito possa ser reproduzido e/ou estendido sem custos.

## 2. Metodologia e Desenvolvimento

Um mapa temático é um mapa que usa uma determinada variedade de estilos gráficos (cores, hachuras e legendas) para apresentar dados graficamente. Apresentam os dados de forma qualitativa, em escalas ou classes, sobre um fundo geográfico. A metodologia usada para a criação de mapas temáticos das informações de acidentes de trânsito foi: a extração dos dados dos acidentes de trânsito, o georreferenciamento da rodovia BR-101/ES e finalmente a criação dos mapas temáticos [Souza, 2011].

### 2.1. Extração das Informações dos Acidentes de Trânsito

Os dados sobre os acidentes ocorridos nas rodovias brasileiras podem ser encontrados no site do Departamento Nacional de Infraestrutura de Transportes [DNIT, 2012]. Os registros são disponibilizados em arquivos no formato “pdf”, com todas as ocorrências de um determinado ano. A Figura 1 mostra a parte superior de uma página do arquivo do ano de 2011, onde é possível verificar que cada acidente é localizado em um trecho de 100 metros, além disso, há informações sobre o uso do solo (zona rural ou urbana), data e hora do acidente, tipo do acidente, gravidade e se houve feridos ou mortos em cada um dos acidentes. Um pequeno aplicativo foi desenvolvido, na linguagem Java, para ler esse arquivo e incluir todos os dados de acidentes da BR-101/ES em uma base de dados mantida no PostgreSQL 9.2 [The PostgreSQL Group, 2012].

ACIDENTES POR QUILÔMETRO (RESUMIDO)													
UF: ES	BR-101	Período de 01/01/2011 00:00:00 a 31/12/2011 23:59:00											
Local Km.0: DIV BA/ES													
KM. 0 - AMBOS OS SENTIDOS													
Km	Uso do Solo	Hora	Data	Tipo do Acidente	Gravidade	Feridos	Mortos						
0,1	URBANO	12:50	29/12/2011 qui	Colisão traseira	Sem Vítima	0	0						
0,2	RURAL	07:40	13/05/2011 sex	Colisão traseira	Sem Vítima	0	0						
0,2	RURAL	00:10	10/07/2011 dom	Capotagem	Sem Vítima	0	0						
0,3	RURAL	08:00	08/02/2011 ter	Colisão traseira	Com Ferido	1	0						

Figura 1. Exemplo dos dados fornecidos pelo DNIT.

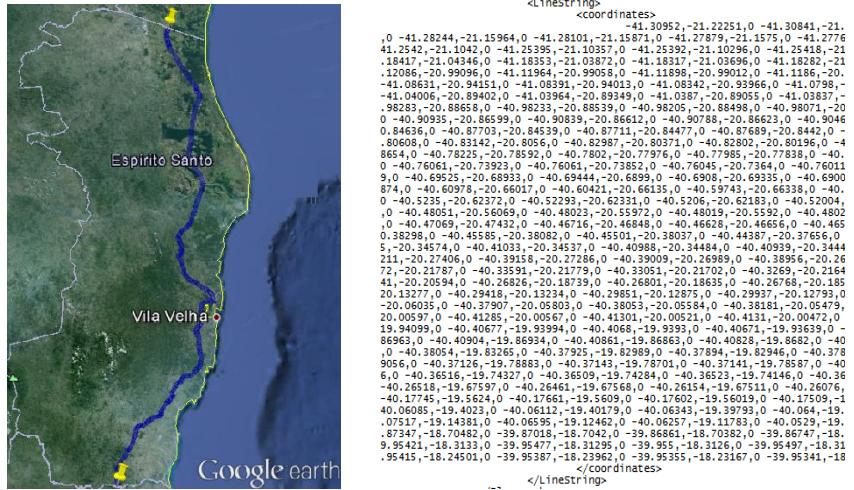
### 2.2. Georreferenciamento da Rodovia BR-101/ES

Georreferenciamento de um mapa, ou qualquer outra forma de informação geográfica, é tornar suas coordenadas conhecidas num dado sistema de referência. Esse processo inicia-se com a obtenção das coordenadas de pontos do mapa, ou da informação geográfica, a serem georreferenciados, conhecidos como pontos de controle. Os pontos de controle são locais que oferecem uma feição física perfeitamente identificável, tais como intersecções de estradas e de rios, represas, pistas de aeroportos, edifícios proeminentes, topos de montanha, entre outros.

A informação geográfica deste estudo é a rodovia BR-101/ES. Para se obter os pontos de controle da Rodovia BR-101/ES foi utilizada a ferramenta “Google Earth” [Google, 2012], que fornece funcionalidade para demarcar pontos no mapa em forma de linha. A partir desta linha foi gerado um arquivo com extensão “kml”, que possui todas as coordenadas geográficas dos vértices que formam a BR-101/ES. A Figura 2 mostra a extensão da rodovia BR-101, localizada no ES, e, ao lado, uma parte do arquivo com as coordenadas dos vértices.

Todas as coordenadas foram inseridas no PostGIS 2.0.2 [PostGIS PSC, 2012]. O PostGIS é uma extensão espacial gratuita do PostgreSQL, que permite o armazenamento e uso de objetos SIG. Utilizando as facilidades do PostGIS, criou-se vários pontos na

rodovia, de forma a serem equidistantes em 100 metros, para que fossem compatíveis com as informações de localização dos acidentes fornecidos pelo DNIT.



**Figura 2.** A primeira imagem mostra a demarcação da linha da BR-101/ES a partir do Google Earth e a segunda imagem mostra as coordenadas dos vértices que formam a BR-101/ES.

### 2.3. Criação dos Mapas Temáticos

Foram criadas visões no PostgreSQL que forneciam o número de acidentes ocorridos por trimestre do ano de 2011, associados aos seus pontos geográficos. Com esses dados criou-se um modelo digital de elevação (um mapa em que o valor de cada célula representa a altitude do terreno) através do Quantum GIS [QGIS PSC, 2012], que é um visualizador de dados geográficos com interface amigável. Os valores das células foram calculados pelo método de interpolação IDW (*Inverse Distance Weighting*) do plugin “Interpolation” do Quantum GIS. Cada altitude é classificada de acordo com uma escala de cores, representando qualitativamente a densidade de acidentes, conforme Figura 3.



**Figura 3.** Acidentes ocorridos por trimestre na BR-101/ES em 2011.

É possível verificar que a densidade de acidentes foi maior no primeiro trimestre do ano, que nos trimestres posteriores. Uma possível hipótese é que o Carnaval ocorreu em março, e durante este feriado é comum o aumento de acidentes [TV Gazeta, 2011].

### 3. Considerações Finais

Este trabalho está em desenvolvimento e neste artigo apresentou-se o resultado parcial, envolvendo a metodologia de desenvolvimento e a criação de um único mapa temático.

Trabalhos correlatos, como os de Camarez e Higashi [2011] e Santos [2006], focam na análise de acidentes que ocorrem dentro de um município específico - Florianópolis e São Carlos, respectivamente. Diferentemente, este trabalho tem o foco em rodovias e, futuramente, espera-se abranger todas as rodovias do país. Como trabalhos futuros, planeja-se:

- Automatizar o processo de entrada de dados;
- Incluir todos os acidentes de todas as rodovias do ES e posteriormente do Brasil;
- Incluir todos os acidentes dos anos de 2009, 2010 e 2011. Caso o DNIT disponibilize o arquivo de 2012 até o meio do ano de 2013, este também será incluído;
- Criação de outros mapas temáticos:
  - acidentes por mês/ano com variações no tipo, gravidade e vítimas fatais;
  - acidentes por dia da semana/ano;
  - acidentes por turno do dia (matutino, vespertino e noturno)/ano.
- Mesclar cores e dados quantitativos na visualização dos mapas;
- Previsão do número de acidentes de acordo com dados temporais.

Assim, por meio de um SIG, pretende-se elaborar vários mapas temáticos, de forma a fornecer uma leitura clara, sobre localização, concentração e comportamento dessas ocorrências, tentando contribuir para a segurança e o planejamento viário.

## Referências

- Associação Brasileira de Prevenção dos Acidentes de Trânsito. (2012) “Por Vias Seguras”, <http://vias-seguras.com>, Dezembro.
- Camarez, M. L. e Higashi, R. A. R. (2011) "Utilização de técnicas de geoprocessamento através de um SIG para a estimativa de características mecânicas dos solos do município de Florianópolis". Em: Anais XV do SBSR, Curitiba, PR, Brasil.
- DNIT. (2012) “Site Oficial”. <http://www.dnit.gov.br/>, Dezembro.
- Google. (2012) “Google Earth”. <http://www.google.com/earth/index.html>, Dezembro.
- PostGIS PSC. (2012) “PostGIS”, <http://www.postgis.org>, Agosto.
- QGIS PSC. (2012) “Quantum GIS”, <http://www.qgis.org>, Agosto.
- Santos, L. (2006) “Análise dos acidentes de trânsito do Município de São Carlos utilizando Sistema de Informação Geográfica – SIG e ferramentas de estatística espacial”. 138p. Dissertação de Mestrado em Eng. Urbana, UFSCar.
- Souza, G. A. (2011) “Georreferenciamento de Acidentes de Trânsito: uma Discussão Metodológica”. *ACTA Geográfica*, pp. 31-40, Ed. Cidades na Amazônia Brasileira.
- The PostgreSQL Group. (2012) “PostgreSQL”. <http://www.postgresql.org/>, Dezembro.
- TV Gazeta (2011) “Carnaval: 413 acidentes 10 mortes nas estradas que cortam o ES”, [http://gazetaonline.globo.com/\\_conteudo/2011/03/noticias/tv\\_gazeta/jornalismo/bom\\_dia\\_es/793938-carnaval-194-acidentes-e-7-mortes-nas-estradas-federais-que-cortam-o-es.html](http://gazetaonline.globo.com/_conteudo/2011/03/noticias/tv_gazeta/jornalismo/bom_dia_es/793938-carnaval-194-acidentes-e-7-mortes-nas-estradas-federais-que-cortam-o-es.html), Março.

# Implementação de um Repositório de Versões de Serviços Web usando OrientDB

Lucas J. K. Alves, Karin Becker

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)  
Porto Alegre, Brasil

lucaskalves@gmail.com.br, karin.becker@inf.ufrgs.br

**Resumo.** Versionamento é uma técnica muito usada para gerenciar mudanças em web services, visando minimizar o impacto em aplicações cliente. Este artigo apresenta uma implementação eficiente de um repositório de versões de serviços usando OrientDB, um sistema de gestão de banco de dados orientado a grafos.

## 1. Introdução

Arquiteturas orientadas a serviço e *web services* tornaram-se padrão para o desenvolvimento de aplicações de baixo acoplamento. Uma descrição usando uma linguagem padronizada (*e.g.* Web Service Definition Language (WSDL)), fornece às aplicações cliente os aspectos externamente relevantes de um serviço. Para evitar impactar clientes devido a mudanças na interface de um serviço, muitas vezes o provedor cria versões deste, possibilitando que o cliente utilize uma versão anterior até que possa ajustar-se às mudanças. O cliente deve identificar as porções do serviço modificadas e se elas lhe causam impacto negativo (*i.e.* incompatíveis). Esta tarefa é difícil, principalmente em caso de descrições extensas e modificações frequentes (*e.g.* serviço Trading<sup>1</sup> do eBay, com 270.000 linhas e modificado a cada duas semanas).

Yamashita *et al.* (2012)(a) propuseram um modelo de versionamento de serviços mais granular que facilita a gestão de versões, junto com um algoritmo de versionamento que converte uma descrição WSDL em versões neste modelo. Assim, é possível versionar apenas as partes da descrição que foram alteradas, e relacioná-las com versões pré-existentes das porções inalteradas do serviço. O presente artigo apresenta uma implementação eficiente de um repositório de versões para este modelo, utilizando um sistema de gestão de banco de dados (SGBD) orientado a grafos.

No restante deste artigo, a Seção 2 apresenta o modelo de versionamento e a sua implementação original. A Seção 3 descreve a implementação baseada em grafos e a Seção 4 analisa seu desempenho. A Seção 5 apresenta conclusões e direções futuras.

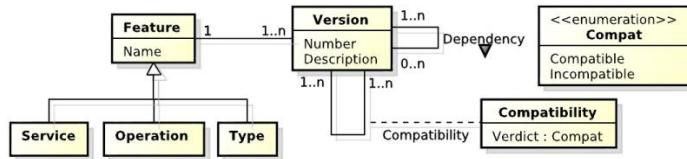
## 2. Modelo de Versões Orientado a *Features* e Implementação Original

Para um maior controle sobre as partes específicas da interface de um serviço que são alteradas, Yamashita *et al.* (2012)(a) propuseram um modelo de versionamento orientado a características (*features*), descrito no diagrama da Figura 1. Uma *feature* relaciona-se a um trecho textual de uma descrição WSDL, correspondendo a um aspecto de um serviço, operação ou tipo de dado. *Features* são versionadas, e uma interface de

---

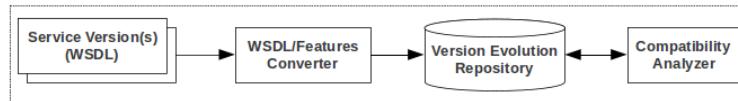
<sup>1</sup> <http://developer.ebay.com/DevZone/XML/docs/Reference/eBay/>

serviço é então descrita por uma coleção de versões interconectadas de *features*, formando um grafo.



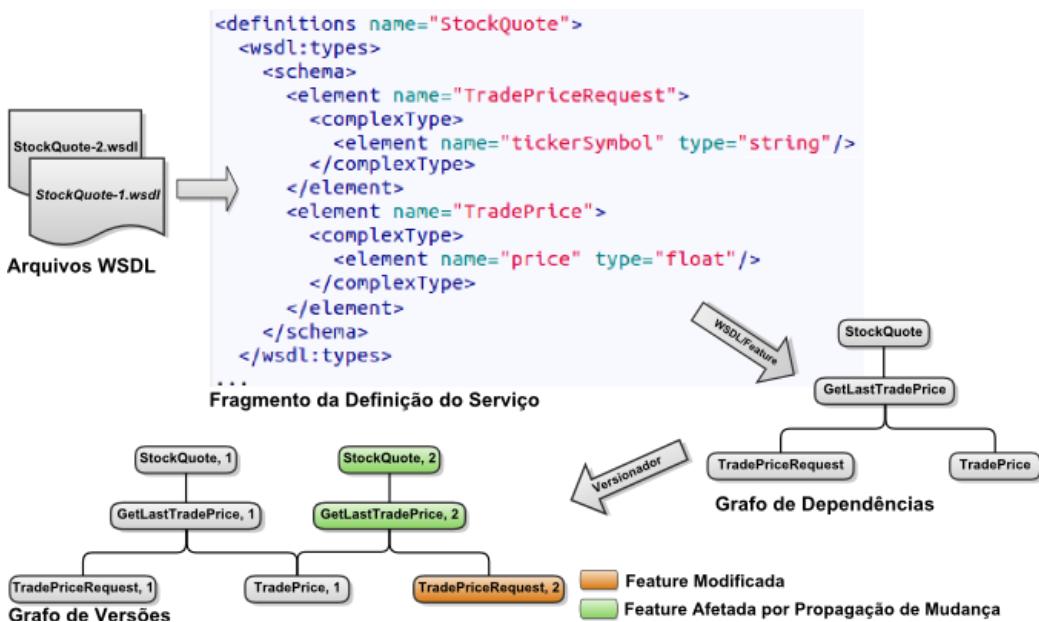
**Figura 1. Modelo de Versões Orientado a Features**

Um *framework* de evolução de serviços (Figura 2) recebe uma nova descrição WSDL, converte esta descrição textual no modelo de versões orientada a *features*, detectando as partes alteradas em relação a versões pré-existentes. Assim, cria novas versões quando detecta mudanças, ou estabelece relações de dependência com versões existentes de *features*. O resultado é armazenado em um repositório de versões.



**Figura 2. Framework de Evolução de Serviços**

O processo de criação de versões está representado na Figura 3. Primeiramente, converte-se cada elemento da definição WSDL em uma *feature*. Depois, cria-se um grafo de dependências, onde os vértices representam *features* e as arestas, a relação de dependência entre elas. Uma dependência existe se a definição de uma *feature* é baseada em outra (e.g. uma operação que utiliza como parâmetro um tipo descrito no mesmo WSDL). Após, compara-se cada *feature* desse grafo com as *features* equivalentes presentes no repositório de versões, versionando a *feature* se uma mudança é detectada. O caminhamento no grafo é *bottom up*, isto é, comparação começa pelas *features* sem dependentes. As mudanças podem ocorrer devido a alterações na descrição da *feature* ou devido a alterações em alguma dependência dela (propagação). Sempre que uma *feature* não é modificada, uma versão equivalente dela existente no repositório é reaproveitada, evitando redundâncias.



**Figura 3. Diagrama de Versionamento**

A implementação original do sistema de versionamento não apresentava bom desempenho devido à forma como as informações eram persistidas, bem como ao processo de comparação de versões. Os dados eram persistidos em arquivos XML puros (sem nenhum SGBD), o que dificultava a centralização e manipulação do repositório de versões. Além disso, considerando as bibliotecas de manipulação de arquivos XML utilizadas (SAX e JDOM), o processo de comparação de versões exigia o caminhamento entre os subgrafos de cada *feature* do repositório para detectar mudanças nas suas dependências. Isso impactava significativamente seu desempenho.

### 3. Implementação Baseada em Grafos

Para resolver esses problemas, foram tomadas duas ações: a) utilizar SGBD orientado a grafos para persistir os dados das versões; b) melhorar o processo de comparação de *features* com versões do repositório para aumentar seu desempenho. Foi escolhido utilizar um SGBD orientado a grafos em vez de, por exemplo, um SGBD XML, pois o tipo de estrutura grafo representa bem o modelo de versionamento utilizado. Utilizou-se o OrientDB<sup>2</sup>, o qual é *Open-Source*, permite diversos níveis de controle de integridade (*ACID*, *Multiversion Concurrency Control* e *Basically Available, Soft State, Eventual Consistency*), tem licença permissiva e disponibiliza diversas maneiras de acesso aos dados (SQL, API Nativa e Gremlin<sup>3</sup>). OrientDB permite representar dados como vértices e arestas, cujos atributos são descritos através de classes associadas a estes.

Considerando o modelo da Figura 1, a classe *Version* foi mapeada para o conceito de vértice do OrientDB, e a associação *dependency* entre versões, no conceito de aresta. Como existem versões de serviços, operações e tipos, foram definidas as classes *ServiceVersion*, *TypeVersion* e *OperationVersion*, e associadas aos vértices. Definiu-se uma superclasse *FeatureVersion* que define os atributos comuns *name*, *description*, *number* e *signature*, este último, detalhado abaixo. A relação *dependency* pode ser consultada utilizando as maneiras de acesso disponíveis pelo OrientDB.

Na implementação original, o processo de comparação envolvia o caminhamento pela nova definição WSDL e compará-la com o repositório de versões XML utilizando as API citadas. Esse processo de comparação de versões foi alterado para evitar a necessidade de comparação recursiva de subgrafos no repositório, visando detectar mudanças nas dependências. O atributo *signature* (assinatura) de uma *feature* é um código *hash* criado a partir da descrição textual da *feature*, combinado com as assinaturas de suas dependências, recursivamente. A partir da definição WSDL, é possível recriar a assinatura e compará-la com as assinaturas das versões presentes no repositório. Assim, é possível detectar mudanças nas *features* apenas comparando as suas assinaturas diretamente. Primeiramente, procuramos por *features* no repositório com o mesmo nome da *feature* sendo analisada. Se existir e tiver uma versão com a mesma assinatura, essa versão é reaproveitada. Se existir, mas nenhuma tiver a mesma assinatura, trata-se de uma nova versão. Se não existir *feature* com o mesmo nome, é uma nova *feature*.

---

<sup>2</sup> <http://www.orientdb.org/>

<sup>3</sup> <http://gremlin.tinkerpop.com/>

## 4. Experimento

Para comparar a eficiência da implementação original com a atual, foram versionadas vinte descrições do serviço Trading do provedor eBay. Esse serviço é bastante complexo, com muitas operações e tipos, logo, o modelo de versionamento é testado em situações compatíveis com aplicações reais. Foram comparados os tempos de criação de uma nova versão utilizando a implementação original XML e a implementação OrientDB, sempre no mesmo ambiente computacional. A melhora de desempenho obtida através da nova implementação pode ser vista no gráfico da Figura 4.

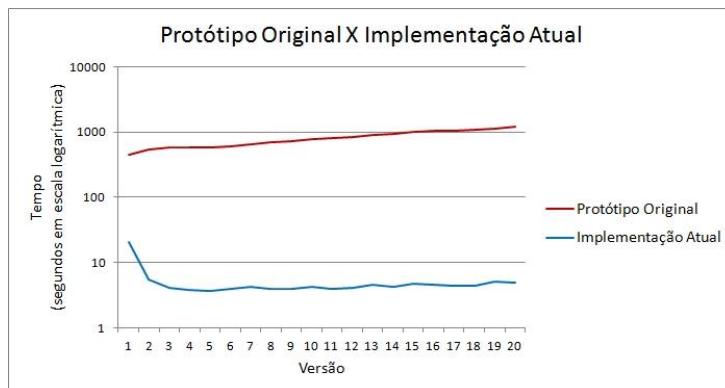


Figura 4. Comparação de Desempenho entre as Implementações

## 5. Considerações Finais e Trabalhos Futuros

As mudanças propostas na implementação do sistema de versionamento aumentaram significativamente seu desempenho. O uso de um SGBD, comparado com o repositório em arquivo, facilitou a centralização das informações. O uso de um SGBD orientado a grafos contribuiu para a melhoria do desempenho, por alinhar-se com as relações de dependência entre versões. A mudança no processo de comparação diminuiu a quantidade de acessos ao repositório. O OrientDB facilitou também as consultas na estrutura de grafo, muito importante para outras aplicações do *framework*, tais como detecção de compatibilidade (Yamashita *et al.* 2012 (b)).

Entre as vantagens percebidas na utilização do OrientDB estão seu bom desempenho, sua facilidade de integração com aplicações Java, interface gráfica para manuseio de seus dados e existência de uma ótima comunidade de desenvolvimento. Como principal dificuldade cita-se a escassez de guias e livros a seu respeito.

Futuramente, integraremos o sistema de gestão de versões com os outros módulos do *framework*, tais como o módulo de gerencia de perfis de uso e de análise de uso (Yamashita *et al.* 2012 (b)).

## Referências

- Yamashita, M., Becker, K. e Galante, R. (2012). “A Feature-based Versioning Approach for Assessing Service Compatibility”. *JIDM* 3(2): 120-131.
- Yamashita M., Vollino B., Becker K. e Galante R. (2012). “Measuring Change Impact based on Usage Profiles”. Em *Proceedings of the ICWS*, 2012. p. 226-233.
- W3C. “WSDL – Web Service Description Language”. Capturado em <http://www.w3c.org/TR/wsdl> (Agosto 2012).

# **DBModeler: Um sistema web para criação, manutenção e consulta de diagramas de bancos de dados**

**Samuel S. Troina<sup>1</sup>, Karina S. Machado<sup>1</sup>**

<sup>1</sup>Centro de Ciências Computacionais – Universidade Federal do Rio Grande(FURG)  
Avenida Itália, km 8, Bairro Carreiros – 96203-900 – Rio Grande – RS – Brazil

{samueltroina, karina.machado}@furg.br

**Resumo.** *Em um ambiente de desenvolvimento de sistemas de informação, os envolvidos no processo, programadores, administradores de banco de dados, analistas de sistemas e outros, necessitam constantemente consultar o modelo lógico do banco de dados. Tal consulta geralmente dá-se através dos diagramas de entidade relacionamento, estes representados na maioria das vezes por meio da notação IE - Engenharia da Informação. Diante dessa necessidade pelos diagramas, o presente trabalho apresenta a ferramenta DBModeler, um sistema web para criação, manutenção e consulta de diagramas de bancos de dados. Com o uso desse sistema é possível manter um ambiente compartilhado de acesso aos diferentes diagramas que podem ser facilmente criados e manipulados em uma interface amigável.*

## **1. Introdução**

Durante o desenvolvimento de um sistema de informação que necessite armazenamento de suas informações em um banco de dados, a equipe responsável por sua criação, programadores, analistas de sistemas, administradores de banco de dados (BD) e outros necessitam determinar o esquema lógico em que será feita a persistência dos dados da aplicação a ser desenvolvida [Bassi Filho, 2008]. Para a realização desta etapa do projeto, utilizam um recurso de extrema importância, o modelo lógico de representação do esquema do BD, onde este representa como os dados serão armazenados.

Para criação do modelo lógico geralmente utiliza-se uma técnica muito popular entre os projetistas de sistemas: a modelagem entidade relacionamento, mais conhecida como ER [Chen, 1976; Ramakrishnan and Gehrke, 2008]. Há inúmeras ferramentas para a elaboração de diagramas ER, entretanto é observado que muitas necessidades dos analistas e administradores não estão atualmente sendo bem atendidas. Por exemplo, a ferramenta para a criação e consulta de diagramas de banco de dados deve apresentar uma interface amigável e de fácil acesso e que permita a disponibilização da última versão do modelo e a fácil alteração e comparação com versões anteriores.

Sendo assim, este trabalho apresenta a ferramenta Web DBModeler para a criação, consulta e atualização de diagramas ER de Bancos de Dados. O DBModeler é uma ferramenta livre, que pode ser executada em diferentes sistemas operacionais e que permite o acesso por parte de diferentes usuários.

## 2. Trabalhos relacionados

Antes do desenvolvimento desta ferramenta foi realizada uma revisão sobre as principais ferramentas utilizadas atualmente para a elaboração de diagramas ER na Web. As ferramentas analisadas foram comparadas em relação as propriedades descritas na Tabela 1, considerando as principais características e deficiências dos sistemas desenvolvidos para esse fim na Web. Os itens da tabela que indicam “TF” correspondem a propriedades que o DBModeler já está preparado para oferecer, porém na versão atual não está incorporado pois se encontra em fase de teste.

**Tabela 1 - Tabela de comparação entre os trabalhos relacionados.**

Propriedades	WWW SQL Designer <sup>1</sup>	Creately <sup>2</sup>	DB Schema Editor <sup>3</sup>	DBModeler
Versionamento de Modelos	Não	Não	Não	TF
Permissões de acesso por modelo	Não	Não	Não	Sim
Disponibilidade na Intranet local	Sim	Não	Não	Sim
Limitação da estrutura do banco de dados	Não	Não	Não	Não
Múltiplos diagramas por modelo	Não	Não	Não	Sim
Engenharia reversa	Sim	Não	Não	TF

1 <http://code.google.com/p/wwwsqldesigner/>

2 <http://creately.com/>

3 <http://www.dbschemaeditor.com/>

Além de o DBModeler ser uma ferramenta gratuita, sendo este um diferencial importante em relação as demais ferramentas existentes para o mesmo fim, destaca-se como características importantes: ser totalmente on-line, oferecer controle de acesso aos modelos baseado em permissão concedida por seu criador, permitir múltiplos diagramas por modelo, identificar os esquemas por meio de diferentes cores, entre outras.

## 3. O sistema DBModeler

O DBModeler é um sistema Web que permite que analistas, programadores e administradores de banco de dados criem seus diagramas de banco de dados relacionais seguindo a notação *Crow's Foot*, mas conhecida como pés de galinha [Everest, 1976]. Essa notação foi escolhida por ser uma notação muito popular e já incorporada na maioria dos softwares existentes. Pretende-se incluir outras notações no futuro.

### 3.1. Requisitos de sistema e funcionalidades oferecidas

O sistema precisa ser instalado localmente ou em um servidor Web que possua suporte a linguagem de programação PHP. O DBModeler foi desenvolvido com suporte para a persistências a três diferentes sistemas gerenciadores de banco de dados (SGBDs): o SimpleDB, o MySQL e o PostgreSQL, selecionando o SGBD de acordo com a configuração definida pelo usuário.

### 3.2 Materiais e métodos

Para o desenvolvimento do software DBModeler foram utilizadas apenas tecnologias de uso gratuito. O *HyperText Markup Language* (HTML) e *Cascading Style Sheets*

(CSS) foram as tecnologias empregadas para a construção da camada de apresentação do sistema, responsável pela interface de autenticação, formulários de cadastro dos novos usuários, listagem dos modelos existentes e para os menus, tanto da estrutura do BD como da listagem dos modelos. A linguagem *Scalable Vector Graphics* (SVG) foi utilizada para a descrição dos desenhos em forma vetorial, sendo suportado pela maioria dos navegadores Web de hoje [SVG Tutorial, 2012].

A linguagem *Javascript* foi utilizada para conceder ao sistema mais dinamismo, como, por exemplo, arrastar as tabelas, os menus de *context*, comunicação com o servidor, entre outras. A linguagem de programação PHP permite o desenvolvimento de sites dinâmicos, voltada à internet, amplamente utilizada e gratuita, sendo por isso escolhida para o desenvolvimento do DBModeler [Auchor et al., 2012].

### 3.3 Base de dados associadas ao DBModeler

O sistema DBModeler utiliza uma pequena base de dados para armazenar os dados referentes aos seus usuários, modelos, suas versões e as permissões dos usuários sobre os modelos existentes. Os modelos criados utilizando o sistema proposto são armazenados por meio de arquivos no formato *Extensible Markup Language* (XML), seguindo a recomendação da W3C para o XML 1.0 [XML; 2012], com codificação dos caracteres padrão UTF-8. Para garantir a segurança da informação, além do controle de acesso, também mantém a proteção aos arquivos, bloqueando os diretórios onde estão armazenados fisicamente os modelos por meio de diretivas de segurança com arquivos de configuração distribuídas.

### 3.4 Funcionamento do DBModeler

A tela principal do DBModeler é apresentada na Figura 1 e é dividida em 3 áreas: a área de desenho, onde se visualiza o diagrama em edição, a área 2, que resume a estrutura do modelo que está sendo criado e a área 3, que lista os diagramas de um determinado usuário. O primeiro passo consiste em criar um novo diagrama, por meio de um clique com o botão direito sobre a área 3. As operações de alteração e remoção de um diagrama são invocadas por meio do clique com o botão direito do mouse sobre a descrição do diagrama desejado.

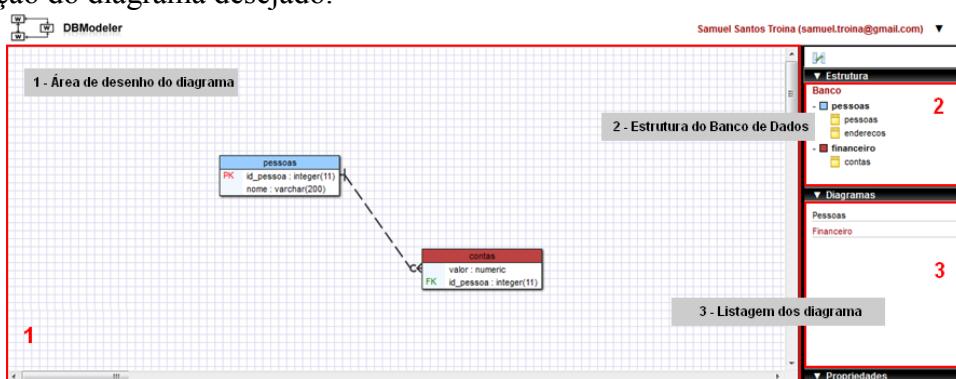
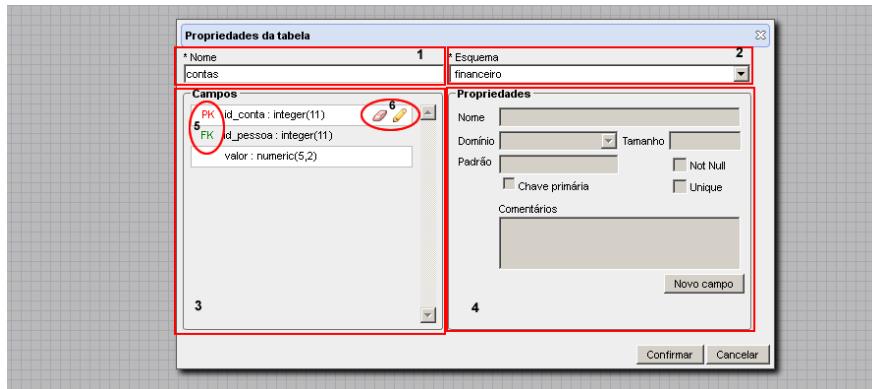


Figura 1. Tela principal: área de desenho do modelo e estrutura do BD.

Para cada tabela, podem ser adicionados e editados os campos, conforme mostra a Figura 2, onde em 1 tem-se o nome da tabela, em 2, a qual esquema ela pertence, em 3 a listagem dos campos, em 4 a área de edição das propriedades dos campos, em 5 a

identificação das chaves primárias e secundárias e em 6 as operações de editar e excluir campo. Para uma melhor identificação dos esquemas no diagrama, estes poderão ser destacados por meio de cores onde se pode definir uma cor para sua representação no diagrama, assim todas as tabelas pertencentes a este esquema possuirão a cor selecionada no seu título.



**Figura 2 - Propriedades de uma tabela no DBModeler.**

#### 4. Considerações finais

Através do desenvolvimento do sistema DBModeler espera-se que uma lacuna existente na web seja preenchida uma vez que o seu uso permite criar, consultar e manter os diagramas de bancos de dados, por meio da Web, bastando possuir um navegador. A versão atual do DBModeler já está preparada para uma série de funcionalidades que serão futuramente incorporadas no software, como a possibilidade de realizar engenharia reversa, suporte a proveniência de modelos, mantendo o versionamento de modelos, geração do modelo do banco de dados físico com SQL, permitindo que o modelo físico se mantenha sempre atualizado em relação ao diagrama ER.

#### 5. Referências

- Achour, Mehdi et al. Manual do PHP. Disponível em <[http://www.php.net/manual/pt\\_BR/](http://www.php.net/manual/pt_BR/)>. Acesso em Dez. 2012.
- Bassi Filho, DL. Experiências com desenvolvimento ágil. Dissertação de Mestrado em Ciência da Computação. Instituto de Matemática e Estatística, Universidade de São Paulo, 2008.
- Chen, P P-S. The Entity-Relationship Model--Toward a Unified View of Data. In: ACM Transactions on Database Systems, Vol 1(1), 9–36, 1976.
- Everest, G. Basic Data Structure Models Explained with a Common Example. In: Proc. Fifth IEEE Texas Conference on Computing Systems, Austin, TX, 39–45, 1976.
- Ramakrishnan, R and Gehrke, J. Sistemas de Gerenciamento de Bancos de Dados - 3.ed. Editora McGraw-Hill. 884 p. 2008.
- SVG Tutorial. Disponível em <http://www.w3schools.com/svg/>. Acessado em Dez. 2012.
- Extensible Markup Language (XML) 1.0 - W3C Recommendation 10. Disponível em <http://www.w3.org/TR/1998/REC-xml-19980210>. Acesso em Dez. 2012

# **Uma implementação de *feedback* da relevância utilizando o algoritmo Rocchio**

**Caroline Tomasini, André Prisco, Eduardo N. Borges**

Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)  
Rio Grande – RS – Brasil

{caroline.tomasini, andre.prisco, eduardoborges}@furg.br

**Abstract.** This paper describes an information retrieval tool that implements the vector space model and the Rocchio algorithm. Once a user has viewed the result of a query, he or she can point out which of the returned items are relevant. The system will optimize the query according to the user relevance feedback.

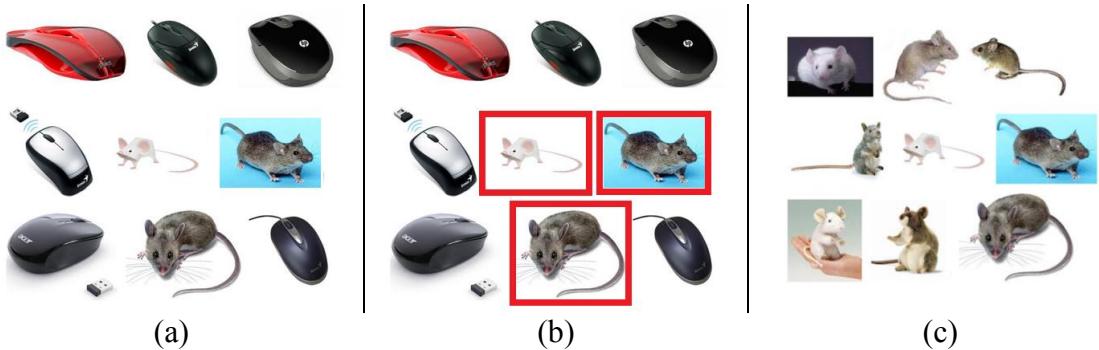
**Resumo.** Este artigo descreve uma ferramenta de recuperação de informações que implementa o modelo espacial vetorial e o algoritmo Rocchio. Depois que um usuário tenha visualizado o resultado de uma consulta, ele pode apontar quais dos itens retornados são relevantes. O sistema otimizará a consulta de acordo com o feedback de relevância do usuário.

## **1. Introdução**

Sistemas de recuperação de informações têm como objetivo encontrar, em grandes coleções, documentos de natureza pouco ou não estruturada que satisfaçam uma necessidade de informação [Baeza-Yates e Ribeiro Neto 1999]. A necessidade de informação do usuário geralmente é mapeada para uma consulta em linguagem natural ou através de palavras-chave. Entretanto, essas palavras podem ter múltiplos significados. Por exemplo, a consulta pelo termo *graça* poderia recuperar documentos com os seguintes segmentos de texto: “fiéis agradecem a graça recebida” e “o produto saiu de graça”. Essa propriedade, denominada polissemia, reduz a qualidade dos sistemas de recuperação de informação porque diminui a precisão do resultado, recuperando documentos que não fazem parte do interesse do usuário.

A sinonímia é outra propriedade que limita a qualidade dos sistemas de recuperação de informações porque diminui a abrangência do resultado. Por exemplo, a consulta pelo termo *carro* não seria capaz de recuperar documentos que contivessem apenas os termos *veículo* ou *automóvel*. O usuário teria que refinar a consulta diversas vezes até satisfazer sua necessidade de informação.

Para solucionar os problemas apresentados, foram propostos sistemas baseados no *feedback* da relevância do usuário [Manning et al. 2008]. A Figura 1 apresenta um exemplo do funcionamento de um sistema de recuperação de imagens deste tipo. O usuário realiza uma consulta pelo termo “*mouse*”. O sistema retorna um conjunto inicial de documentos (a). O usuário marca alguns documentos retornados como relevantes (b). O sistema recalcula a representação da necessidade de informação com base no *feedback* dado pelo usuário. Por fim, é exibido um conjunto revisto de resultados recuperados (c).



**Figura 1.** Exemplo de recuperação de informações com base no *feedback* da relevância. (a) Resultado original. (b) Seleção de relevantes. (c) Resultado final.

Este artigo apresenta uma ferramenta de recuperação de informações que implementa o modelo espacial vetorial e o algoritmo Rocchio, descritos nas próximas seções. Ela foi desenvolvida para apoiar o aprendizado na disciplina de Sistemas de Informações Avançados oferecida pelo Centro de Ciências Computacionais da FURG.

## 2. Modelo espacial vetorial

No modelo espacial vetorial [Baeza-Yates e Ribeiro Neto 1999], cada documento  $d$  é representado por um vetor com  $t$  dimensões, uma para cada termo do vocabulário de toda a coleção. O peso  $w_i$  de cada dimensão  $i$  é calculado a partir da frequência dos termos e tem a função de quantificar a relevância de cada termo para as consultas e para os documentos. A relevância do documento em relação a uma consulta  $q$  é dada por uma função de similaridade baseada no cosseno do ângulo formado pelos dois vetores, conforme a Equação 1 [Salton e Buckley 1988].

$$\text{sim}(d, q) = \frac{\sum_{i=1}^t w_{id} w_{iq}}{\sqrt{\sum_{i=1}^t w_{id}^2} \sqrt{\sum_{i=1}^t w_{iq}^2}} \quad (1)$$

Se uma coleção possui  $N$  documentos e  $df_i$  é a quantidade de documentos que possuem o termo  $t_i$ , então os pesos  $w_{id}$  são calculados através da métrica  $tf \times idf$  conforme a Equação 2, onde  $freq_{id}$  é a frequência do termo  $i$  no documento  $d$  e  $\max(freq_{td})$  é a máxima frequência de qualquer termo  $t$  presente no mesmo documento. Para as consultas, essa frequência normalizada ainda pode ser suavizada através de um fator de amortecimento. Perceba que quanto mais frequente é um termo num documento, maior o peso nessa dimensão. Entretanto, a frequência inversa  $idf$  reduz o peso de termos comuns na coleção, ou seja, daqueles que aparecem em muitos documentos.

$$w_{id} = tf_{id} idf_i = \frac{freq_{id}}{\max(freq_{td})} \log \frac{N}{df_i} \quad (2)$$

## 3. Algoritmo Rocchio

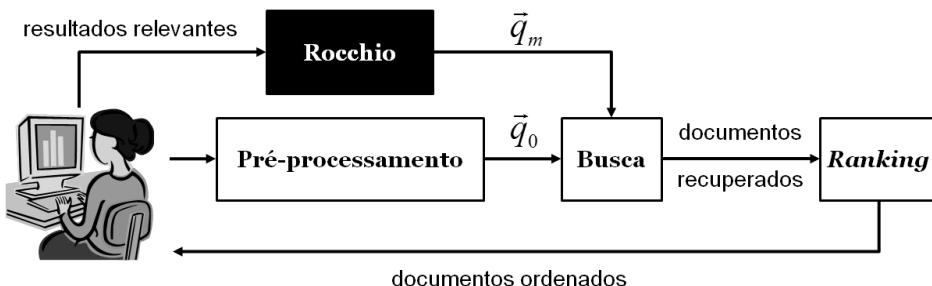
Rocchio (1971) é um algoritmo que incorpora as informações de *feedback* da relevância no modelo espacial vetorial, melhorando o resultado final das consultas. O algoritmo tem como objetivo encontrar um vetor consulta modificado  $\vec{q_m}$  que maximiza a similaridade com os documentos marcados como relevantes pelo usuário e que minimiza a semelhança com documentos não relevantes. A Equação 3 define o vetor

consulta modificado, onde  $\vec{q}_0$  é a consulta original,  $\vec{d}_j$  é a representação vetorial de um documento que pode pertencer ao conjunto de relevantes para o usuário  $D_r$  ou ao de não relevantes  $D_{nr}$ . A equação soma à consulta original a representação centroide dos documentos relevantes e diminui o centroide dos não relevantes. Os pesos  $\alpha$ ,  $\beta$  e  $\gamma$  ponderam a importância do *feedback* em relação à consulta original.

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j \quad (3)$$

#### 4. Implementação da ferramenta

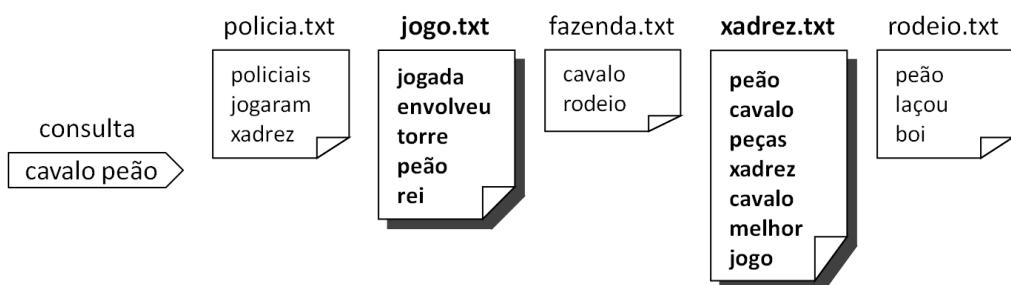
A Figura 2 apresenta a arquitetura da ferramenta desenvolvida. O usuário realiza uma consulta em linguagem natural que é entregue ao componente *Pré-processamento*. Este componente é responsável pelas etapas de decomposição e normalização do texto, remoção de palavras irrelevantes, composição do vocabulário de termos (dimensões do modelo espacial vetorial) e pela atribuição de pesos. Ele é utilizado para gerar a representação vetorial das consultas do usuário  $\vec{q}_0$  e de todos os documentos da coleção. O componente *Busca* pesquisa na coleção pelos documentos que contenham os termos de  $\vec{q}_0$ . A seguir, o componente *Ranking* calcula a similaridade entre  $\vec{q}_0$  e cada documento recuperado. Estes documentos são ordenados de acordo com a similaridade calculada e entregues ao usuário. Analisando o *ranking* de documentos, o usuário seleciona um conjunto de resultados relevantes para a consulta. Este *feedback* é usado para retroalimentar o sistema, permitindo ao componente *Rocchio* calcular uma nova representação da necessidade de informação do usuário  $\vec{q}_m$ . O novo vetor de consulta modificado é submetido ao componente *Busca* e o restante do processo é repetido.



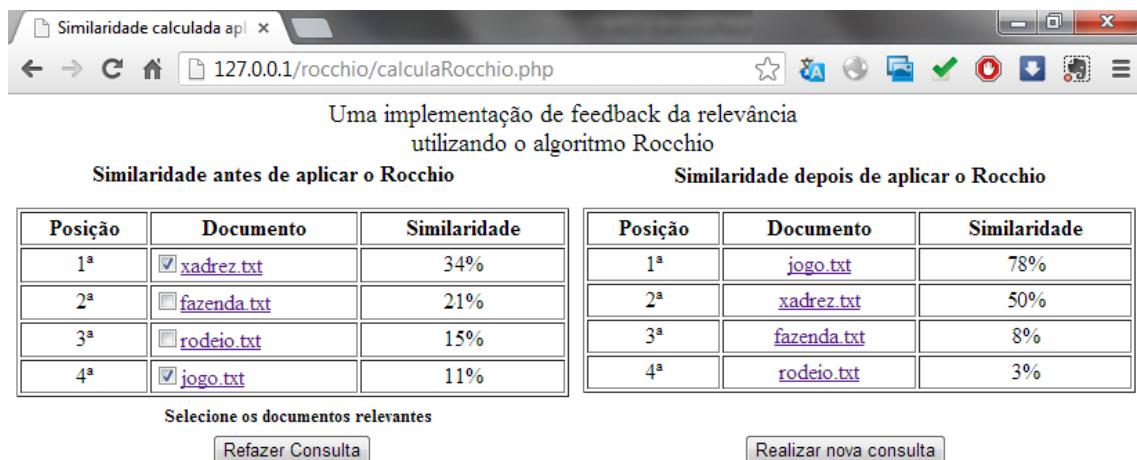
**Figura 2. Arquitetura da ferramenta desenvolvida.**

Foi utilizada a linguagem de programação PHP para implementar os componentes apresentados. Conforme sugeridos por Manning et al. (2008), foram utilizados os pesos  $\alpha = 1$ ,  $\beta = 0.75$  e  $\gamma = 0.15$ . Entretanto, a ferramenta permite a alteração desses e de outros parâmetros como o diretório contendo a coleção.

A Figura 3 apresenta uma consulta e uma coleção de documentos pré-processados. A Figura 4 mostra o resultado intermediário em que o *ranking* foi gerado apenas pela aplicação do modelo espacial vetorial e o *ranking* final, gerado após a aplicação do *feedback* da relevância. Perceba que o documento *jogo.txt*, que é do interesse do usuário, ficou na última posição com apenas 11% de similaridade em relação à consulta. Depois do *feedback*, o mesmo documento passou para a primeira posição do *ranking* junto de *xadrez.txt* que também é relevante, melhorando significativamente a qualidade do resultado. A precisão média [Manning et al. 2008] passou de 75 para 100%.



**Figura 3.** Exemplo destacando os documentos relevantes para a consulta do usuário.



**Figura 4.** Ranking original (à esquerda) e recalculado a partir do feedback da relevância do usuário (à direita).

A ferramenta também exibe os passos intermediários da contagem de frequências e dos cálculos da métrica  $tf \times idf$ , da similaridade e algoritmo Rocchio. É possível visualizar os vetores de cada documento e das consultas  $\vec{q}_0$  e  $\vec{q}_m$ . Entretanto, por restrições de espaço, essas informações não puderam ser apresentadas neste artigo.

## 5. Conclusão

Para ser utilizada em sala de aula, ainda são necessárias melhorias na interface gráfica, as quais serão realizadas até o início do próximo semestre letivo. Com a ferramenta finalizada, espera-se que os estudantes possam aprender os conceitos apresentados neste artigo com mais facilidade.

## Referências

- Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.
- Manning, C. D.; Raghavan, P. and Schutze H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- Rocchio, J. J. (1971). Relevance feedback in information retrieval. In Salton, G. (Ed.), *The Smart Retrieval System – Experiments in Automatic Document Processing*, Prentice Hall., p. 313 -323.
- Salton, G. and Buckley, C (1988). Term-weighting approaches in Automatic Retrieval. In *Information Processing & Management*, 24(5), p. 513–523.

# Uma implementação do algoritmo Naïve Bayes para classificação de texto

Giancarlo Lucca, Igor A. Pereira, André Prisco, Eduardo N. Borges

Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)  
Rio Grande – RS – Brasil

{giancarlo.lucca, igor.pereira, andre.prisco, eduardoborges}@furg.br

**Abstract.** This paper present a text classification tool based on the Naïve Bayes algorithm. We have described some basic concepts about textual classification in the Information Retrieval area, the algorithm chosen, an example of use and the architecture of our tool.

**Resumo.** Este artigo apresenta uma ferramenta para classificação de texto baseada no algoritmo Naïve Bayes. São descritos alguns conceitos básicos sobre classificação textual na área Recuperação de Informações, o algoritmo escolhido, um exemplo de utilização e a arquitetura da ferramenta.

## 1. Introdução

Uma das técnicas de mineração de dados amplamente utilizada é a classificação de dados. A classificação consiste no processo de encontrar, através de aprendizado de máquina, um modelo ou função que descreva diferentes classes de dados [Han e Kamber 2006]. O objetivo da classificação é rotular, automaticamente, novas instâncias da base de dados com uma determinada classe aplicando o modelo ou função “aprendidos”. Este modelo é baseado no valor dos atributos das instâncias de treinamento. Diversos classificadores foram propostos nos últimos anos. Alguns utilizam árvores de decisão para rotular registros. CART [Breiman et al. 1984] e C4.5 [Quinlan 1993] são exemplos bem conhecidos. Outros algoritmos se baseiam em redes neurais artificiais, modelos probabilísticos (bayesianos) ou em regras [Mitchell 1997].

A classificação pode ser especializada na categorização textual, que consiste na organização de documentos em tópicos preestabelecidos. Esta categorização tem diversas aplicações na área de Recuperação de Informação, tais como detecção de SPAM, organização automática de e-mails, identificação de páginas com conteúdo adulto e detecção de expressões multipalavras [Manning et al. 2008].

Dado um conjunto de  $j$  classes  $C = \{c_1, c_2, \dots, c_j\}$  e outro de  $i$  documentos  $D = \{d_1, d_2, \dots, d_i\}$  descritos por um espaço multidimensional  $X$  composto pelas palavras ou termos que aparecem em toda a coleção, o algoritmo aprende uma função de classificação  $f: X \rightarrow C$  que mapeia os documentos nas classes.

## 2. O Algoritmo Naïve Bayes

Implementado por ferramentas como MALLET, Apache Mahout e NLTK<sup>1</sup>, Naïve Bayes computa a probabilidade  $P(c|d)$  de um documento pertencer a uma determinada classe a partir da probabilidade a priori  $P(c)$  de um documento ser desta classe e das

---

<sup>1</sup> <http://mallet.cs.umass.edu>, <http://mahout.apache.org> e <http://nltk.org>, respectivamente.

probabilidades condicionais  $P(t_k|c)$  de cada termo  $t_k$  ocorrer em um documento da mesma classe. O objetivo do algoritmo é encontrar a melhor classe  $C_{map}$  para um documento maximizando a probabilidade a posteriori conforme Equação 1, onde  $n_d$  é o número de termos no documento  $d$ .

$$C_{map} = \arg \max_{c \in C} P(c|d) = \arg \max_{c \in C} P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \quad (1)$$

Para evitar o *underflow* de ponto flutuante, o produto das probabilidades é substituído pela soma dos logaritmos das probabilidades, resultando no algoritmo apresentado na Figura 1. A função de treinamento extrai o vocabulário da coleção de documentos  $D$ . A seguir é calculado um vetor de probabilidades a priori dividindo o número de documentos de cada classe pelo tamanho da coleção. Ao final do treinamento é estimada uma matriz de probabilidades condicionais através da frequência relativa dos termos nos documentos que pertencem a uma determinada classe. Para evitar que essa estimativa seja nula para uma combinação de termo e classe que não ocorra na coleção de treinamento, é usada uma suavização de Laplace [Manning et al. 2008] que incrementa cada contagem. A função de classificação recebe como parâmetros, além do documento de teste, o conjunto de classes, o vocabulário e as probabilidades estimadas no treinamento. Para cada classe, a probabilidade a posteriori é calculada somando o logaritmo da probabilidade a priori com os logaritmos das probabilidades condicionais de cada termo do documento de teste. O documento então é rotulado com a classe  $c$  que obtiver a maior probabilidade a posteriori.

```

TRAINMULTINOMIALNB(C, D)
1  V ← EXTRACTVOCABULARY(D)
2  N ← COUNTDOCS(D)
3  for each c ∈ C
4  do  $N_c \leftarrow \text{COUNTDOCSINCLASS}(D, c)$ 
5     $prior[c] \leftarrow N_c/N$ 
6     $text_c \leftarrow \text{CONCATENATETEXTOFTODOSDOCSENCASAS}(D, c)$ 
7    for each t ∈ V
8    do  $T_{ct} \leftarrow \text{COUNTTOKENSOFTERM}(text_c, t)$ 
9    for each t ∈ V
10   do  $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_t T_{ct}+1}$ 
11  return V, prior, condprob

APPLYMULTINOMIALNB(C, V, prior, condprob, d)
1  W ← EXTRACTTOKENSFROMDOC(V, d)
2  for each c ∈ C
3  do  $score[c] \leftarrow \log prior[c]$ 
4    for each t ∈ W
5    do  $score[c] += \log condprob[t][c]$ 
6  return  $\arg \max_{c \in C} score[c]$ 

```

**Figura 1. Pseudocódigo do algoritmo Naïve Bayes [Manning et al. 2008].**

A Tabela 1 apresenta um exemplo de classificação textual executado na ferramenta para validação da implementação. Os documentos representam um conjunto de manchetes, extraídas do jornal Diário Catarinense, pré-processadas e rotuladas com uma das seguintes classes: cultura, esportes e policial. Para facilitar a visualização, apenas algumas palavras-chave (termos) foram apresentadas. Uma vez aprendida a função de classificação dos documentos de treinamento, o modelo é aplicado sobre o documento de teste.

Para estimar a classe do documento 9 são calculadas as probabilidades a priori  $P(\text{Cultura}) = 3/8$ ,  $P(\text{Esportes}) = 3/8$ ,  $P(\text{Polícia}) = 2/8$ . Após, o algoritmo calcula as

probabilidades condicionais da seguinte forma. As frequências 27, 24 e 18 representam o número de termos por classe no conjunto de treinamento e 67 é o tamanho do vocabulário.

$$P(\text{clubes}|\text{Cultura}) = P(\text{catarinenses}|\text{Cultura}) = \frac{0 + 1}{27 + 67} = \frac{1}{94}$$

$$P(\text{cultural}|\text{Cultura}) = \frac{1 + 1}{27 + 67} = \frac{2}{94}$$

$$P(\text{clubes}|\text{Esportes}) = P(\text{catarinenses}|\text{Esportes}) = \frac{1 + 1}{24 + 67} = \frac{2}{91}$$

$$P(\text{cultural}|\text{Esportes}) = \frac{0 + 1}{24 + 67} = \frac{1}{91}$$

$$P(\text{clubes}|\text{Polícia}) = P(\text{catarinenses}|\text{Polícia}) = P(\text{cultural}|\text{Polícia}) = \frac{0 + 1}{18 + 67} = \frac{1}{85}$$

**Tabela 1. Termos que compõe cada documento distribuídos por classe.**

Classe	d	Termos contidos nos documentos
Cultura	1	brinquedos criativos estimulam encantam criancas pais summer balneario
	2	prefeitura camboriu abre inscricoes concurso rainha princesas camboriu festa ru
	3	concurso cultural vai batizar filhotes zoo beto carrero world
Esportes	4	desafio estrelas pilotos famosos estarao em penha
	5	apos entrega laudos clubes catarinenses esperam liberacao estadios
	6	presidente confederacao brasileira de tenis acreditamos nesta nova geracao
Polícia	7	pais chamam policia encontrar drogas escondidas quarto filho anos timbo
	8	prefeita luzia recebe visita comandante batalhao policia militar
?	9	clubes catarinenses clubes cultural

Por fim, são calculadas as probabilidades a posteriori para cada classe.

$$P(\text{Cultura}|d_9) = \log(P(\text{Cultura})) + 2 \log(P(\text{clubes}|\text{Cultura})) + \log(P(\text{catarinenses}|\text{Cultura})) \\ + \log(P(\text{cultural}|\text{Cultura}))$$

$$P(\text{Cultura}|d_9) = \log\left(\frac{3}{8}\right) + 2 \log\left(\frac{1}{94}\right) + \log\left(\frac{1}{94}\right) + \log\left(\frac{2}{94}\right) \cong -8,02$$

Seguindo o mesmo raciocínio para as demais classes, as probabilidades a posteriori são  $P(\text{Esportes}|d_9) \cong -7,36$  e  $P(\text{Polícia}|d_9) \cong -8,32$ . Assim, o documento de teste é rotulado com a classe Esportes porque obteve-se a maior estimativa.

### 3. Implementação

O código-fonte foi desenvolvido na linguagem Java e está coberto principalmente pela classe *NaiveBayes* descrita pelos atributos e métodos apresentados na Figura 2. A maioria dos métodos é análoga às funções apresentadas na Figura 2. Necessitam explicações adicionais o atributo *vetClasses* que armazena o conjunto de classes e *listaTextos* que armazena a coleção. O método *buscaTextos* é um procedimento que percorre o sistema de arquivos investigando os diretórios configurados como classes, analisando os documentos contidos nesses diretórios e chamando a função *extractVOCABULARY* internamente. *imprimeInfos* é um método utilizado como teste que exibe informações como número de classes, textos, vocabulário, etc. *indiceVocabulario* recebe um termo e retorna o índice equivalente no vocabulário.

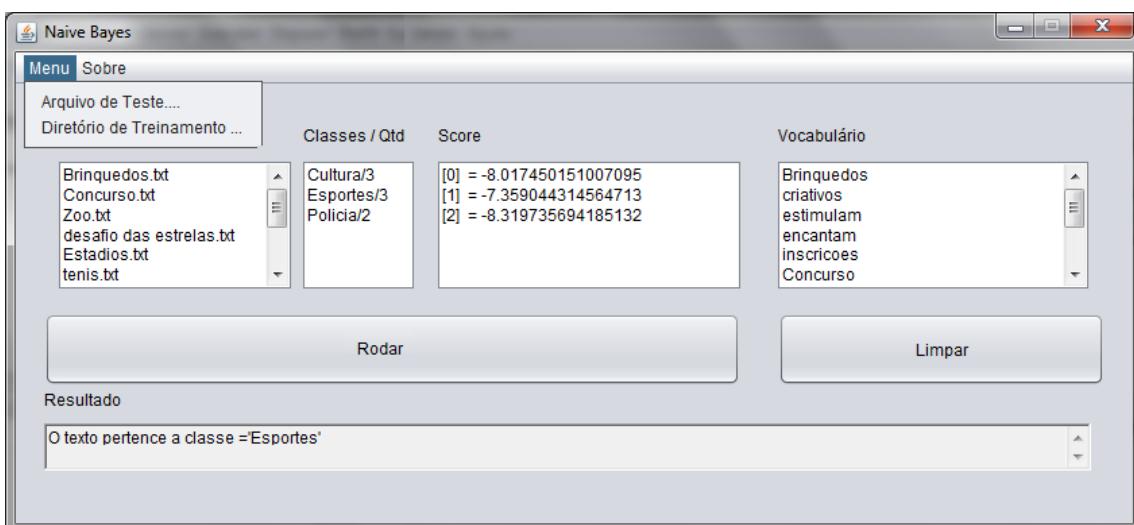
A Figura 3 apresenta a interface gráfica da ferramenta que destaca os documentos de treinamento, o número de documentos por classe, a probabilidade a posteriori do documento de teste pertencer a cada classe (*Score*) e as palavras que compõem o vocabulário. Por fim, a classe predita é apresentada.

```

NaiveBayes
-ListaVocabulario: ArrayList<String>
-vetClasses: File[]
-listaTextos: ArrayList<File>
-condProb: double[][][]
-prior: double[][]
NaiveBayes(arquivo_teste:File)
-extractVOCABULARY(um_texto:File): ArrayList<String>
-buscaTextos(): void
-contClasses(): int
-countDocs(): int
-countDocsInClass(uma_classe:File): int
-ConcatenateTextOfAllDocsInClass(uma_classe:File): ArrayList<String>
-imprimeInfos(): void
-countTokensOfTerm(listaTermos:ArrayList): int[]
-trainMultinomialNBC(): void
-argMax(vetScore:double[]): int
-indiceVocabulario(um_termo:String): int
-applyMultinomialNB(um_arquivo:File): void

```

**Figura 2.** Classe representando o algoritmo Naïve Bayes.



**Figura 3.** Interface gráfica da ferramenta proposta.

#### 4. Conclusões e trabalhos futuros

A ferramenta desenvolvida será utilizada para apoiar o ensino da disciplina Sistemas de Informações Avançados oferecida pelo Centro de Ciências Computacionais da FURG. Ainda é necessário codificar determinadas melhorias como os recursos de interface. Pretende-se diferenciar a ferramenta proposta de outras implementações evidenciando os cálculos dos passos internos do algoritmo quando o usuário seleciona determinada linha do pseudocódigo, facilitando o aprendizado dos estudantes.

#### Referências

- Breiman, L.; Friedman, J.; Olshen, R. and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth and Brooks.
- Han, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2nd ed.
- Manning, C. D.; Raghavan, P. and Schutze H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- Mitchell, T. (1997). *Machine Learning*. McGraw Hill.
- Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers.