

GroupShare: Distribuição e Gerência de Conteúdo em Grupos de Interesse para Dispositivos Móveis

Elisa Mannes, Fernando Gielow, Paulo Ferreira, Aldri Santos, Carmem Hara

¹Departamento de Informática - Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 - 81.531-980 – Curitiba – PR – Brasil

{elisam, fhg07, phkf07, aldri, carmem}@inf.ufpr.br

Abstract. This paper introduces an application for multimedia content sharing, focusing on the availability and management of the content between mobile devices. The application, called GroupShare, groups the devices by the interest in shared data, originating groups of interest. The content is shared in a P2P fashion, with a coordinator responsible for sharing content and its location. It also answers requests from clients and manages the location of replicas. GroupShare is developed in J2ME and communicates via Bluetooth channels.

Resumo. Esse artigo propõe um aplicativo para o compartilhamento de conteúdo multimídia, focando na disponibilidade e na comunicação direta entre dispositivos móveis. O aplicativo, chamado de GroupShare, agrupa os dispositivos por interesse no conteúdo, formando grupos de interesse. O conteúdo é compartilhado em uma abordagem P2P, sendo que um coordenador gerencia o armazenamento e a localização do conteúdo compartilhado. GroupShare é desenvolvido em J2ME, e comunica-se por meio de canais Bluetooth.

1. Introdução

As atuais tecnologias de processamento e de comunicação disponíveis em dispositivos móveis, como celulares e *tablets*, permitem que as aplicações utilizem uma arquitetura distribuída e descentralizada. Além disso, o uso de comunicação sem fio permite que uma nova categoria de aplicações sejam oferecidas, tais como o compartilhamento de conteúdo entre os próprios dispositivos móveis, sem a necessidade de um servidor central para a gerência dos dados compartilhados. Essas redes descentralizadas, tais como as redes *ad hoc* móveis (MANET) [Chlamtac et al. 2003] e as redes *peer-to-peer* (P2P) [Ding et al. 2005] são interessantes porque o compartilhamento de conteúdo entre os dispositivos pode ser feito de forma fácil e rápida.

Contudo, tais redes necessitam de abordagens que considerem as características específicas dos dispositivos, como a mobilidade e a escassez de recursos. Nesse contexto, este trabalho apresenta o GroupShare, um aplicativo distribuído para o compartilhamento de conteúdo multimídia, como vídeos, músicas e documentos, entre dispositivos móveis. Diversas aplicações podem se beneficiar do Groupshare, tais como aplicações de troca de idéias em eventos, divulgação e pedido de carona, divulgação de promoções em centros comerciais e compartilhamento de arquivos. O GroupShare é inspirado no aplicativo NotePals [Davis et al. 1999], e permite que conteúdos sejam compartilhados diretamente entre dispositivos por meio de canais *Bluetooth*. Os dispositivos são organizados em grupos, criados a partir do interesse em conteúdos distintos. O GroupShare emprega uma forma

simples de replicação baseada em [Vallur et al. 2008], que permite o balanceamento de carga entre os participantes e se mostra apropriada para ambientes de rede dinâmicos.

O artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 introduz o GroupShare. A Seção 4 descreve as ferramentas utilizadas na implementação do aplicativo e a Seção 5 apresenta o seu funcionamento. Por fim, a Seção 6 apresenta a conclusão e trabalhos futuros.

2. Trabalhos relacionados

[Davis et al. 1999] apresenta uma aplicação para o compartilhamento de notas entre dispositivos móveis, por meio da disponibilização em um servidor com acesso à Internet. Porém, os dispositivos atuais possuem acesso à Internet diretamente pelo dispositivo e o uso de um computador para a troca de anotações se torna obsoleto. Nesse caso, uma nova abordagem para o compartilhamento de conteúdo focando na disponibilidade do dado e na interação entre os dispositivos é interessante. A disponibilidade é geralmente obtida através de técnicas de replicação de dados. Em [Vallur et al. 2008] foi proposto um modelo de replicação para MANETs, chamado REALM, em que as cópias dos dados são replicadas próximas aos clientes que mais as acessam. Essa técnica de replicação acompanha um mecanismo de predição de particionamento da rede. Com base nesses trabalhos, é proposta uma nova aplicação para o compartilhamento de conteúdo, baseada em grupos de interesse. Para aumentar o desempenho do grupo, é proposta a replicação do conteúdo entre os dispositivos, baseada na quantidade de acessos. Essa replicação, apesar de simples, fornece ao sistema um melhor balanceamento de consultas entre os dispositivos, além de ajudar na economia de recursos, como energia.

3. GroupShare

O GroupShare é uma aplicação que tem por objetivo o compartilhamento de conteúdo diretamente entre os dispositivos móveis. Para isso, eles são organizados em grupos, formados a partir do interesse em conteúdos específicos. Assim, a rede é formada por vários grupos de interesse, em que cada grupo é responsável pelos seus conteúdos. Dentro de cada grupo, seu criador tem a responsabilidade de indexar o conteúdo disponível e a sua localização. Ao visualizar um conteúdo do grupo, o dispositivo deve consultar o coordenador, que retorna a localização do conteúdo. O dispositivo inicia uma conexão diretamente com o dono do conteúdo e a comunicação acontece diretamente entre os dois. Essa abordagem é semelhante à encontrada nas abordagens de P2P centralizado, como o Napster [Ding et al. 2005], em que um computador é responsável por catalogar o conteúdo da rede, além de responder consultas sobre a localização de determinado conteúdo. A diferença está na quantidade de dados que o coordenador deve gerenciar. Enquanto que no Napster o coordenador é responsável por todo o conteúdo de uma rede, no GroupShare o coordenador é responsável somente pelo conteúdo de seu grupo. Além disso, o GroupShare provê uma forma de replicação de dados, visando o balanceamento de carga entre os dispositivos. Para isso, o coordenador analisa a quantidade de consultas realizadas em cada conteúdo, e decide se o conteúdo precisa ser replicado.

3.1. Especificação

No GroupShare os dispositivos são servidores quando fornecem conteúdo para outros clientes, e clientes quando o consomem de outros servidores. Ainda nesse contexto, o

coordenador é responsável por um grupo de dispositivos com interesse em um mesmo conteúdo, e deve indexar e localizar o conteúdo disponível no grupo. Os clientes se comunicam diretamente com o dono do conteúdo após a descoberta da localização. A replicação acontece quando a quantidade de solicitações atinge um limiar previamente estabelecido. As principais funcionalidades do GroupShare são apresentadas a seguir.

Criação dos grupos: Os grupos são criados por qualquer dispositivos que deseja compartilhar conteúdo como vídeos e músicas. Ele deve criar um grupo com um nome que sugira o compartilhamento de tal conteúdo, e a partir de então, esse dispositivo se torna o dono do grupo e seu coordenador. Outros dispositivos interessados em compartilhar conteúdo sobre o mesmo tema se juntam ao grupo, formando uma rede de compartilhamento.

Compartilhamento de conteúdo: Todos os dispositivos da rede podem compartilhar conteúdo com os outros. Para isso, ele deve enviar ao coordenador o nome do conteúdo que ele deseja disponibilizar. O coordenador guarda essa informação e mantém uma tabela atualizada sobre o conteúdo do grupo. Os dispositivos sempre avisam a inclusão e exclusão de conteúdos ao coordenador.

Visualização de conteúdo: Somente os participantes de um grupo podem visualizar o conteúdo compartilhado por esse grupo. O coordenador, que possui o índice de conteúdo da rede, responde às requisições de localização do conteúdo. Ao receber a resposta com a localização do conteúdo solicitado, o cliente requisita o conteúdo diretamente ao dispositivo que o possui fisicamente, estabelecendo uma conexão ponto a ponto.

Saída de dispositivos do grupo: Quando um dispositivo sai do grupo, ele avisa ao coordenador que seu conteúdo não está mais disponível. Se um coordenador decidir deixar o grupo, ele deve também replicar a sua tabela de conteúdo e localização para outro dispositivo no grupo, escolhido aleatoriamente. Esse dispositivo se tornará o coordenador do grupo, e os participantes devem se juntar novamente ao grupo.

Replicação: Ao atingir um limite de consultas pré-estabelecido na rede, o coordenador dispara o processo de replicação. Esse processo necessita que o dono do conteúdo que informe ao coordenador um outro dispositivo para replicar o conteúdo. O dispositivo escolhido deve avisar o coordenador que ele também pode atender às requisições desse conteúdo, e dessa forma as requisições são balanceadas entre as réplicas.

Sincronização de réplicas: Visto que é necessário que as réplicas estejam sempre consistentes com a réplica principal, o dispositivo dono do conteúdo, ao atualizá-lo, deve perguntar ao coordenador quais os outros dispositivos que precisam atualizar seu conteúdo. Deste modo, o conteúdo é atualizado nos dispositivos que possuem réplicas.

4. Implementação do aplicativo

O GroupShare¹ foi desenvolvido em Java 2 Micro Edition (J2ME), apropriada para dispositivos com recursos escassos. Além disso, a máquina virtual Java é comumente encontrada nos celulares, tornando o GroupShare portável entre os dispositivos. A comunicação é feita pelo protocolo de comunicação *Bluetooth*, por ser amplamente encontrado nos dispositivos móveis e projetado para consumir pouca energia. Utilizou-se duas *Application Programming Interface* (API): o projeto Marge, um *middleware* para o gerenciamento da

¹Disponível em <http://www.nr2.ufpr.br/groupshare/gp.html>

conexão *Bluetooth*, e o Kuix para a construção das interfaces gráficas do aplicativo. Para emular o funcionamento das operações do GroupShare, utilizou-se o emulador *Wireless Toolkit*, na versão 2.5.2. A Figura 1 ilustra o menu de navegação principal, com opções para grupos e conteúdos. Os menus permitem criar, procurar e entrar em um grupo previamente selecionado. Pode-se também criar novos conteúdos e publicá-los no grupo, além de atualizá-lo. Essas opções são apresentadas na Figura 2.

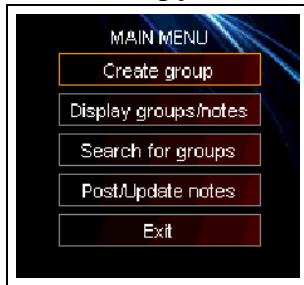


Figura 1. Menu inicial



Figura 2. Opções para o conteúdo

A opção de procura por grupos apresenta como resultado os grupos disponíveis dentro do raio de alcance de comunicação do dispositivo. Os resultados da busca acompanham a opção de entrar no grupo. Se o usuário listar os grupos aos quais ele pertence, uma opção de deixar o grupo é apresentada. Um asterisco ao lado de um grupo representa os grupos em que o dispositivo é o dono. Ao compartilhar algum conteúdo com o grupo, o coordenador deve indexar esse conteúdo, para que os dispositivos sejam capazes de identificar a localização deste conteúdo. Ao procurar por conteúdos, o cliente recebe uma lista com o conteúdo, que é emitida pelo coordenador. O GroupShare foi testado em aparelhos celulares com máquina virtual Java, MIDP2.0 e comunicação *Bluetooth*. O GroupShare não funciona em dispositivos com versões de *software* anteriores às mencionadas.

5. Conclusão

Este artigo apresentou o GroupShare, um aplicativo para o compartilhamento de conteúdo entre dispositivos móveis. O GroupShare divide os dispositivos em grupos, baseados no interesse ao conteúdo. Cada grupo possui um coordenador, que é responsável por indexar o conteúdo disponibilizado. Para aumentar a disponibilidade e balancear a carga na rede, o GroupShare replica o conteúdo baseando-se na quantidade de acessos ocorridos. Como trabalhos futuros, pretende-se empregar um roteamento multissalto para a interligação dos dispositivos, que torna o sistema independente sem a necessidade de um coordenador.

Referências

- Chlamtac, I., Conti, M., and Liu, J. J. (2003). Mobile ad hoc networking: imperatives and challenges. *Ad Hoc Networks*, 1(1):13–64.
- Davis, R. C., Landay, J. A., Chen, V., Huang, J., Lee, R. B., Li, F. C., Lin, J., Morrey, III, C. B., Schleimer, B., Price, M. N., and Schilit, B. N. (1999). Notepals: lightweight note sharing by the group, for the group. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 338–345, New York, NY, USA. ACM.
- Ding, C. H., Nutanong, S., and Buyya, R. (2005). *Peer-to-Peer Networks for Content Sharing*, chapter 2, pages 28 – 65. Idea Group Inc., Hershey, PA, USA.
- Vallur, A., Gruenwald, L., and Hunter, N. (2008). Realm: Replication of data for a logical group based manet database. In *Proceedings of the 19th international conference on Database and Expert Systems Applications*, pages 172–185, Berlin, Heidelberg. Springer-Verlag.

Representação Gráfica de Documentos XML

Matheus K. Zanella, Deise de B. Saccò¹

Curso de Ciência da Computação - Universidade Federal de Santa Maria (UFSM) - Santa
Maria - RS - Brasil

{mzanella,deise}@inf.ufsm.br

Abstract. *XML is widely used for storing and exchanging information. However, the structure understanding of XML instances may become hard due to the size and the number of tags that a document can present. Besides, there is a lack of tools that enable viewing XML files in a graphical format. Therefore, the objective of this work is to implement a tool to generate e visualize the graphical representation of a XML document in a graph format, facilitating the understanding of the document structure and its corresponding visualization.*

Resumo. *A linguagem XML é muito utilizada para armazenamento e troca de informações. Entretanto, o entendimento da estrutura de uma instância XML pode se tornar difícil devido ao tamanho e ao número de tags que um documento pode apresentar. Além disso, existe uma carência de ferramentas que possibilitem a visualização de arquivos XML em um formato gráfico. Assim, o objetivo deste trabalho é implementar uma ferramenta que possibilite a representação e visualização gráfica de um documento XML em formato de grafo, facilitando o entendimento da estrutura do documento e a sua correspondente visualização.*

1. Introdução

A linguagem XML tem se tornado um padrão no intercâmbio de informações na Web. No entanto, para documentos XMLs grandes e com estruturas complexas, visualizar o documento apenas num editor de texto pode não ser suficiente para compreendê-lo. Para facilitar o seu entendimento, é conveniente utilizar um programa que gere a visualização deste documento em um formato gráfico. Existem alguns softwares que permitem esta visualização, dentre eles os softwares para edição e validação de documentos XML, tais como *XMLSpear* [2005] e *Liquid XML Studio* [2011]. O primeiro gera a visualização do documento em modo textual e também no formato de árvore, onde o usuário pode navegar através dos elementos. Porém, o usuário deve estar familiarizado com esta navegação, tendo em vista que deve-se expandir os nodos para que se acesse a informação desejada. O segundo, além das visualizações geradas pelo primeiro, oferece suporte à visualização gráfica de esquemas XSD. Documentos XSD são esquemas especificados na linguagem *XML Schema*, utilizados para validação de documentos XML. No entanto, este tipo de visualização apenas mostra a estrutura do documento; informações sobre o conteúdo do arquivos, por não estarem presentes em esquemas XSD, não podem ser visualizadas.

As ferramentas disponíveis analisadas são poucos utilizadas para visualizar graficamente documentos XML devido aos pontos negativos relatados. Assim, constatou-se a necessidade de um software estável, funcional, intuitivo e que gere visualizações gráficas de fácil entendimento dos documentos XML. Por isso, este trabalho propõe a implementação de um software para gerar visualizações gráficas de documentos XML em formato de grafo.

¹ Este trabalho foi parcialmente financiado por SESU/MEC (PET-Programa de Educação Tutorial) e FAPERGS (Auxílio Recém Doutor – Processo número 11/0748-6).

2. A Ferramenta *XMLGraph*

A ferramenta *XMLGraph* tem como finalidade a criação de uma visualização gráfica, em formato de grafo, de documentos XML. Para gerar a visualização do documento, o programa realiza a tradução da estrutura e conteúdo do arquivo XML para a linguagem de especificação de grafos DOT. Para realizar esta tradução, o documento XML é analisado e para cada tipo de nodo no arquivo XML é criado um nodo no grafo. A relação dos nodos no grafo obedecerá a mesma relação dos nodos no documento XML. Feita esta tradução, a ferramenta utiliza uma API, em Java, da ferramenta *GraphViz* [2011] para gerar a visualização do grafo. O programa *XMLGraph* também dá suporte à criação de subgrafos no documento DOT, garantindo que as informações sejam melhores organizadas.

A interface gráfica é composta por 5 menus e 3 abas. Um menu é utilizado para abertura e salvamento de arquivos, três menus são utilizados para o usuário personalizar o grafo gerado e o último menu é utilizado para auxiliar na utilização da ferramenta. A primeira aba é utilizada para visualização do documento XML carregado no programa, a segunda é utilizada para mostrar o documento DOT gerado e a última para visualizar o grafo gerado. As seções a seguir explicam a utilização da ferramenta.

2.1. Visualização do documento XML

A primeira aba é destinada a visualização do documento XML carregado pelo usuário. A figura 1 mostra a visualização de um documento XML carregado na ferramenta.

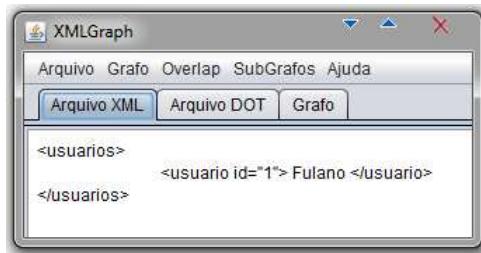


Figura 1. Visualização do documento XML.

2.2. Visualização do documento DOT

A segunda aba é destinada a visualização do documento DOT, gerado a partir do documento XML de entrada. A figura 2 mostra ao usuário o resultado da tradução do documento XML para a linguagem DOT.

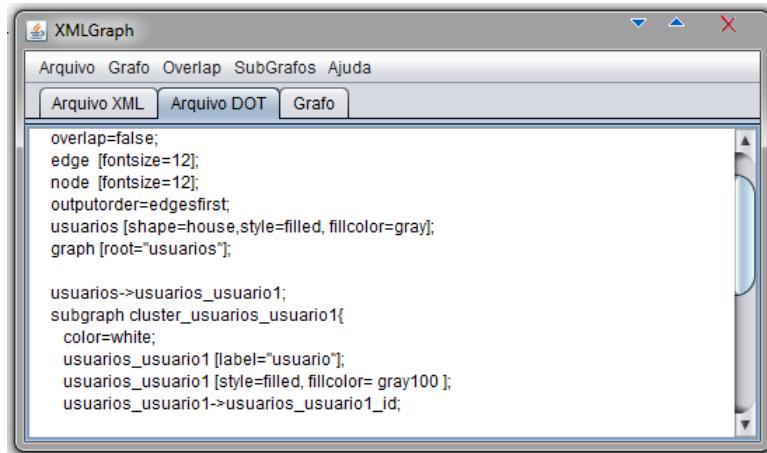


Figura 2. Visualização do documento DOT.

2.3. Opções para visualização do grafo

Os menus *Grafo*, *Overlap* e *SubGrafos* são utilizados para personalizar a geração do grafo. O menu *Grafo* permite alterar o modo de *layout* do grafo (DOT, DOT - LR, NEATO, FDP, SFDP, TWOPI e CIRCO). O menu *Overlap* permite alterar o método utilizado para tratamento de sobreposição de nodos (FALSE, SCALE e DEFAULT). O menu *SubGrafos* permite ao usuário definir a criação, ou não, de subgrafos quando o grafo é gerado.

Mais informações, sobre os modos de *layout* e opções para tratamento de *overlaps*, podem ser encontradas no manual da ferramenta *GraphViz*, disponível em: <http://graphviz.org/Documentation.php>.

2.4. Visualização do grafo

A terceira aba é destinada à visualização do grafo. A figura 3 mostra ao usuário o grafo gerado a partir do documento DOT da figura 2.

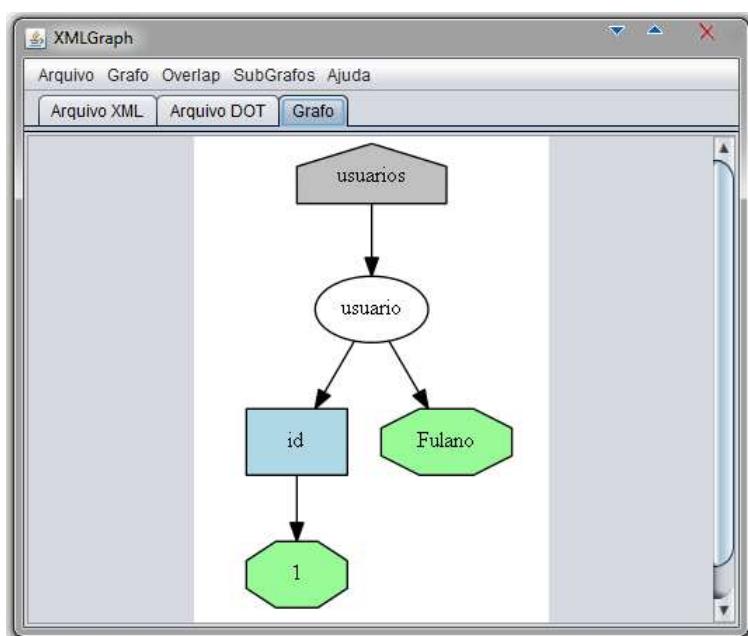


Figura 3. Visualização do grafo do documento DOT da figura 2.

3. Implementação

A ferramenta foi desenvolvida em linguagem Java e utiliza uma API em Java do software *GraphViz* para gerar a visualização do grafo na ferramenta. O programa possui dez classes, dentre as quais se destacam as quatro a seguir:

- Interface: contém todos os componentes da interface gráfica. Invoca métodos da classe *Documento* para carregamento do arquivo XML, obtenção do documento DOT e da imagem do grafo.
- Documento: classe onde o documento XML, o documento DOT e a imagem do grafo ficam salvos. Invoca métodos da classe *DocumentoDOT* para gerar o documento DOT e métodos da classe *GraphViz* para gerar a imagem do grafo.
- DocumentoDOT: classe utilizada para gerar o documento DOT a partir do arquivo XML de entrada. Seu principal método percorre recursivamente o documento XML e, assim, cria os nodos do grafo no documento DOT.

- GraphViz: esta classe, fornecida pela API em Java do *GraphViz*, fornece funções para integrar algumas funcionalidades do *GraphViz* na ferramenta *XMLGraph*. Seu principal método gera a imagem do grafo especificado em DOT.

Os códigos fonte e a aplicação estão disponíveis em <http://www.inf.ufsm.br/~mzanella/XMLGraph/>.

4. Comparação entre *XMLGraph*, *XMLSpear* e *Liquid XML Studio*

Os softwares *XMLSpear* e *Liquid XML Studio* são editores de arquivos XML e, por isso, são ferramentas bastantes completas que possuem métodos para criação e validação de documentos XML. Na parte de visualização gráfica do documento XML, estas ferramentas deixam a desejar. Nelas, é necessário que o usuário esteja familiarizado com a navegação em estruturas de árvores ou que conheça a linguagem *XML Schema* para poder entender e navegar na visualização gráfica gerada. Outro ponto negativo é que não é possível exportar a visualização gráfica para um arquivo de imagem.

O foco da ferramenta *XMLGraph* é a visualização gráfica do documento XML. Nela é possível visualizar o documento através de um grafo que pode ser personalizado pelo usuário através da interface. A ferramenta fornece opções para exportação da visualização gerada e também para salvamento do documento DOT. Outro ponto positivo é que o usuário, se tiver conhecimento da linguagem DOT, poderá salvar o documento DOT e realizar alterações próprias, além das oferecidas pela ferramenta, para personalizar ainda mais o grafo gerado e visualizá-lo no software *GraphViz*. Um ponto negativo é que a *XMLGraph* não oferece suporte à edição e validação do documento XML, sendo necessário que o usuário utilize outro software para realizar estas funções.

5. Considerações Finais

Testes com arquivos XML, de diferentes tamanhos e estruturas, e potenciais casos de erros, tais como: arquivo XML de entrada mal estruturado, elementos mistos e caractéres inválidos; já foram realizados e tratados pelos autores. Pela análise destes testes e pela comparação entre as funcionalidades, para geração de visualização gráficas de documentos XML, dos programas *XMLGraph*, *XMLSpear* e *Liquid XML Studio*, conclui-se que a ferramenta atingiu com sucesso seu principal objetivo de suprir as necessidades dos programas disponíveis existentes.

Para trabalhos futuros, sugere-se que seja suportada a opção para edição e validação do arquivo XML de entrada. Oferecer mais opções, suportadas pelo *GraphViz*, para criação do documento DOT, melhorando a personalização do grafo gerado. Por fim, realizar melhorias para geração de grafos menores que facilitariam o seu entendimento.

Referências

GraphViz (2011). Disponível em <http://www.GraphViz.org>. Acesso em agosto de 2011.

Liquid XML Studio 2011 (2011). Disponível em: <http://www.liquid-technologies.com/xml-studio.aspx>. Acesso em: setembro de 2011.

XMLSpear (2005). Disponível em: <http://www.donkeydevelopment.com/>. Acesso em: setembro de 2011.

Mineração de Dados Para Inferir Padrões Associados aos Fenômenos: *El Niño*, *La Niña* e Anos Neutros

Alexandre T. Lazzaretti¹, Vinicius P. Lima¹, José Mauricio Fernandes², Willington Pavan³, Josué Toebe⁴

¹Instituto Federal Sul-Riograndense (IFSul) - Passo Fundo, RS - Brasil

²Centro Nacional de Pesquisa de Trigo (CNPT) - Passo Fundo, RS - Brasil

³Universidade de Passo Fundo (UPF) - Passo Fundo, RS - Brasil

⁴Instituto Federal Riograndense (IFRS) - Sertão, RS - Brasil

alexandre.lazzaretti@passofundo.ifsul.edu.br, vini.p.lima@gmail.com, mauricio@cnpt.embrapa.br, pavan@upf.br, josue.toebe@sertao.ifrs.edu.br

Abstract. *The climate changes affecting many areas of human life, of which agriculture is heavily influenced. The occurrence of variations in climate may cause losses and damage in various agricultural sectors. Among these variations, El Niño, La Niña and Neutral years. Therefore, the prevision of this is an important factor. Thus, this paper shows an analysis of data mining algorithms in order to infer patterns and models that determine the occurrence of El Niño, La Niña and Neutral years phenomena.*

Resumo. *O clima interfere em várias áreas da vida humana, das quais a agricultura é fortemente influenciada. A ocorrência de variações climáticas podem ocasionar perdas e danos em vários setores agrícolas. Dentre essas variações, pode-se citar os fenômenos El Niño, La Niña e Anos Neutros. Portanto, a possibilidade de previsão desses fenômenos é um fator importante. Nesse sentido, esse trabalho mostra uma análise dos algoritmos de mineração de dados com o objetivo de inferir padrões e modelos que determinam a ocorrência dos fenômenos El Niño, La Niña e Anos Neutros.*

1. Introdução

O processo de descoberta de conhecimentos em banco de dados e consequentemente a mineração de dados, tornam, desde 1989, os grandes aglomerados de dados em conhecimento utilizável, tanto para administradores quanto pesquisadores.

Essas mudanças nos cenários, provocadas pelo avanço da tecnologia, afetou também uma das práticas mais antigas da humanidade, a observação do clima. As práticas agrícolas estão diretamente relacionadas às variáveis climáticas como por exemplo: temperatura, precipitação, etc. No entanto o clima e as estações não se comportam de maneira igual todos os anos. Um exemplo disso, são os fenômenos sazonais *El Niño* e *La Niña*, que são ocasionados pela mudança de temperatura, aquecimento ou resfriamento respectivamente, das águas tropicais do Pacífico. No caso de não ocorrer alteração na temperatura considera-se a existência de um fenômeno denominado “Anos Neutros”. A ocorrência desses eventos ocasiona mudanças nas correntes de ar do globo terrestre, mudando o comportamento do clima (Babkina, 2003).

Bases de dados climatológicas geram grandes massas de dados, os quais, dependendo do tipo de informação desejada, torna-se impossível extrair de maneiras convencionais. Dessa forma, seria interessante inferir, através de técnicas de mineração de dados, padrões de comportamento em bases de dados climatológicas associados aos fenômenos naturais *El Niño*, *La Niña* e Anos Neutros.

Devido a importância do tema existem trabalhos relacionados (STEINBACH *et. al.*, 2002; SHIKOUN *et. al.*, 2005) que buscam fazer a inferência de ocorrência de fenômenos naturais. Entretanto esses trabalhos não são específicos do local aqui proposto, pois o efeito dos fenômenos varia conforme a região do globo terreste, e não fazem a uso e análise de algoritmos de mineração de dados.

Nesse sentido, esse trabalho tem como objetivo apresentar um estudo de algoritmos de mineração de dados, buscando identificar os que apresentam melhores resultados de classificação dos fenômenos naturais.

2. Metodologia

Para a realização desse trabalho foi utilizado uma base de dados disponibilizada pelo IAPAR (Instituto Agronômico do Paraná). Essa base de dados é composta por dados diários observados de clima de 28 estações do estado do Paraná. Os dados correspondem ao período de 1980 a 2009. As variáveis de clima diárias disponíveis são: temperatura máxima, temperatura mínima, umidade relativa, precipitação e radiação solar. Essa base foi denominada “base original”. A informação dos anos de ocorrências dos fenômenos naturais foram obtidas a partir do Instituto Nacional de Pesquisa Espacial (INPE).

Foi necessário adotar um método que pudesse ser aplicado na comparação dos algoritmos selecionados para teste. O método escolhido é uma adaptação do descrito por Han e Kamber (2006), que apresentam as seguintes características: (*i*) *Acurácia*: é definida como a habilidade do algoritmo de classificar assertivamente novos dados quando aplicados ao modelo gerado; (*ii*) *Velocidade*: referente ao tempo que o algoritmo leva para apresentar os resultados; (*iii*) *Robustez*: refere-se a assertividade do modelo quanto aos dados incorretos, inexistentes ou com ruído; (*iv*) *Escalabilidade*: referente ao desempenho do algoritmo, ou o número de recursos computacionais usados, quando aplicados a grandes conjuntos de dados. (*v*) *Interpretabilidade*: refere-se ao nível de comprehensibilidade apresentado pelo modelo gerado.

Após o estabelecimento dos critérios de análise, necessitou-se, a partir dos dados da base original, gerar mais duas bases de dados. Isso foi necessário para que se pudesse testar todos os critérios estabelecidos. A segunda base foi gerada com uma granularidade anual com os dados agrupados mensalmente. Nesse caso, para cada mês do ano foram agrupados os dados de cada variável de clima. Para as variáveis temperatura mínima, temperatura máxima e umidade relativa foi feita uma média mensal. Já para as variáveis precipitação e radiação solar foi feito um somatório mensal. Nessa base, denominada “Base 01”, foi avaliado o critério de acurácia. Uma terceira base, denominada “Base 02”, foi gerada a partir da Base 01. Nessa base, aleatoriamente foram retirados alguns valores com o objetivo de avaliar o critério de robustez. Após as bases de dados terem sido criadas, e o método de comparação de algoritmos ter sido escolhido, elegeu-se a tarefa de mineração de dados. Devido ao potencial de utilidade

das bases de dados climáticas para a área de meteorologia e observando que a previsão de eventos climáticos são características marcantes, foi analisada uma tarefa de mineração de dados que pudesse classificar fenômenos a partir de dados fornecidos por estações meteorológicas, de modo a identificar os algoritmos de classificação mais adequados. Para tal, levou-se em consideração, principalmente a acurácia, haja visto que as outras características não apresentaram diferenças significativas.

Então, passou-se ao processo de execução e análise dos algoritmos de classificação. Para a execução foi utilizada a ferramenta WEKA (*Waikato Environment for Knowledge Analysis*), onde foram testados 64 algoritmos, de seis famílias (*Bayes, Functions, Lazy, Meta, Rules e Trees*) diferentes e seus resultados foram registrados e comparados para que se pudesse eleger os que obtivessem maiores probabilidades de retornar modelos de classificação úteis. Os resultados de acurácia foram organizados em tabelas para facilitar a compreensão e comparação. Foram avaliadas a interação de todas as variáveis climatológicas e individualmente cada uma.

3. Resultados e Discussão

A estimativa da acurácia foi feita utilizando um método oferecido pela ferramenta WEKA chamado de *cross-validation* (Santos & Azevedo, 2005). Este método apresenta maior confiabilidade, já que baseia o seu resultado em um conjunto de dez execuções do algoritmo selecionado, sendo que em cada uma destas execuções 90% dos dados oferecidos são utilizados como grupo de treinamento e os outros 10% restantes são aplicados ao modelo gerado excluindo-se a *class label* (variável que explica o fenômeno), garantindo a impossibilidade do algoritmo fazer uso da simples relação trivial já presente. Lembrando que, esses dados escolhidos para o grupo de treinamento são aleatórios diminuindo assim o risco de exceções. A acurácia do modelo é a porcentagem dos testes corretamente classificados pelo algoritmo. Considerando algoritmos com as maiores taxas de acurárias, igual ou maior que 70%, temos um conjunto de modelos que descrevem com relativo grau de assertividade os fenômenos de *El Niño*, *La Niña* e de anos neutros. A análise e comparação desses modelos pode contribuir de forma interessante ao conhecimento que se tem desses eventos climáticos, bem como mostrar a capacidade das técnicas de mineração de dados quando aplicados à fins meteorológicos. A seguir tem-se a Tabela 01 para comparação dos algoritmos que apresentaram os melhores resultados.

Tabela 01 - Algoritmos com maior acurácia.

Algoritmos	Acurácia	Robustez	Atributos Utilizados
<i>AdaBoostM1</i>	76,67%	60%	temperatura mínima de janeiro
<i>OneR</i>	70%	70%	radiação solar de novembro
<i>DecisionStump</i>	70%	70%	temperatura mínima de janeiro
<i>SimpleCart</i>	70%	70%	temperatura mínima de janeiro e radiação solar de novembro

Analizando os modelos gerados pelos algoritmos, verifica-se o o algoritmo *AdaBoostM1* leva em consideração temperatura mínima do mês de janeiro para explicar a ocorrência dos fenômenos, conforme mostra a Figura 1 (a). Já o algoritmo *OneR* leva em consideração a variável da radiação solar do mês de novembro, conforme é mostrado na Figura 1 (b). O algoritmo *DecisionStump* gera um modelo que leva em consideração a variável temperatura mínima do mês de janeiro, conforme mostra a Figura 1 (c). Por fim, o algoritmo *SimpleCart*, cujo modelo leva em consideração as

variáveis temperatura mínima de janeiro e a radiação do mês de novembro, isso é mostrado na Figura 1 (d).

<pre> class distributions tminjan <= 19.859043778801798 neutral El Niño La Niña 0.625 0.375 0.0 tminjan > 19.859043778801798 neutral El Niño La Niña 0.0 0.0 1.0 tminjan is missing neutral El Niño La Niña 0.5 0.3 0.2 weight: 0.85 </pre> <p style="text-align: center;">(a)</p>	<pre> rnov: < 18.193272297619053 -> El Niño < 20.596421967261897 -> neutral >= 20.596421967261897 -> La Niña </pre> <p style="text-align: center;">(b)</p>
<pre> ==== Classifier model (full training set) ==== Decision Stump classifications tminjan <= 19.859043778801798 : neutral tminjan > 19.859043778801798 : La Niña tminjan is missing : neutral class distributions tminjan <= 19.859043778801798 neutral El Niño La Niña 0.625 0.375 0.0 tminjan > 19.859043778801798 neutral El Niño La Niña 0.0 0.0 1.0 tminjan is missing neutral El Niño La Niña 0.5 0.3 0.2 </pre> <p style="text-align: center;">(c)</p>	<pre> ==== Classifier model (full training set) ==== CART Decision Tree tminjan < 19.859043778801798 rnov < 18.193272297619053: El Niño(7.0/1.0) rnov >= 18.193272297619053: neutral(14.0/2.0) tminjan >= 19.859043778801798: La Niña(6.0/0.0) </pre> <p style="text-align: center;">(d)</p>

Figura 1 - Modelo Inferidos pelos Algoritmos.

De forma resumida, os resultados mostram o uso de apenas dois atributos (média da temperatura mínima do mês de janeiro e a radiação solar acumulada do mês de novembro). Conforme consulta com especialista¹ da área pode-se verificar que a radiação solar do mês de novembro é um fator que indica a presença dos fenômenos, pois é o inverso da precipitação, ou seja, em anos que se obteve maior radiação são associados ao fenômeno *La Niña*. Estes atributos podem ter sido suficientes para classificar corretamente os anos de *La Niña*, no entanto apresentaram deficiências na classificação dos fenômenos *El Niño* e de anos neutros. Por fim, acredita-se que os modelos inferidos possam ser usados associados a prognósticos de clima no sentido de prever fenômenos naturais.

4. Referências

- BABKINA, A. M. El Niño: Overview and Bibliography. Nova York, 2003.
- HAN, Jiawei; KAMBER, Micheline. Data Mining: Concepts and Techniques. 2^a ed. San Francisco: Morgan Kaufmann, 2006.
- SANTOS, M. F. & AZEVEDO, C. (2005). Data Mining – Descoberta de Conhecimento em Base de Dados. FCA: Lisboa.
- SHIKOUN, N.; H. EL-BOLOK; M. A. ISMAIL. Climate Change Prediction Using Data Mining. *IJICIS*, Vol. 5, No. 1, July 2005
- STEINBACH, MICHAEL; TAN, PANG-NING; KUMAR, VIPIN; STEVEN KLOOSTER; CHRISTOPHER POTTER. Temporal Data Mining for the Discovery and Analysis of Ocean Climate Indices. NASA. 2002.

¹ Dr. Gilberto Cunha - Agrometeorologista Centro Nacional de Pesquisa de Trigo - Passo Fundo - RS

CMAP: Geração e Representação de Equivalências entre Documentos XML, Ontologia e Modelo Relacional

Douglas Negrini¹, Deise de Brum Saccò²

¹ Curso de Ciência da Computação – Universidade Federal de Santa Maria (UFSM)

² Departamento de Eletrônica e Computação – Universidade Federal de Santa Maria
(UFSM)

Santa Maria – Rio Grande do Sul – Brasil

{deise, negrini}@inf.ufsm.br

Abstract. This paper presents a proposal of generation and representation of equivalences between concepts presents in XML documents, ontology and relational model, within the scope of the framework X2Rel. This framework proposes the storage of XML documents belonging to a same domain but with different structures, in a unified relational database, generated with the assistance of ontologies.

Resumo. Este artigo apresenta uma proposta de geração e representação de equivalências entre conceitos presentes em documentos XML, ontologia e modelo relacional, dentro do âmbito do framework X2Rel. Tal framework propõe o armazenamento de documentos XML, pertencentes a um mesmo domínio porém com diferentes estruturas, em uma base de dados relacional unificada, gerada com o auxílio de ontologias.

1. Introdução

Nos dias atuais a posse de informação e a agilidade na sua manipulação podem ter um maior valor para certas empresas do que seus próprios recursos físicos e produtos. O meio digital aplicado a este contexto permite que se ganhe em velocidade e redução de custos graças à automatização de processos.

Com a popularização da internet, o acesso a diferentes fontes de informação está se tornando cada vez mais comum. Junto com essa popularização, veio se agravando o problema da heterogeneidade na representação da informação, obtendo-se dados sem padronização, com estruturas irregulares e bastante heterogêneas (MELLO, 2007). Para solucionar este problema, surgiu um novo padrão para representação de informações, a linguagem XML (eXtensible Markup Language) (W3C XML, 2011). Por mais que esta linguagem tenha criado um padrão para representação de informação estruturada, cada sistema utiliza a estrutura que for mais conveniente e eficiente para si, mesmo se tratando de informações de um mesmo domínio.

Uma proposta apresentada por Saccò (2005) e implementada por Mello (2007) é realizar a unificação das diferentes estruturas dos arquivos XML com o auxílio de ontologias, e gerar uma ontologia global que represente as estruturas iniciais. Andrade (2010) apresentou uma metodologia para transformar este modelo unificado dos arquivos XML em um modelo de banco de dados relacional, tornando possível assim,

armazenar todas essas informações em uma única base de dados. A etapa subsequente deste processo consiste em realizar a inserção das informações contidas nos documentos XML no modelo de banco de dados relacional gerado. Desta forma, se obtém a representação de toda a informação de um modo padronizado e, consequentemente, com um menor custo de manipulação. Para realizar esta injeção de dados no modelo relacional, foi verificada a necessidade de realizar um mapeamento do processo de integração, permitindo a identificação das equivalências entre os conceitos presentes nos arquivos XML e no modelo relacional gerado. Dentro deste cenário, este trabalho apresenta uma proposta para geração e representação de equivalências entre conceitos de documentos XML, ontologia e modelo relacional usados no processo citado.

De modo a estruturar o texto, a seção 2 apresenta trabalhos relacionados à geração de equivalências e a seção 3 apresenta o *framework* X2Rel (*XML to Relational*). A seção 4 apresenta a ferramenta proposta, e a seção 5 explica detalhes a respeito da implementação. Por fim, na seção 6 são descritos os trabalhos futuros e a conclusão.

2. Trabalhos Relacionados

A detecção de similaridades semânticas entre conceitos mostra-se uma tarefa árdua devido ao alto grau de ambiguidade que pode ser encontrado nos meios utilizados para sua representação. Além disso, similaridades podem variar dependendo do contexto aos quais estes conceitos estão inseridos, possibilitando que dois conceitos sejam considerados similares ou equivalentes em um cenário, e diferentes em outro.

Frozza (2007) propõe uma metodologia para determinar equivalências semânticas entre esquemas GML, na qual o processo é baseado em métricas de similaridade para buscar equivalências entre pares de elementos. Um conjunto de métricas foi definido para os diferentes quesitos analisados durante a verificação de equivalências, tais como equivalência de nomes, equivalência de tipos, equivalência de cardinalidades, entre outros. Estas métricas indicam o quanto um par de elementos é equivalente dentro deste quesito. Após, é feita a unificação destas métricas de modo que um índice de similaridade entre 0 e 1 seja apresentado para o par de elementos citado. O autor definiu um *threshold* padrão de 0.75 para a métrica gerada, de forma que todos os pares de elementos que possuírem a métrica maior que o valor apresentado, são considerados equivalentes. Esta proposta permite que equivalências semânticas sejam geradas de forma que certa porcentagem das regras possa não ser atendida, ou até mesmo que uma regra seja atendida de forma parcial.

Já Vidal e Vilas Boas (2002) apresentaram uma abordagem para geração de correspondências entre XML *Schemas* baseada no uso de assertivas de correspondência. Tais assertivas consistem em um conjunto de regras que devem ser atendidas para que conceitos sejam considerados equivalentes semanticamente, mesmo possuindo diferenças de nomenclatura e representação, entre outros.

A proposta apresentada por Frozza não é uma abordagem adequada para este trabalho, visto que as equivalências serão geradas entre estruturas criadas durante um processo de unificação. Se fossem usadas métricas de similaridade, estas seriam geradas com base nas regras do processo de unificação, o que tornaria incoerente que alguma delas não fosse obedecida. Desta forma, o ponto forte da proposta não seria usado, já que o único valor de *threshold* aceitável seria 1.

Assim sendo, este trabalho usa uma abordagem baseada em assertivas de correspondência, assim como propuseram Vidal e Vilas Boas (2002). Foi definido um conjunto de regras, o qual valida a equivalência entre dois conceitos analisados.

3. Framework X2Rel

A linguagem XML permite que a informação seja estruturada de diversas maneiras diferentes, fazendo com que cada sistema adote a estrutura que lhe seja mais conveniente. Devido a essa flexibilidade, pode-se obter um problema quando se deseja manipular o mesmo tipo de informação vinda de diversas fontes. Na ocorrência deste cenário, existirá a necessidade de criar manipuladores para cada diferente estrutura utilizada pelas fontes de dados, o que pode tornar o processo inviável.

A fim de solucionar este problema de heterogeneidade presente em arquivos XML pertencentes ao mesmo domínio, mas com diferentes estruturas, o *framework* X2Rel propõe o armazenamento das informações em um banco de dados relacional, permitindo que a submissão de uma única consulta SQL (Structured Query Language) recupere o conteúdo de diversos elementos estruturados originalmente de formas diferentes. Desta forma, a tecnologia já desenvolvida para o uso de banco de dados relacional pode ser aproveitada para a solução do problema citado.

Para tal proposta, este framework emprega ontologias em um processo de unificação das diferentes estruturas de entrada, gerando um único esquema global. Na etapa seguinte, o modelo global é traduzido para um modelo de banco de dados relacional. Deste modo, consegue-se mapear as diferentes estruturas dos arquivos XML em uma única, agora padronizada. A partir daí, os dados podem ser migrados para este modelo e resgatados através de consultas SQL.

Buscando prover essas funcionalidades, o framework é dividido em cinco diferentes módulos chamados OntoGen (Ontology Generator), OntoRel (Ontology to Relational), CMap (Concept Mapper), XMap (XML Mapping) e QMap (Query Mapping), conforme descritos a seguir:

- OntoGen – é responsável pela geração da ontologia que descreve documentos XML pertencentes a um mesmo domínio, mas estruturalmente incompatíveis. Inicialmente identifica e transforma a estrutura de cada arquivo de entrada em uma ontologia que a represente. Após isso, baseia-se em um conjunto de regras de unificação para gerar uma ontologia global que representa todos os arquivos de entrada. Este módulo recebe na entrada os arquivos XML e gera na saída uma ontologia em OWL.
- OntoRel – é responsável por, a partir da ontologia global e um conjunto de regras de transformação, criar um modelo de base de dados relacional equivalente, tendo como saída um script SQL com os comandos de criação de tabelas e colunas, além de um arquivo XML descrevendo este modelo.
- CMap – É responsável pela geração e representação das equivalências entre os conceitos presentes nos arquivos XML iniciais e os conceitos do modelo relacional. Para tal, recebe como entrada os arquivos iniciais, a ontologia global e a descrição do banco de dados gerada pela OntoRel, e tem como saída um documento de mapeamento.

- XMap – é responsável pela inserção dos dados contidos nos arquivos XML no esquema relacional criado. Recebe como entrada todos os arquivos XML iniciais, o documento de mapeamento e a descrição do modelo relacional. Gera na saída um script SQL com os comandos de inserção de dados no banco de dados relacional.
- QMap – é responsável pelo mapeamento de consultas em XQuery (*XML Query Language*) para consultas em SQL que serão aplicadas no banco de dados gerado. Tem como entrada uma consulta XQuery e como saída uma consulta correspondente em SQL.

A arquitetura do framework X2Rel é apresentada na Figura 1.

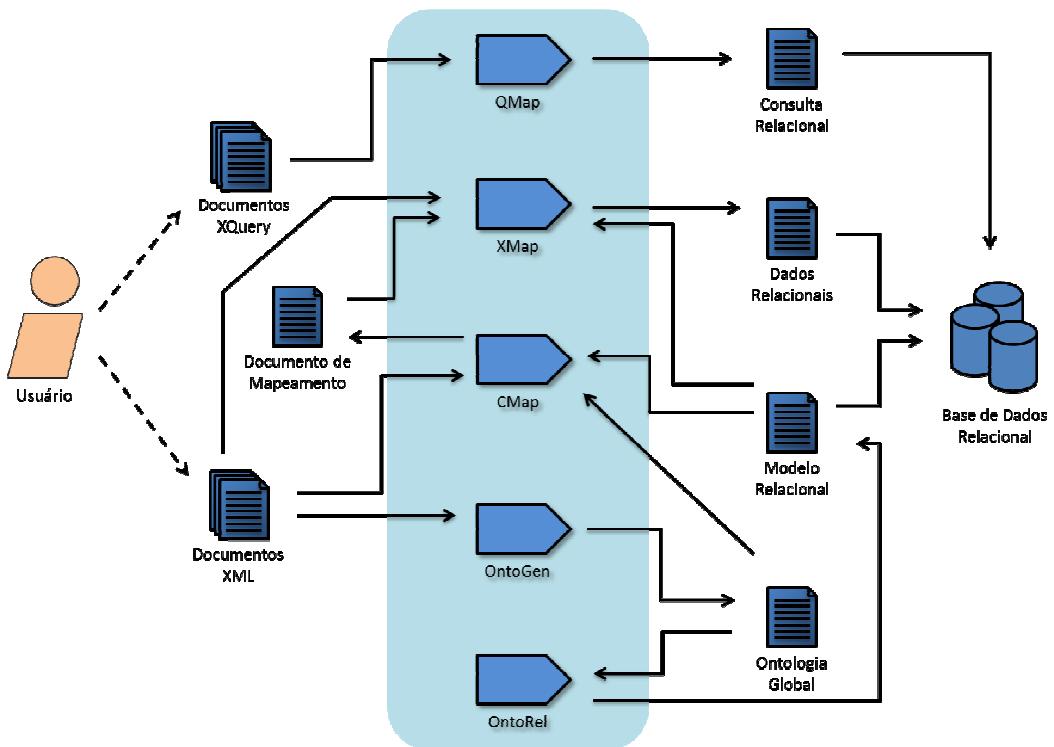


Figura 1. Arquitetura do framework X2Rel

O processo proposto leva em consideração que a estrutura utilizada nos documentos XML descreve o conteúdo que está ali inserido, ou seja, os nomes dos elementos XML descrevem o seu significado semântico. Caso os documentos XML não possuam esta característica, como nas próprias ontologias utilizadas na unificação dos esquemas, a utilização deste framework não é indicada. O framework ainda está em fase de desenvolvimento, de modo que ainda não foi definida uma forma de atualização do esquema integrado caso alguma das entradas sofra alguma modificação estrutural.

4. CMap

A proposta apresentada pelo framework X2Rel pode ser de grande valia em alguns cenários, devido a grande simplificação do processo de manipulação de documentos XML. Após realizar a unificação dos modelos, verificou-se a necessidade de um mapeamento de equivalências entre os modelos iniciais, a ontologia, e o modelo

relacional gerado, para permitir a migração e o resgate das informações inicialmente presentes nos arquivos XML.

Para realizar geração e documentação das equivalências entre esquemas, esta ferramenta baseou-se nas regras e garantias dos processos de unificação e transformação feitos pelos módulos OntoGen e OntoRel. Deste modo, o processo de mapeamento foi separado em quatro diferentes níveis, conforme mostrados na Figura 2.

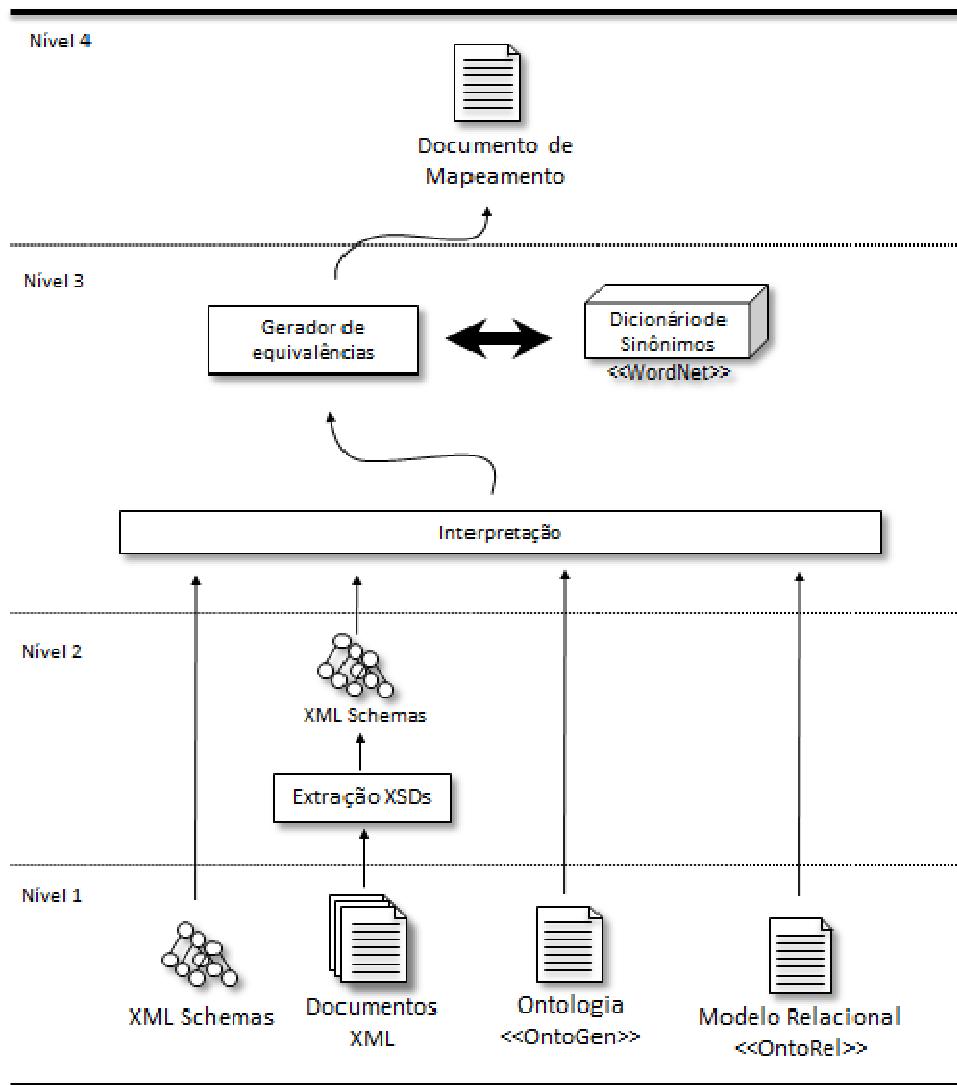


Figura 2 – Abordagem proposta para o módulo CMap

No primeiro nível são encontrados os documentos ou esquemas XML, a ontologia e o modelo relacional, fornecidos como entrada pelo usuário. A entrada direta dos esquemas no módulo acelera o processo realizado, visto que apenas a estrutura do XML é descrita. Porém, como é comum que documentos XML não estejam associados a um esquema, permitiu-se também que o mesmo fosse usado diretamente como entrada.

No nível dois, se a entrada teve documentos XML, é feita a extração automatizada dos esquemas dos mesmos, através de uma funcionalidade presente na biblioteca Castor [Castor, 2011]. Após esta etapa, agora já no nível três, os esquemas, a

ontologia e a descrição do modelo relacional passam por um módulo de interpretação, a fim de representar os conceitos e relacionamentos em memória de forma apropriada.

A partir disso, dá-se início ao processamento no módulo gerador de equivalências, onde a ferramenta realizará a geração de equivalências a partir dos conceitos e relacionamentos encontrados na ontologia, e de um conjunto de assertivas de correspondência para solução dos conflitos encontrados. Nem todos os conflitos encontrados podem ser resolvidos de forma automática, fazendo com que seja necessária a intervenção do usuário em alguns casos, e, consequentemente, tornando o processo semi-automático.

Ao final da geração de equivalências, chega-se ao nível quatro, onde as mesmas são documentadas através de um documento de mapeamento descrito na linguagem XML, que por sua vez, será a saída da ferramenta proposta. Realizado este mapeamento, será possível partir de um conceito de um documento XML e encontrar seu equivalente no modelo de banco de dados, assim como saber qual(is) conceito(s) de cada um dos arquivos XML foi(ram) inserido(s) em uma coluna ou tabela do banco de dados.

4.1. Proposta de Geração de Equivalências

A geração de equivalências é dada de forma que, para cada conceito da ontologia, é encontrado o conceito equivalente no modelo relacional, e realizada uma varredura nos esquemas extraídos dos documentos XML buscando encontrar o(s) conceito(s) equivalente(s) em cada esquema. Tal processo dar-se-á com o uso de um conjunto de assertivas de correspondência que definem se dois conceitos são ou não equivalentes.

Para a definição das assertivas, foram utilizadas algumas formalidades buscando facilitar sua compreensão. Um conceito é considerado léxico se seu conteúdo é representável diretamente em computador, através de cadeias de bits (números, cadeias de caracteres, etc), caso contrário, será considerado um conceito não léxico (objetos complexos).

Além disso, considera-se que dois conceitos possuem afinidade se seus nomes forem iguais, ou sinônimos de acordo com o dicionário de sinônimos WordNet [Wordnet, 2011]. Por fim, dois relacionamentos possuem afinidade se os conceitos de origem e os conceitos de destino destes relacionamentos forem equivalentes, ou ainda se o conceito de origem de um relacionamento for equivalente ao conceito de destino do outro.

A geração de equivalências ocorre de maneiras distintas para conceitos léxicos e conceitos não léxicos. Caso um conceito C_i presente na ontologia seja do tipo não léxico, ele será mapeado apenas uma vez, e, caso seja do tipo léxico, ele será mapeado uma vez para cada diferente relacionamento R que possua com um conceito não léxico.

A seguir, são apresentadas as assertivas de correspondência criadas:

- Definição 1: Um conceito não léxico C_i presente na ontologia será considerado equivalente a um conceito C_j de um documento XML, se C_i e C_j possuírem afinidade.

- Definição 2: Um conceito não léxico C_i presente na ontologia será considerado equivalente a um conceito C_j no modelo relacional se C_i e C_j possuírem afinidade, e se C_j representar uma tabela.
- Definição 3: Um conceito léxico C_i presente na ontologia e em um relacionamento R_i que possua também um conceito não léxico C_k , será considerado equivalente a todo conceito C_j de um documento XML em que C_i e C_j possuam afinidade, e se C_j possuir um relacionamento R_j que possua afinidade com o relacionamento R_i citado.
- Definição 4: Um conceito léxico C_i presente na ontologia e em um relacionamento R_i que possua também um conceito não léxico C_k , será considerado equivalente a um conceito C_j do modelo relacional, se C_i e C_j possuírem afinidade, se C_j for uma coluna no modelo relacional, e se esta coluna pertencer a uma tabela equivalente ao conceito C_k citado.

Pode-se afirmar que, em relação a um conceito presente na ontologia, sempre haverá um conceito equivalente no modelo relacional, e haverá ao menos um conceito equivalente em um dos esquemas XML de entrada.

Com estas definições, apenas um conflito não é resolvido. Caso haja uma equivalência entre um conceito léxico presente em um documento XML e um conceito não léxico na ontologia, consequentemente ter-se-á uma equivalência entre um conceito léxico e uma tabela do modelo relacional. Neste caso, a interversão do usuário é requisitada para informar em qual coluna desta tabela o conteúdo em questão deve ser inserido.

Dadas estas assertivas é possível afirmar que, dentro do cenário proposto, todos os conceitos e relacionamentos serão corretamente mapeados, possibilitando a continuação do processo realizado pelo *framework*.

4.2. Documento de Mapeamento

Após gerar as equivalências entre conceitos no contexto citado, optou-se por descrevê-las em um documento de mapeamento estruturado com a linguagem XML. A estrutura proposta para este documento baseia-se nos conceitos presentes na ontologia, assim como na metodologia de mapeamento apresentada.

Na Figura 3 podem-se perceber as equivalências dos conceitos *paper* e *title*. O conceito *paper* do documento “Doc A.xml” transformou-se na tabela *paper* no modelo relacional. Pode-se perceber também que o conceito *title* presente no documento “Doc A.xml” transformou-se na coluna *title* da tabela *paper*, dentre outras equivalências ali representadas.

O documento baseia-se no elemento *concept* (linha 03) para representar as equivalências. É criado um elemento deste tipo para cada um dos conceitos presentes na ontologia, armazenando o nome do conceito no atributo *name*. Dentro destes elementos são referenciados os conceitos equivalentes em cada um dos arquivos XML, e no modelo relacional.

Para cada documento presente na entrada é criado um elemento *source*, filho do elemento *concept* juntamente com o atributo *id*, o qual tem como conteúdo o nome do

arquivo (linha 04). Caso haja uma equivalência neste documento, é criado um ou mais subelementos *xpath* que terão como conteúdo as expressões XPath dos conceitos equivalentes na estrutura deste arquivo (linhas 05 e 06). Caso não haja uma equivalência, o elemento *xpath* não é inserido (linha 27).

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <mapping>
03   <concept name="paper">
04     <source id="Doc A.xml">
05       <xpath>/conference/papers/paper</xpath>
06       <xpath>/symposium/papers/paper</xpath>
07     </source>
08     <source id="Doc B.xml">
09       <xpath>/conference/papers/paper</xpath>
10     </source>
11     <source id="Doc C.xml">
12       <xpath>/papers/paper</xpath>
13     </source>
14     <relational>
15       <type>table</type>
16       <name>paper</name>
17     </relational>
18   </concept>
19   <concept name="title">
20     <source id="Doc A.xml">
21       <xpath>/conference/papers/paper/title</xpath>
22       <xpath>/symposium/papers/paper/title</xpath>
23     </source>
24     <source id="Doc B.xml">
25       <xpath>/conference/papers/paper/title</xpath>
26     </source>
27     <source id="Doc C.xml" />
28     <relational>
29       <type>column</type>
30       <name>title</name>
31       <table>paper</table>
32       <domain>string</domain>
33     </relational>
34   </concept>
35 </mapping>
```

Figura 3 – Documento de mapeamento

Para a representação do mapeamento da equivalência no modelo relacional, inicialmente é inserido um elemento *relational* como filho do elemento *concept* (linhas 14 e 28). A partir disto, podem ocorrer duas situações distintas. Se a equivalência for referente a uma tabela do modelo relacional, são criados dois subelementos *type* e *name*, onde o conteúdo do elemento *type* será “*table*” e do elemento *name* será o nome da tabela (linhas 15 e 16). Caso a equivalência seja referente a uma coluna no modelo relacional, o elemento *type* terá o conteúdo “*column*” e o elemento *name* terá o respectivo nome da coluna. Neste caso, além destas, serão criados mais dois elementos de nomes *table*, que conterá o nome da tabela que a coluna pertence, e *domain*, que conterá o tipo de dado usado na coluna (linhas 29 a 32).

5. Implementação

A implementação da ferramenta foi feita utilizando a linguagem Java, e sua interface desenvolvida com o uso do *Swing*. Foram criadas classes que permitem representar os conceitos e relacionamentos das diferentes estruturas de entrada através de objetos do mesmo tipo, facilitando a manipulação dos mesmos durante o processo.

Após a extração dos esquemas XML, caso necessário, o módulo de intepretação realiza a extração dos conceitos e relacionamentos de todos os arquivos de entrada e os agrupa em objetos representando cada documento. Para análise dos documentos XSD foram também utilizados os códigos usados na ferramenta OntoGen, os quais vieram de um trabalho implementado por Garcia (2005), que consiste em extrair o modelo conceitual canônico de documentos XSD. Já as análises da ontologia e do modelo relacional descrito em XML foram construídas especificamente para esta ferramenta.

Os objetos que representam os documentos possuem métodos que recebem um conceito como parâmetro e, com base nas assertivas de correspondências definidas, retornam todos os conceitos equivalentes presentes naquele documento. Desta forma, uma classe de controle parte de cada um dos conceitos presentes na ontologia e recebe suas equivalências, as quais são repassadas para uma classe responsável por montar o documento de mapeamento.

A interface construída consiste em três janelas principais e algumas secundárias. A primeira é onde o usuário define os documentos de entrada e dá inicio ao processo. A partir disso, a janela inicial é substituída por uma janela de progresso, que, ao final do processo, é substituída por outra que apresenta o documento de mapeamento gerado. Em certos momentos o usuário pode optar por visualizar o conteúdo dos arquivos de entrada, o que faz com que uma janela se sobreponha apresentando o conteúdo do arquivo. Por fim, há uma janela de resolução de conflito que é apresentada para o usuário toda vez que uma tomada de decisão externa se diz necessária.

6. Conclusão e Trabalhos Futuros

A unificação de documentos XML mostrou-se uma tarefa complexa ao fazer uma análise dos processos realizados pelos módulos OntoGen e OntoRel. Buscando documentá-la, este trabalho apresentou uma proposta para gerar equivalências entre as estruturas de representação de informação geradas, permitindo assim a continuação do desenvolvimento do framework através do desenvolvimento do módulo XMap.

A ferramenta proposta tem como entrada os documentos XML ou XSD que se deseja unificar, a ontologia gerada pelo módulo OntoGen e o modelo relacional, descrito em XML, gerado pelo módulo OntoRel. A partir das informações inseridas nestes documentos, as equivalências são geradas tomando-se como base as assertivas de correspondência definidas, e, por fim, as equivalências são representadas em um documento de mapeamento, o qual é também a saída da ferramenta.

Mesmo que a geração de equivalências proposta dependa de intervenções do usuário em alguns casos, o processo é realizado quase que em sua totalidade de forma automática, e permite o mapeamento de todas as equivalências existentes, tal como a representação das mesmas.

Porém, mesmo que se tenha alcançado o objetivo, a unificação de conceitos presentes nos esquemas iniciais se dá apenas através do uso de um dicionário de sinônimos da língua inglesa e uma análise direta na *string* que compõe o nome do elemento, fazendo com que alguns elementos equivalentes possam não ser considerados como tal.

Buscando melhorar a ferramenta seria interessante a adição do uso de um dicionário de sinônimos em português, e a proposta de algum método que permita realizar o processo de interpretação de elementos de nome composto por duas ou mais palavras, separadas por algum marcador, assim como a detecção e geração de equivalências entre estes elementos. Além disso, o uso de uma busca por equivalências através da análise da raiz das palavras que nomeiam os elementos também seria de grande valia.

Agradecimentos

Este trabalho foi parcialmente financiado por SESU/MEC (PET-Programa de Educação Tutorial) e FAPERGS (Auxílio Recém Doutor – Processo número 11/0748-6).

Referências

- Andrade, T. Mapeamento de esquemas XML integrados para bancos de dados relacionais. Trabalho de Graduação - Universidade Federal de Santa Maria. Santa Maria. 2010.
- Castor. The Castor Project, 2011. Disponível em: <<http://www.castor.org/>>. Acesso em: 29 Setembro 2011.
- Frozza, A. A. Um Método para Determinar a Equivalência Semântica entre Esquemas GML. Dissertação de Mestrado - Universidade Federal de Santa Catarina. Florianópolis. 2007.
- Garcia, L. G. Uma Ferramenta para Engenharia Reversa de Esquemas XML em Esquemas Conceituais no Ambiente BInXS. Trabalho de Graduação - Universidade Federal de Santa Catarina. Florianópolis. 2005.
- Mello, M. OntoGen: Uma Ferramenta para Integração de Esquemas XML. Trabalho de Graduação - Universidade Federal do Rio Grande do Sul. Porto Alegre. 2007.
- Saccol, D. Integração de Esquemas em Fontes Heterogêneas XML. Programa de Pós-Graduação em Informática - Universidade Federal do Rio Grande do Sul. Porto Alegre. 2005.
- Vidal, V. M. P.; Vilas Boas, R. M. D. F. Uma Abordagem Top-Down para Geração das Correspondências entre XML Schemas Semânticos. Universidade Federal do Ceará. Fortaleza. 2002.
- Wordnet. WordNet - A lexical database for English, 2011. Disponível em: <<http://wordnet.princeton.edu/>>. Acesso em: 1 Outubro 2011.

XVersioning - Uma Ferramenta para Versionamento de Esquemas XML

Renan Bet Rodrigues¹, Denio Duarte²

¹Departamento de Ciência da Computação
Universidade do Estado de Santa Catarina (UDESC) – Joinville, SC – Brasil

²Universidade Federal da Fronteira Sul (UFFS) – Chapecó, SC – Brasil

renanbet@hotmail.com, duarte@uffs.edu.br

Abstract. This work presents an XML document versioning study. Based on this study, we propose a tool, named XVersioning, for versioning XML schemas. The aim of XML schema versioning is keep the XML document database valid in relationship to updated XML schemas without changing the database. Roughly speaking, this is achieved storing all XML schema versions in the database.

Resumo. Este artigo apresenta um estudo de abordagens para versionamento de documentos XML e, baseado neste estudo, a proposta de uma ferramenta para o versionamento de esquemas XML implementados na linguagem XML Schema e a validação das instâncias desses esquemas. O objetivo do versionamento de esquemas XML é garantir que uma instância XML D anteriormente válida em relação a um esquema S não perca a validade após atualizações em S ocorridas durante o seu ciclo de vida. Para tal, é necessário recuperar as várias versões de S existentes S_0, \dots, S_n . A validação, então, é feita sobre uma dessas versões S_i ($0 \leq i \leq n$) até que D seja ou não validado. Em muitas situações, o versionamento de esquemas evita perda de dados de instâncias XML bem como a inoperância de aplicações baseadas em esquemas para a troca de dados XML.

1. Introdução

A tecnologia XML (*eXtensible Markup Language*) foi padronizada em 1996 pela *World Wide Web Consortium* (W3C). A principal característica desta linguagem é a sua independência de plataforma, a sua aplicabilidade a uma grande variedade de aplicações heterogêneas, e a sua flexibilidade possibilitando ao usuário definir seus próprios elementos e linguagens de marcação. Assim, um documento XML descreve os dados e suas estruturas. Devido às características citadas, a linguagem XML é o principal mecanismo para representação de dados semi-estruturados. Os dados semi-estruturados são caracterizados por não possuir uma estrutura explícita e regular a ser seguida.

Muitas aplicações se apoiam na tecnologia XML para trocar dados entre si e a maioria delas precisa conhecer a estrutura dos documentos a "priori", a fim de aperfeiçoar consultas ou então integrar documentos de várias fontes. Para tornar isto possível, é necessário que existam padrões estabelecidos entre as aplicações, ou seja, definir a estrutura dos documentos XML a serem trocados.

O padrão da estrutura de documentos XML é estabelecido por esquemas. Ao associar um esquema a um documento XML, esse se torna uma instância do esquema.

Um documento XML que respeita as regras impostas pelo esquema X é um documento válido em relação a X . Portanto a validade de um documento XML está totalmente ligada ao padrão previamente especificado e associado.

Conforme [Galante 2003], a padronização de documentos XML é necessária devido às variações de estruturas dos documentos para uma mesma aplicação, ou seja, vários documentos diferentes com a mesma informação. Ao estabelecer este padrão algumas vantagens podem ser citadas como:

- A estrutura dos documentos é definida a “priori”.
- Auxílio na consulta sobre os documentos associados.
- Otimização na consultas dos dados.
- Permite a padronização de um documento XML.

Segundo [Bex et al. 2004], existem inúmeras formas de padronizar os documentos XML, podendo ser por uma aplicação ou então por uma alguma linguagem. Duas das principais linguagens para este fim são:

- *Document Type Definition* (DTD).
- *XML Schema Definition* (XSD).

Neste trabalho é utilizada XSD como linguagem de esquema para os documentos. Considera-se, também, que esquemas não são entidades estáticas. Na verdade, esquemas necessitam muitas vezes evoluir devido a vários fatores: (i) erros de projeto, (ii) adequação da aplicação a novas regras, (iii) novas necessidades dos usuários, entre outros.

Ao se atualizar um esquema, documentos XML que estão associados e válidos a este esquema podem perder sua validade devido às alterações realizadas. Segundo [Silveira 2007], a alteração dos documentos para a nova estrutura seria uma solução. Contudo, para esquemas com instâncias que apresentam alto grau de distribuição, tal abordagem não pode ser adotada, pois normalmente a totalidade do conjunto de instâncias não é conhecido com antecedência, ou alguns documentos podem não ser acessados no momento da propagação das alterações. Logo, determinados documentos XML podem se tornar inválidos. Para contornar este problema, é aplicado o versionamento nos esquemas XML, pois, segundo [Galante 2003], a evolução e versionamento de esquemas são duas técnicas para realizar modificações na estrutura dos esquemas, mantendo a consistência entre o esquema e os documentos associados. Enquanto a evolução mantém apenas a versão corrente do esquema e seus respectivos documentos associados, o versionamento preserva todas as versões anteriores à evolução, e também seus documentos associados em cada evolução, garantindo a funcionalidade das instâncias armazenadas frente às alterações, evitando a perda de informações dos documentos e garantindo a compatibilidade das aplicações.

Este trabalho propõe *XVersioning*, uma ferramenta que além de tratar a evolução e versionamento de esquemas XML, valida as instâncias perante o conjunto de versões de seu esquema associado. Ao ocorrer a evolução de um esquema X , cria-se uma nova versão do esquema, garantindo que documentos XML válidos em relação a X em determinado momento do tempo, possam ser acessados pelas aplicações através de X . Essa ferramenta é proposta como um passo inicial para a construção de um sistema de versionamento de esquemas XML. Assim, duas das abordagens que serão apresentadas (*snapshot-collection* e *snapshot-delta*) foram escolhidas para serem implementadas e testadas na *XVersioning*.

O restante deste trabalho está organizado da seguinte forma: a próxima seção descreve algumas abordagens para versionamento, apresentando ao fim, um breve comparativo entre as abordagens. Em seguida é apresentada a ferramenta *XVersioning*, contribuição deste trabalho, e, finalmente, a Seção 4 apresenta a conclusão e os trabalhos futuros.

2. Abordagens para Versionamento de Documentos

Esta seção apresenta algumas abordagens para o versionamento de documentos XML. O estudo destas abordagens é importante para a proposição da contribuição deste trabalho.

2.1. Snapshot-collection e Snapshot-delta

Em [Chawathe et al. 1998] são propostas duas abordagens que podem ser empregadas para o versionamento de documentos semi-estruturados que evoluem linearmente¹, denominadas *snapshot-collection* e *snapshot-delta*.

A *snapshot-collection* consiste em armazenar todos os estados assumidos pelo documento no seu ciclo de evolução. Nesta abordagem é armazenado o estado original do documento X . Quando X sofre uma modificação, é gerada uma nova cópia X' com a parte modificada, ou seja, contém todos os objetos alterados de X e os que permaneceram intactos.

A *snapshot-delta* consiste em armazenar um único estado do documento atualizado, por exemplo o mais recente, em conjunto com uma coleção de *scripts* de conversão (chamados *deltas*). Dessa forma, a partir do estado armazenado em sua totalidade pode-se gerar qualquer estado do documento aplicando os *deltas*.

A primeira abordagem otimiza o tempo de recuperação de um estado qualquer de um documento, pois todos estão presentes em sua totalidade, porém o custo em espaço de armazenamento é alto pois é necessário armazenar todos os estados do documento. A segunda abordagem otimiza o espaço consumido para armazenar o conjunto de estados, considerando que um estado cujo conteúdo está armazenado explicitamente ocupa um espaço maior do que um *delta*. Em compensação, este método requer um maior processamento para reconstruir um determinado estado, maximizando o tempo de recuperação de estados.

2.2. Usefulness-Based Change Control

Em [Chien et al. 2001] é apresentada a abordagem *page-usefulness*, um método para reduzir os custos de recuperação de estados de um documento versionado. Neste método há um agrupamento físico de todos os trechos válidos de um determinado estado em algumas páginas de dados. Quando o número de trechos válidos em determinada página cai abaixo de um certo limiar, os trechos são copiados para uma nova página. Na reconstrução de um estado, são acessadas as páginas úteis (*useful*) para este estado.

A abordagem *Usefulness-based Change Control* (UBCC) foi baseada nas abordagens propostas por [Chawathe et al. 1998] e possui duas variantes: (i) *Edit-based UBCC*, baseado em *scripts* de conversão; e (ii) *Copy-based UBCC*, baseado em cópias de

¹Não geram duas versões paralelas do mesmo documento e apenas esse tipo de evolução é considerado neste trabalho.

segmentos. Além das variantes, o UBCC foi projetado para funcionar com os métodos *snapshot-collection* e *snapshot-delta* de forma híbrida, juntamente com o conceito *page-usefulness*.

No *Edit-based UBCC* ao ocorrer uma instrução de inserção de um objeto em determinada posição, primeiramente obtém-se o *script* de transição dos estados. Em seguida, são buscados, através do método *page-usefulness*, os trechos em estados anteriores do presente estado, até que o trecho para inserção seja reconstruído e assim aplicado a instrução. Já no *copy-based UBCC*, ao invés de representar as transições de estado com *scripts*, é aplicado o conceito de referência para fragmentos equivalentes. Referências são nós da árvore que referenciam outro nó da mesma árvore. A reconstrução de um estado é feita através do *page-usefulness* que contém os objetos e referências de subárvore. Ao encontrar uma referência, é feita uma recursão para recuperar o trecho específico do estado.

Segundo [Chien et al. 2002], um estudo de caso comparativo entre as duas variantes do UBCC, mostrou que ambas as abordagens apresentaram um desempenho e custo de armazenamento similares. Ora o *copy-based UBCC* supera o *edit-based UBCC* ora não, porém oferece flexibilidade para a escrita de consultas, mas com o mesmo desempenho.

2.3. Reference-Based Veersion Model

Proposto em [Chien et al. 2002], o *Reference-Based Version Model* (RBVM) contém as seguintes características:

- Preserva a estrutura lógica do documento, permitindo recuperação de estados.
- A história de evolução de um documento XML pode ser representada por outro documento XML.
- Utiliza técnicas para reduzir o custo de armazenamento da informação. Umas delas é o conceito *page-usefulness*, detalhado em [Chien et al. 2001].

Além dos elementos do XML, a árvore proposta no *RBVM* inclui nós de estado e de referência. Nós de estado servem de raiz para as subárvore, identificando cada uma das versões de um documento. Nós de referência servem de apontadores para subárvore em comum a dois estados distintos. Assim a recuperação de um estado começa a partir do próprio estado a ser reconstruído. Os dados preservados são recuperados recursivamente através dos nós de referência.

2.4. Xyleme

Em [Marian et al. 2001] foi proposto um sistema de gerenciamento de documentos XML em evolução. O sistema consiste na obtenção de novos estados para documentos através da *Web* e comparados com os estados armazenados na base de dados. Nesta comparação, é executado o algoritmo *XyDiff* [Cobena et al. 2002] que obtém as diferenças entre os dois estados. A partir deste algoritmo é gerado um *script* para controlar as transições entre os estados de um documento. Esse *script* é armazenado em um banco de dados. O novo estado adquirido torna-se o estado corrente da aplicação. O sistema trabalha como um sistema observador, na medida em que é encontrado algum documento evoluído é feito o processamento para um novo estado corrente.

Algumas características desse sistema pode ser consideradas, tais como:

- A representação lógica é baseada em deltas, assim como o método *snapshot-delta* [Chawathe et al. 1998]. Porém são armazenados deltas completos, pois possuem instruções para navegação em ambos sentidos para a recuperação de estados.
- Há identificadores persistentes para cada nó do documento XML, obtidos através do algoritmo *XyDiff*, que permitem registrar modificações nos dados e otimizar a recuperação.

O armazenamento dos estados consiste em armazenar somente o estado corrente, e através da atribuição de identificadores que contém todos os deltas completos é possível recuperar um estado anterior desejado.

2.5. Identificadores Persistentes

Em [Wong and Lam 2002] é proposto um método que utiliza um conceito de identificadores persistentes para referenciar nós da árvore XML, armazenando a informação através de um conjunto de deltas e estados. Este método há diversos pontos de partida para a recuperação do estado desejado em um conjunto de estados. A intenção para o armazenamento de um conjunto de estados para o inicio da recuperação surge do problema de recuperar a primeira versão, caso o estado armazenado for o último, ou então a última versão, caso o estado armazenado for o primeiro. Outra vantagem nesta abordagem é o caso do armazenamento de um *delta* com espaço maior em relação ao estado gerado, onde é armazenado o estado e não o delta.

As operações suportadas neste modelo são: inserção, remoção, atualização, movimentação e cópia. O conhecimento das operações é a base para o funcionamento deste método, pois é atribuído um custo ao *delta*. Esse custo é medido pelo número de operações contidas. Ao solicitar a recuperação de um estado, varre-se uma estrutura de dados que armazena informações referente a cada estado armazenado. Nessa estrutura há informações se o estado está materializado integralmente e o custo do *delta* de seu antecessor para obtê-lo. O estado mais próximo do solicitado a recuperar é o com menor número de operações necessárias.

2.6. Comparativo das Abordagens

Essa seção apresentou algumas abordagens para a evolução e versionamento de documentos XML, essenciais para a realização deste trabalho. A Tabela 1 apresenta um comparativo entre as abordagens apresentadas. Para cada abordagem, é marcado se esta é baseada em *snapshot-delta*, *snapshot-collection* ou abordagem própria. Assim, as abordagens apresentadas em [Chien et al. 2001], [Marian et al. 2001] e [Wong and Lam 2002] baseiam-se na abordagem de [Chawathe et al. 1998]. Os métodos apresentados nas abordagens utilizam ora o *snapshot-delta* ora o *snapshot-collection*, ou ainda ambos em uma abordagem híbrida. Considerando os dados apresentados na Tabela 1, percebe-se que as duas abordagens para versionamento propostas em [Chawathe et al. 1998] são métodos funcionais para o versionamento de documentos XML pois servem como base para outros métodos.

Com base nesses dados, a próxima seção apresenta a contribuição deste trabalho *XVersioning*: uma ferramenta para versionamento de esquemas XML. Esta ferramenta implementa as abordagens de versionamento *snapshot-collection* e *snapshot-delta*, por ser os dois métodos mais comuns entre as abordagens estudadas e que possui maior referência

Tabela 1. Comparativo entre as abordagens

Abordagem	<i>snapshot-delta</i>	<i>snapshot-collection</i>	Abordagem própria
[Chien et al. 2001]	X	X	
[Chien et al. 2002]			X
[Marian et al. 2001]	X		
[Wong and Lam 2002]	X	X	

bibliográfica disponível. Também é implementado um validador de documentos XML nas versões do esquema associado.

3. XVersioning

A ferramenta *XVersioning* foi desenvolvida para gerenciar versões de Esquemas XML em evolução. Ao ocorrer uma evolução em um esquema S armazenado no banco de dados, *XVersioning* cria uma nova versão S' preservando as versões antigas. Ao validar um documento XML X associado a S , X é validado perante todas as versões de S (isto é, S_0, S_1, \dots, S_n) até que X respeite uma das versões. Desta forma, é garantido que documentos associados a um esquema, e previamente válidos, não percam sua validade após ocorrer uma evolução no esquema.

Na abordagem proposta, os esquemas XML são atualizados utilizando a linguagem de atualização de documentos XML *XUpdate*. O principal motivo dessa escolha foi pelo fato de *XUpdate* ser escrito na linguagem XML e, assim, o tratamento, tanto do documento, do esquema e da atualização pode ser feito da mesma forma pela ferramenta.

A validação de documentos XML é efetuada pelo validador da API SAX pois é necessária apenas o processamento serial do documento. Ao contrário do *DOM*, que carrega todo o documento na memória, *SAX* mantém na memória apenas as *tags* que estão sendo visitadas. Obtendo assim um processamento mais rápido na validação de documentos XML.

As funcionalidades de *XVersioning* utilizadas pelo usuário podem ser divididas em quatro principais: (*i*) cadastro do esquema cuja as versões serão controladas pelo *XVersioning*, (*ii*) atualização do esquema cadastrado que provocará a geração das versões conforme uma das abordagens utilizadas, (*iii*) validação de documentos que utilizam as versões dos esquemas armazenados, e (*iv*) configurar a ferramenta que permite configurar algumas funcionalidades básicas.

A ferramenta *XVersioning* foi implementada em *Java* e utiliza o *eXist* para o gerenciamento de dados. Sua arquitetura é representada na Figura 1.

Conforme a Figura 1, a interface do *XVersioning* é Web e, assim, é acessada via *browser*. A camada JSP/Servelet é responsável pelas requisições (*request*) de uma página *JSP* e pelos envios (*response*) da solicitação ao usuário. Na camada onde se encontram as páginas *JSP*, há três servlets implementadas, são elas:

- *Servlet* de cadastro de Esquema XML.
- *Servlet* de versionamento de Esquemas XML.
- *Servlet* de validação de documentos XML.

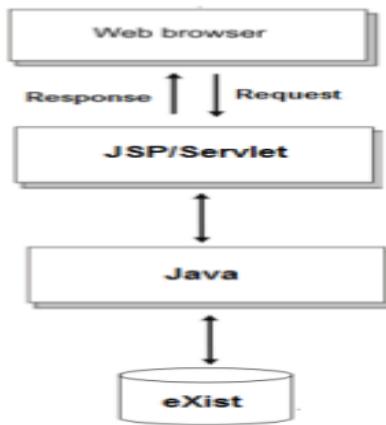


Figura 1. Arquitetura

Na requisição de um *servlet*, esta acessa a camada *Java* da aplicação. Esta camada é uma *API* desenvolvida para a ferramenta *XVersioning*, afim de cadastrar, versionar esquemas XML e validar documentos XML.

Para a *Xversioning* realizar o gerenciamento das versões de esquemas XML em evolução, foram criadas quatro coleções de documentos no banco de dados, que são: *Delta*, *Schema*, *Collection* e *FirstVersion*. Na coleção *Delta* há um único documento XML armazenado, o *version.xml*, este respeita as regras de estrutura do esquema XML apresentado na Figura 2.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <x: schema attributeFormDefault="unqualified" elementFormDefault="qualified"
  xmlns:x="http://www.w3.org/2001/XMLSchema">
3   <x: element name="versao">
4     <x: complexType>
5       <x: sequence>
6         <x: element maxOccurs="unbounded" name="esquema">
7           <x: complexType>
8             <x: sequence>
9               <x: element name="nome" type="xs:string" />
10              <x: element maxOccurs="unbounded" name="delta" type="xs:string" />
11              <x: element name="metodo" type="xs:string" />
12            </x: sequence>
13          </x: complexType>
14        </x: element>
15      </x: sequence>
16    </x: complexType>
17  </x: element>
18 </x: schema>

```

Figura 2. Esquema XML

Ao ocorrer o cadastro de um esquema XML, é adicionado na tag *versao*, uma nova tag *esquema*, com o nome e método de versionamento utilizado no esquema em questão. Em uma evolução de um esquema XML, é adicionado na tag *esquema* referente ao esquema evoluído uma nova tag *delta* em *esquema*, com o *script* de evolução como conteúdo.

Na coleção *Esquema* são armazenados todas as versões correntes dos esquemas cadastrados. Essa coleção é utilizada efetuar a primeira validação de um documento XML. A coleção *FirstVersion*, armazena a primeira versão de cada esquema. Esta coleção é acessada para selecionar a primeira versão do esquema. Caso o método de

versionamento seja o *snapshot-delta*, são aplicados deltas para materializar novas versões. Já a coleção *Collection*, caso o método escolhido para versionamento seja o *snapshot-collection*, armazena as versões diferentes da primeira e última versão dos esquemas cadastrados. Em uma validação de um documento XML, se o documento for inválido para a última e primeira versão, são selecionados as versões em sequência da coleção *Collection* para validação.

3.1. Métodos de Validação do XVersioning

Um esquema XML *S* armazenado no banco de dados da ferramenta e um documento XML *X* são as entradas para a execução da validação. Conforme os diagramas de atividades apresentados na Figura 3 e Figura 4, *X* é validado perante todas as versões de *S* (isto é, S_0, S_1, \dots, S_n) até que *X* respeite uma das versões.

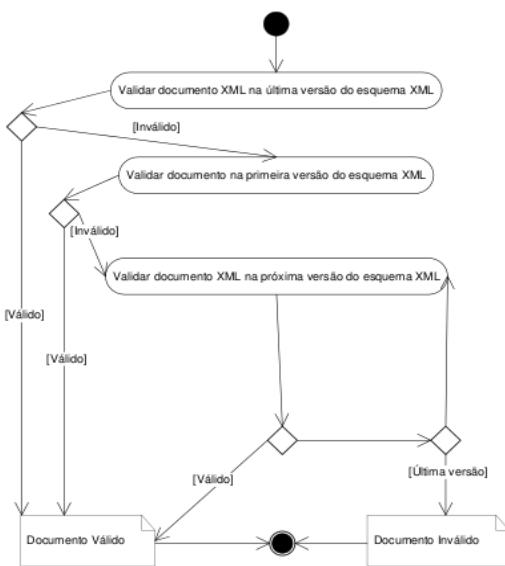


Figura 3. Validação do método *snapshot-collection*

Neste processo há duas abordagens de validação dependentes do método escolhido para o versionamento no momento do cadastro do esquema XML. No método *snapshot-delta*, as próximas versões são materializadas de acordo com os deltas armazenados no banco de dados, onde a cada validação há a materialização do esquema aplicando a evolução. No método *snapshot-collection* os esquemas XML já estão materializados, ordenados por um *id* na coleção *Collection* do banco de dados.

3.2. Interface gráfica do XVersioning

A interface gráfica do XVersioning foi desenvolvida na linguagem de programação JSP (*Java Server Pages*). A Figura 5 mostra a tela principal do XVersioning, esta é dividida em 3 áreas representadas pelas marcas numeradas na figura.

A Marcação 1 representa a área que contém o menu do XVersioning, composto pelas opções: *Esquema*, *XML* e *Opções*. A opção *Esquema*, contém submenus com as opções: (i) Criar e Versionar um Esquema XML, (ii) Validar um documento XML, e (iii) Ativar o console de monitoramento da validação do documento XML. A Marcação

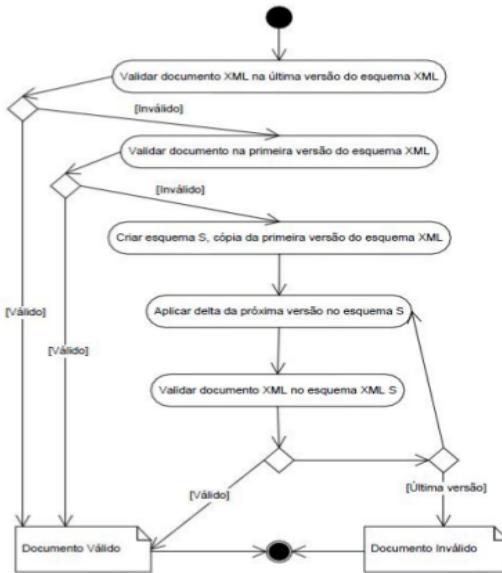


Figura 4. Validação do método *snapshot-delta*

2 representa a área onde é mostrado todos os esquemas XML e suas versões armazenadas no banco de dados do *XVersioning*. Nesta área há uma *tree-view* que contém *links* para visualização do conteúdo para cada versão de cada esquema. A Marcação 3 é onde o conteúdo das versões dos esquemas XML são apresentadas. Ao clicar em um do *links* para versões de um dos esquemas XML apresentados na Marcação 2, é mostrado o conteúdo do esquema XML naquela versão.

4. Conclusão

Neste trabalho foram estudadas algumas abordagens para o versionamento de documentos XML (Seção 2). Dentre as abordagens estudadas, duas foram utilizadas neste trabalho: *snapshot-collection* e *snapshot-delta*. Um protótipo de versionamento, então, foi implementado para identificar o comportamento dessas abordagens no intuito de apoiar futuras pesquisas nesta área. Ao versionar um esquema XML, suas instâncias estarão válidas em relação ao conjunto de versões do esquema associado, e não mais a um único esquema XML.

Assim, a ferramenta *XVersioning* e uma API podem ser utilizada em outras implementações. *XVersioning* contém além do processo de versionamento de esquemas XML, o processo de validação de um documento XML perante um conjunto de versões do esquema XML associado. A ferramenta foi validada através de um estudo de caso que pode ser encontrado em [Rodrigues 2011]. Neste estudo de caso, a ferramenta se mostrou adequada para o versionamento de esquemas e validação de documentos XML associados aos esquemas XML versionados.

Como sugestão de trabalhos futuros enumera-se: (i) implementar o método proposto em [Wong and Lam 2002] no processo de versionamento de esquemas XML da *XVersioning*, (ii) fazer um teste de desempenho do protótipo utilizando várias coleção de documentos com tamanhos e quantidades diversas, (iii) otimizar o processo de versionamento de esquemas XML, (iv) criar uma interface gráfica para criação de *deltas* dos esquemas XML, e (v) implementar o processo de validação de forma incremental, ou

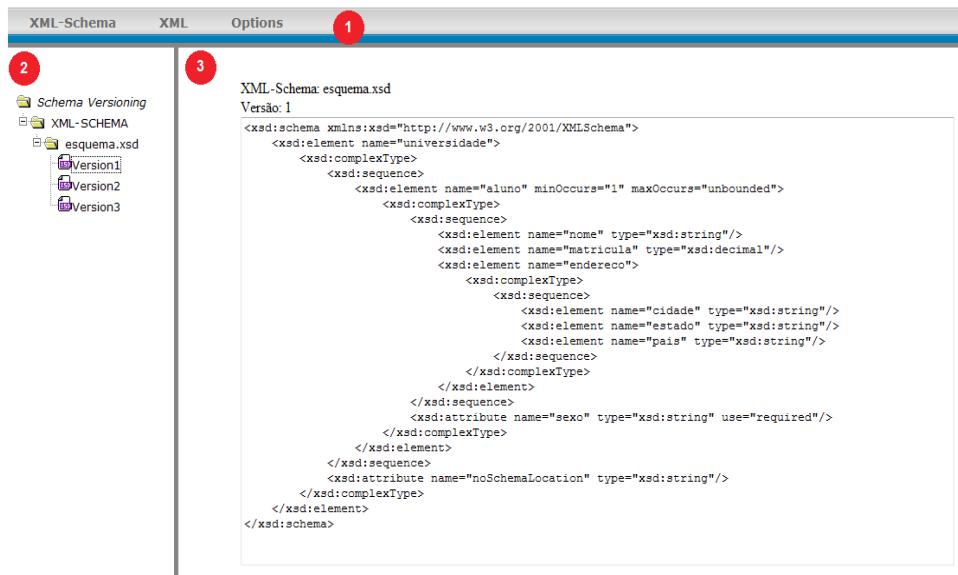


Figura 5. Interface gráfica do XVersioning

seja, após um erro na validação em relação a um esquema, a próxima versão valida apenas a parte do documento que provocou o erro.

Referências

- Bex, G. J., Neven, F., and Bussche, J. V. (2004). DTDs versus XML schema: a practical study. *WebDB 2004*, pages 79–84.
- Chawathe, S., Abiteboul, S., and Widom, J. (1998). Representing and querying changes in semistructured data. *Data Engineering, International Conference on*, 0:4.
- Chien, S. Y., Tsotras, V. J., and Zaniolo, C. (2001). XML document versioning. *SIGMOD Rec.*, 30:46–53.
- Chien, S.-Y., Tsotras, V. J., and Zaniolo, C. (2002). Efficient schemes for managing multiversion XML documents. *The VLDB Journal*, 11:332–353.
- Cobena, G., Abiteboul, S., and Marian, A. (2002). Detecting changes in xml documents. *Data Engineering, International Conference on*, 0:0041.
- Galante, R. d. M. (2003). Modelo temporal de versionamento com suporte à evolução de esquemas. Master's thesis, Instituto de Informática da UFRGS, Porto Alegre - RS.
- Marian, A., Abiteboul, S., Cobena, G., and Mignet, L. (2001). Change-centric management of versions in an XML warehouse. In *Proceedings of the 27th VLDB*, San Francisco, CA, USA.
- Rodrigues, R. B. (2011). XVersioning - uma ferramenta para versionamento de esquemas XML. TCC - Udesc - Joinville.
- Silveira, V. (2007). X-Spread: Um mecanismo automático para propagação da evolução de esquemas para documentos XML. Master's thesis, UFRGS, Porto Alegre - RS.
- Wong, R. K. and Lam, N. (2002). Managing and querying multi-version xml data with update logging. In *Proceedings of DocEng '02*, New York, NY, USA. ACM.

Análise de Abordagens para *Matching* de Formulários na *Deep Web*¹

Augusto Ferreira de Souza, Ronaldo dos Santos Mello

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina

Caixa Postal 476 – 88.040-900 – Florianópolis –SC – Brasil

augustofs@gmail.com, ronaldo@inf.ufsc.br

Abstract. This paper presents a survey of Web form matching approaches in the context of the Deep Web. First, it introduces the features used by each author regarding matching processes. In the following, a comparison of related work is provided. It concludes with an analysis of their contributions and limitations, and suggests some topics for future research.

Resumo. Este artigo apresenta uma revisão da literatura sobre matching de formulários Web no contexto da Deep Web. Primeiramente, abordam-se as características utilizadas por cada autor para a execução do matching. Na sequência, é apresentado um comparativo dos trabalhos relacionados. O artigo conclui com uma análise das principais contribuições e limitações das abordagens, além de sugerir alguns tópicos para pesquisas futuras.

1. Introdução

O aumento do volume de dados disponíveis na *Deep Web* [Halevy et al. 2009] faz com que aumente a necessidade de acesso a essas informações por parte dos usuários. A *Deep Web* representa dados disponíveis na *Web* que são visíveis apenas quando mostrados em páginas dinâmicas criadas a partir do resultado de uma pesquisa geralmente definida sobre um formulário *Web* [Bergman 2001]. O formulário *Web* (*Web form*) é a principal interface de pesquisa para um banco de dados na *Deep Web*.

Para se ter acesso às informações da *Deep Web*, são necessários sistemas para a coleta destas informações, como por exemplo, *crawlers*, *metasearchers* e sistemas de integração de dados na *Web*. Porém, após a coleta, se faz necessária à utilização de técnicas de casamento (*matching*) para facilitar o acesso dos usuários a essas fontes de dados, principalmente as fontes de dados relevantes. O *matching* de dados nesse contexto corresponde ao casamento de esquemas de formulários *Web* e é um problema a ser tratado em diversos processos de gerenciamento de dados, como por exemplo, integração de dados, consulta a diferentes fontes de dados e pesquisa por similaridade.

¹ Este trabalho conta com recursos do CNPq através do projeto WF-Sim (Nro. processo: 481569/2010-3).

A motivação deste artigo é a dificuldade associada à execução do *matching* de formulários Web. Tais dificuldades ocorrem devido à existência de uma grande variedade de formulários com atributos diferentes que representam um mesmo conceito, a existência de atributos semelhantes para conceitos diferentes, a alta variação da frequência de atributos de um determinado domínio, a existência de atributos sem rótulos ou com vários campos, a diversidade de restrições de valores entre os atributos, dentre outros.

A Figura 1 exemplifica a dificuldade para execução do *matching* em formulários Web no domínio de revendas de automóveis. Percebe-se que diferentes rótulos são utilizados para representar o mesmo conceito: *by brand*, *car make-model*, *model* e *manufacturer*, ou rótulos com grafia similar podem representar conceitos diferentes: *year of manufacture* e *manufacturer*.

Figura 1: Matching de formulários Web no domínio de automóveis [Freire et al. 2010].

As vantagens da utilização de um processo de *matching* é a visão centralizada de esquemas de formulários Web, a distribuição de consultas e a possibilidade de integrar diversos bancos de dados. Com o suporte de uma atividade de *matching* é possível para o usuário definir filtros de busca sobre uma visão mais ampla do domínio ou ainda especificar uma consulta em um determinado formulário Web que pode ser propagada para outros formulários com atributos similares.

Com o objetivo de contribuir com as pesquisas que necessitam da execução de *matching* de dados na *Deep Web*, este artigo apresenta uma revisão a respeito de algumas abordagens sobre esse assunto de *matching* que pode ocorrer em formulários Web, dados na *Deep Web* e em esquemas de dados. Primeiramente, alguns trabalhos relacionados são descritos brevemente. Na sequência, é apresentada uma análise comparativa de características destes trabalhos, contribuindo com o estabelecimento de um estado da arte sobre o assunto. Por fim, sugere-se alguns pontos que estão em aberto e que podem ser utilizados para trabalhos futuros.

Cabe salientar que, durante a revisão bibliográfica, identificou-se alguns *surveys* sobre *matching* de esquemas de dados. ([Shvaiko and Euzenat 2005], [Bernstein et al. 2011] e [Dorneles et al. 2011]). Entretanto, este trabalho apresenta o estado da arte para trabalhos mais recentes e focados em formulários Web.

2. Trabalhos Relacionados

Esta seção revisa e comenta as principais características e diferenciais de alguns trabalhos relacionados à problemática de *matching* de formulários *Web*.

A proposta de [Freire et al. 2010] apresenta o sistema PruSM (*Prudent Schema Matching*). O sistema recebe os dados de uma coleção de formulários previamente coletados. Estes formulários servem como entrada para um módulo de agregação (*Aggregation*) que, utiliza técnicas para a preparação dos dados, como por exemplo a remoção de *stop words*. A próxima etapa é o módulo de Descoberta de Relacionamentos (*Matching Discovery*), onde são analisados os relacionamentos e a frequência entre os atributos. Primeiramente, é calculada a similaridade do rótulo, utilizando a distância do cosseno entre os termos dos vetores de seus rótulos, onde se fornece uma medida da importância do termo dentro do conjunto baseado na sua frequência no conjunto como, por exemplo, *manufacturer* e *year of manufacturer*. Na sequência, calcula-se a similaridade entre os valores do atributo. Por fim, é realizada a correlação dos valores dos atributos do conjunto, para detectar casos como, por exemplo, *Make* e *Brand*, que apesar de possuírem grafia diferente, possuem o mesmo significado. Ao final do processo, os atributos com baixa frequência têm seus pesos recalculados através de um algoritmo chamado STF (*Singular Token Frequency*). Em seguida, aplica-se um algoritmo conhecido como 1NN (*1-Nearest-Neighbor Clustering*), que agrupa o atributo raro ao seu vizinho mais próximo de maior frequência. Estes passos geram um novo cluster de atributos considerados representativos. Finalmente, incorpora-se este novo cluster ao cluster confiável, utilizando um algoritmo denominado HAC (*Hierarchical Agglomerative Clustering*), onde os atributos são representados em uma estrutura de árvore e são aglomerados aos mais próximos sucessivamente. Ocorre então a identificação de conjuntos de alta confiança, ou seja, clusters de atributos afins.

O *DynaBot* [Rocco et al. 2005] é definido como um sistema de descoberta de fontes de dados orientado à *Deep Web*. Ele possui arquitetura modular com suporte a *focused crawlers*, com ênfase em *matching*, sondagem e ranqueamento de páginas rastreadas. O *DynaBot* utiliza um componente plugável denominado analisador semântico específico, o qual analisa o banco de dados candidato e possui um Combinador de Classe de Serviço (*Service Class Matcher*). A descrição de classe serviço (*service class description*) é uma descrição abstrata utilizada para determinar os recursos das fontes encontradas e relacionar fontes da *Deep Web*. Uma classe de serviço (*service class*) é um conjunto de fontes da *Deep Web* que apresentam a mesma funcionalidade, fornecendo uma descrição geral dos dados. A descrição da classe de serviço articula uma interface de consulta e fornece uma referência para determinar a relevância de um serviço específico para uma classe de serviço. A descrição da classe de serviço é composta inicialmente por um desenvolvedor ou usuário do serviço e ainda pode ser revisto através de algoritmos de aprendizagem automatizados embutidos na execução do *matching*. Quando o rastreador cruza com a interface de consulta, este invoca o combinador de classe de serviço para determinar se a fonte é de fato candidata à classe de serviço. Em caso positivo, ela é classificada como membro de uma classe de serviço baseada na descrição de classe de serviço.

O trabalho apresentado por [Hong et al. 2010] propõe uma abordagem que combina múltiplos *matchers* utilizando a denominada Teoria de Evidência de *Dempster-Shafer*. Esta teoria matemática fornece mecanismos para a representação da incerteza, imprecisão e informação incompleta. Ela combina evidências de diferentes fontes para alcançar um grau de confiança, considerando todas as evidências apresentadas. Quatro *matchers* individuais são utilizados, dos quais três são baseados nas características da semântica dos rótulos e o último utiliza o tipo de dado dos atributos para o *matching*. O primeiro *matcher* utiliza o dicionário semântico *WordNet*² que ajuda no cálculo de similaridade entre os termos que possuem o mesmo significado, como por exemplo, carro e veículo. O segundo utiliza distância de edição para medir a similaridade entre duas *strings*, ou seja, quantas transformações são necessárias para transformar uma *string* na outra. No terceiro *matcher*, a medida de similaridade é definida através do total e da ordem dos caracteres comum a ambas. Por fim, o último *matcher* define se dois tipos de dados são compatíveis, ou seja, se são do mesmo tipo ou se um é subordinado ao outro, como por exemplo, quando se depara com uma data pode-se encontrar, dia, mês e ano. Experimentos com um conjunto de 88 formulários distribuídos em 6 domínios obtiveram um valor de precisão de 96% no domínio de *Autos* e a sua menor percentagem foi no domínio de *Jobs* com 91,9%. Apesar dos excelentes resultados, encontra-se nesta abordagem uma limitação pelo fato do sistema utilizar o recurso do dicionário semântico comparando somente duas *strings* por vez.

Na proposta de [Wang et al. 2004], primeiramente define-se um esquema como um conjunto de atributos, cada um dos quais com um único significado. Nessa abordagem, os bancos de dados *Web* podem ser categorizados em um domínio específico, como por exemplo, venda de livros. Em cada domínio específico, existe um esquema global, o esquema de interface e o esquema de resultado. O esquema global representa o conhecimento geral sobre o domínio e consiste dos atributos representativos dos objetos dos dados. O esquema de interface consiste de atributos sobre os quais os usuários podem consultar. Já o esquema de resultado consiste de atributos que os usuários podem navegar. O *matching* entre dois esquemas determina que certos atributos de um esquema correspondem semanticamente a certos atributos do outro esquema. O *matching* pode ocorrer nas seguintes formas: entre esquema global e esquema de interface, entre esquema global e esquema de resultado, entre esquema de interface e esquema de resultado, esquema de resultado e esquema de resultado e por fim, esquema de interface e esquema de interface. Considera-se cada atributo de um esquema de interface ou de resultado como um "documento" e cada atributo do esquema global como um "conceito", gera-se uma matriz sendo que cada linha representa um vetor de documento correspondente. Portanto, calcula-se a similaridade entre os atributos de diferentes esquemas. Essa proposta descreve as relações de correspondência entre diferentes esquemas de bancos de dados da *Web* em um domínio específico, mas também fornece uma visão global sobre como reforçar a precisão da correspondência através da realização de vários tipos de *matching* de esquema simultaneamente.

A análise de um amplo conjunto de características é realizada por [Silva et al. 2007]. As informações extraídas dos formulários e do *site* em geral formam um conjunto com os termos e atributos mais significativos do domínio, sendo esse conjunto utilizado para a criação de um modelo base. Este modelo consiste de uma tripla FP

² <http://wordnet.princeton.edu/>

(*Backlink*, PC, FC), onde FP corresponde ao *Form Page*, PC a *Page Contents*, FC a *Form Contents* e *backlink* consiste de uma lista de URLs que apontam para um FP. Um algoritmo denominado CAFC-CH (*Context-Aware Form Clustering – Content Hub*) utiliza o modelo anteriormente definido para a determinação de centroides iniciais que são utilizados para a clusterização, sendo que quanto mais relevantes os termos melhor será o resultado. Os centroides são determinados com o cálculo da média dos pesos dos termos das páginas. O algoritmo define o domínio ao qual o formulário pertence, além de utilizar as informações visíveis do site, que são automaticamente obtidas através da remoção das *tags* do HTML para a obtenção dos termos. Ele utiliza a métrica TF-IDF para a definição dos pesos dos atributos e dependendo da localização do atributo o sistema atribui pesos maiores para os termos mais significativos. Com a utilização do modelo base e com as triplas que ajudam na determinação do modelo, em 454 *form pages* apenas 17 foram classificadas incorretamente. Porém, apesar da utilização de um modelo base, a abordagem não possui técnicas para a atualização deste modelo, o que acaba sendo um grande problema devido à dinamicidade das páginas *Web*. Esse trabalho possui um diferencial que é a atribuição de pesos de acordo com a importância de cada termo recuperado do site. Este processo de determinação/resgate de termos relevantes é um tema atual de pesquisa na área [He et al. Chang 2004]. Nele, *crawlers* fazem uso da localização e formatação dos termos no *site*, como por exemplo, o tamanho e estilo da fonte, entre outras características, para a determinação de pesos para cada termo existente. Esses termos, quando relevantes, são de grande valia no processo de execução do *matching*, pois eles podem ajudar na determinação de centroides para a formação de clusters ou até na formação de modelos para a comparação dos atributos.

Segundo [Freire et al. 2008], uma boa solução para a execução do *matching* começa com uma boa extração dos rótulos dos formulários, formando um conjunto de tuplas (rótulos-atributos) manualmente derivado para uma dada amostra de formulários. Estes são utilizados para classificar os mapeamentos candidatos gerados automaticamente: caso pertençam ao conjunto, eles são marcados como amostras positivas; caso contrário são marcados como amostras negativas. Essas tuplas são utilizadas em um primeiro classificador *Naïve Bayes*. O classificador *Naïve Bayes* emprega um raciocínio probabilístico que engloba teoria de grafos para o estabelecimento de relações entre sentenças e ainda teoria de probabilidades para a atribuição de níveis de confiabilidade. Ele tem uma parte estrutural refletindo relações causais entre as variáveis de entrada (*inputs*) e a variável de saída (*output*) do sistema e valores de probabilidade refletindo a força da relação. Após passarem pelo classificador *Naïve Bayes*, as tuplas passam por um segundo classificador que é uma árvore de decisão. Os atributos não classificados retornam ao início do processo, sendo novamente comparados com os conjuntos já classificados, ou seja, o conjunto é atualizado para uma nova tentativa de classificação. Esse sistema possui a desvantagem de requerer o processamento manual, mas possui a vantagem de ter um modelo de comparação que é atualizado dinamicamente com os atributos que já foram processados. Utilizando aproximadamente 170 amostras para os experimentos, o valor de *recall* do classificador *Naïve Bayes* para a classificação de atributos negativos no conjunto, mas sem remover os positivos, foi de 98%.

Em [Do and Rahm 2002] utilizam-se *matchers* simples e *matchers* híbridos. Nos *matchers* simples, os valores representam a fonte para a avaliação da similaridade entre os dados do esquema. Isto pode ser feito sintaticamente, comparando as *strings*, ou semanticamente, comparando seus significados. Os *matchers* híbridos usam uma combinação fixa de *matchers* simples e outros híbridos para obter maior precisão nos valores de similaridade. O *matcher* híbrido combina a comparação de *strings* e a hierarquia dos nomes. Primeiramente, constrói-se uma frase concatenando todos os nomes dos atributos em uma única seqüência. A seguir, a *string* e a estrutura são analisadas, como por exemplo, *PurchaseOrder.ShipTo.Street* e *PurchaseOrder.shipToStreet*. Em seus experimentos, este trabalho obteve uma média de precisão acima de 0,9 combinando as suas duas soluções de *matching*.

Em [Chang and Cheng 2007] um mecanismo de busca pesquisa entidades e executa o *matching* das entidades. No contexto deste trabalho, uma entidade representa determinados tipos de dados, como por exemplo, um número de telefone, um nome ou uma data. Realiza-se uma consulta com as entidades alvo e após a busca, cada resultado corresponde a uma tupla. Uma tupla pode conter uma ou mais instâncias, cada uma associada a um tipo de entidade. Por exemplo, (David, david@hotmail.com) é uma tupla do tipo (#name, #email). O objetivo é encontrar, a partir das tuplas correspondentes na ordem de classificação, quais correspondem a consulta. A ordem de classificação é gerada a partir da frequência da tupla, frequência da instância e a incerteza da instância da entidade. Na sua essência, o sistema assume a "integração em um sentido probabilístico". Dadas entidades independentes com probabilidades, a pesquisa encontra casamentos com outras entidades.

A proposta de [Hao et al. 2010] emprega uma abordagem de integração *Local as View* (LAV) com suporte de ontologia. Neste sistema, o método de mapeamento e *matching* entre o esquema do mediador e as visões locais da *Deep Web* é gerada pela afirmação das relações dos grupos semânticos de acordo com uma ontologia. No LAV, o mapeamento associa cada atributo do esquema a uma consulta. Do ponto de vista da modelagem, a abordagem LAV é baseada na ideia de que o conteúdo de cada elemento de uma fonte local deve ser caracterizada em termos de uma visão sobre o esquema global. Esta ideia é eficaz sempre que o sistema de *matching* de dados é baseado em um esquema global que é estável e bem estabelecido.

A proposta de [Bhattacharjee and Jamil 2009] apresenta um sistema de *matching* de esquemas chamada de *OntoMatch*, composto por métodos que definem os algoritmos de *matching*. Estes métodos são utilizados para projetar uma função para a seleção e ordenação dos *matchers* propostos. Um destes é o *Structure Matcher*, utilizado para estruturas aninhadas e que mede a similaridade entre subárvores enraizadas nos nós correspondentes aos valores. Quanto mais as duas subárvores são parecidas, mais próximos os valores são. A semelhança das duas subárvores é definida por uma combinação de características analisadas de forma recursiva, entre elas as relações de cardinalidade e algumas restrições. Já o *Synonym Matcher* é uma caracterização que identifica se dois valores são sinônimos entre si.

3. Comparativo

O quadro comparativo na Tabela 1 mostra algumas características relevantes para a comparação das abordagens descritas na seção 2.

Tabela 1: Comparativo das abordagens.

Trabalho	Técnica	Itens de dados utilizados	Análise sintática e/ou semântica	Utiliza intervenção manual?
[Freire et al. 2010]	Correlação de atributos	Atributos	Ambas	Não
[Rocco et al. 2005]	Descrição com a funcionalidade	Atributos	Semântica	Opcional
[Hong et al. 2010]	<i>Dempster-Shafer</i> e dicionário semântico	Rótulos e tipos de dados	Ambas	Não
[Hao et al. 2010]	Grupos semânticos e ontologia	Atributos	Ambas	Não
[Silva et al. 2007]	Clusters	Atributos	Ambas	Não
[Freire et al. 2008]	<i>Naïve Bayes</i> e Árvore de decisão	Atributos	Ambas	Sim
[Do and Rahm 2002]	Comparação de <i>strings</i> e hierarquia dos nomes	Valores	Ambas	Opcional
[Bhattacharjee and Jamil 2009]	Ontologia	Valores	Ambas	Não
[Wang et al. 2004]	Definição de esquemas	Atributos	Semântica	Sim
[Chang and Cheng 2007]	Holística	Tipos de dados	Ambas	Não

Com relação ao critério *Técnica*, verifica-se uma grande diversidade de abordagens utilizadas para a execução do *matching*, observando-se que é uma linha de pesquisa bastante diversificada e que está aberta para novos estudos, pois muitas técnicas apresentam vantagens e desvantagens e não existe uma unanimidade para a utilização de determinada técnica.

Com relação ao tópico *Itens de Dados Utilizados* em cada abordagem, verifica-se devido a um bom desempenho em trabalhos mais abrangentes, uma tendência na utilização das informações dos atributos (rótulos e valores), para a execução do *matching*, ao invés de somente os rótulos ou só os valores. Eles levam vantagem pelo fato de terem mais informações disponíveis para a execução do processo de casamento, já que mais informações podem contribuir para a confirmação de uma alta similaridade entre os dados.

Outra faceta é o *Tipo de Análise* realizado pelo trabalho, que pode abranger somente uma análise sintática, semântica ou ambas juntas. Observa-se a tendência da utilização da análise sintática e semântica na maioria dos trabalhos, pois geram mais variáveis para a execução do *matching*. Cabe observar a utilização de uma ontologia nos trabalhos ([Hao et al. 2010] e [Bhattacharjee and Jamil 2009]) como apoio à determinação da importância de um atributo. A utilização de ontologia não é plenamente aceita entre os autores. Essa divergência ocorre devido ao grande esforço necessário para construí-las e mantê-las.

Característica presente em poucos trabalhos é a intervenção manual. Em ([Freire et al. 2008] e [Wang et al. 2004]) esta característica faz parte do processo. Em ([Rocco et al. 2005] e [Do and Rahm 2002]) pode ser opcional, possibilitando a intervenção em algumas etapas do processo no qual o usuário pode dar um *feedback*, por exemplo, para fornecer manualmente correspondências ou para confirmar ou rejeitar correspondências que são automaticamente propostas pelo sistema. Apesar de melhorar o resultado do *matching*, não se mostra uma solução atrativa considerando a intervenção humana necessária para a execução do processo.

4. Considerações Finais

O volume de dados na *Deep Web* cresce a cada dia e a procura de mecanismos robustos para o *matching* destes dados é relevante, considerando a necessidade dos usuários de encontrar dados e informações relevantes para as suas necessidades, como por exemplo, uma oferta de emprego ou um veículo que deseja comprar. Este artigo apresenta o estado da arte referente à execução de *matching* focado na *Deep Web*, bem como uma análise comparativa de algumas de suas características.

Algumas considerações e sugestões de trabalhos futuros podem ser levantadas a partir do comparativo apresentado. Primeiramente, uma análise mais ampla do conjunto de dados disponíveis nos *sites* a fim de encontrar semelhanças que ajudem na determinação do domínio dos formulários, como por exemplo, descobrir o nome das imagens no site, a que assunto se refere às propagandas, o nome do *site*, para que estas informações também se tornem uma premissa importante para a execução do casamento dos dados. Esse tipo de abordagem poderia ser melhor explorada.

Quanto à utilização de ontologias no processo de *matching* deve-se considerar o custo inicial de criá-la em virtude das especificidades de cada domínio. Entretanto, quando a ontologia estiver pronta e consideravelmente robusta, ela pode ajudar na decisão da importância de um determinado atributo no domínio. Após essa decisão, pode-se atribuir pesos aos atributos determinando o quanto importante ele é, ou até mesmo descartá-lo, para que não interfira negativamente no resultado do *matching*.

Um aspecto que é pouco comentado nos trabalhos é o tempo de processamento das técnicas de *matching*. Nesse sentido, devido ao crescimento na utilização de computação distribuída é interessante a aplicação de técnicas de multiprocessamento para a execução dos casamentos, visando diminuir o *overhead* durante a execução do *matching* e possibilitando um maior número de comparação entre os dados. Além disso, outros pontos relevantes para trabalhos futuros são:

- Adoção de técnicas de pré-processamento, como limpeza dos dados e análise de restrições de valor dos atributos dos formulários coletados para posteriormente executar o *matching* a fim de melhorar a acurácia do resultado;
- Utilização de técnicas de inteligência artificial para a simulação e determinação de centroides e distribuição de clusters de atributos afins, bem como técnicas de *machine learning* para a classificação dos atributos;
- Consideração dos dados escondidos nos bancos de dados como mais um parâmetro para determinar a similaridade dos formulários que permitem o acesso a eles.

Esse estudo foi realizado a fim de identificar contribuições que possam embasar uma proposta de pesquisa em andamento na área de *matching* de dados na *Deep Web* no Programa de Pós Graduação em Ciência da Computação da Universidade Federal de Santa Catarina - UFSC.

5. Referências

- Bergman, Michael K. (2001). “The Deep Web: Surfacing Hidden Value”, In BrightPlanet.
- Bernstein, P. A., Madhavan, J., Rahm, E. (2011). “Generic Schema Matching, Ten Years Later”, In: Proceedings Very Large Data Base (VLDB).
- Bhattacharjee, A. Jamil, H. (2009). “OntoMatch: A Monotonically Improving Schema Matching System for Autonomous Data Integration”, In: IEEE International Conference on Information Reuse and Integration (IRI’09).
- Chang, K. C.-C and Cheng, T. (2007) “Entity Search Engine: Towards Agile Best-Effort Information Integration over the Web”, In: Proceedings of the Second Conference on Innovative Data Systems Research (CIDR).
- Do, H-H., Rahm, E. (2002). “COMA - A system for flexible combination of schema matching approaches”, In: Very Large Data Base (VLDB).
- Dorneles, C. F., Gonçalvez, R., Mello, R. S. (2011). “Approximate data instance matching: a survey. Knowledge and Information Systems”, In: Knowledge and Information Systems, volume 27, number 1, p. 1-21.

- Freire, J., Nguyen, H. and Nguyen, T. (2008). "Learning to Extract Form Labels", In: Proceedings of the Very Large Data Base (VLDB), p 684-694.
- Freire, J., Nguyen, H. and Nguyen, T. (2010). "PruSM: A Prudent Schema Matching Approach for Web Forms", In: Conference on Information and Knowledge Management (CIKM), p. 1385-1388.
- Halevy, A., Madhavan, J., Afanasiev, L., Antova, L. (2009). "Harnessing the Deep Web: Present and Future", In: Conference on Innovative Data Systems Research (CIDR).
- Hao, L., Wan-Li,Z., Fei, R., (2010) "Describing the Semantic Relation of the Deep Web Query Interfaces Using Ontology Extended LAV", In: Journal Of Software, Vol. 5, n°. 1, p. 89-98.
- He, B., Tao, T. and Chang, K. C.-C. (2004). "Organizing structured web sources by query schemas: a clustering approach", In: Conference on Information and Knowledge Management (CIKM), p. 22-31.
- Hong, J., He, Z., and Bell, David A. (2010). "An evidential approach to query interface matching on the deepWeb", In: Journal Information Systems, p. 140-148.
- Rocco, D., Caverlee, J., Liu, L., Critchlow, T. (2005). "Exploiting the deep web with dynabot: matching, probing, and ranking", In: Poster Proceedings of the 14th International World Wide Web Conference (WWW F05), p. 1174-1175.
- Shvaiko, P. and Euzenat, J. (2005) "A Survey of Schema-based Matching Approaches", In: Journal on Data Semantics (JoDS) - IV.
- Silva, A., Freire, J. and Barbosa, L. (2007). "Organizing Hidden-Web Databases by Clustering Visible Web Documents", In: IEEE International Conference on Semantic Computing (IEEE), p.326-335.
- Wang, J., Wen, J-R., Lochovsky, F., Ma, W-Y. (2004). "Instance-based Schema Matching for Web Databases by Domain-specific Query Probing", In: Very Large Data Base (VLDB), p. 408-419.

Uma Ferramenta para Coleta e Análise de Dados do Segmento Naval

Tiago F. Otero, Lucas R. de Farias, André P. Vargas, Eduardo N. Borges

Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brasil

{tiagootero, lfarias, andre.prisco, eduardoborges}@furg.br

Resumo. *Este artigo descreve uma ferramenta para análise de dados oriundos da construção naval que coleta os dados da Web, extrai informações dos dados coletados e as analisa em busca de inconsistências. Foi utilizado como estudo de caso o Catálogo Navipeças, mantido pela Organização Nacional da Indústria do Petróleo. Os resultados mostram empresas com inconsistências nos relacionamentos de produção de bens e fornecimento de serviços.*

1. Introdução

A perspectiva de grande crescimento do setor naval no Brasil exige ações na direção de se consolidar e ampliar a participação da indústria nacional fornecedora de bens e serviços para este setor. Neste momento há centenas de embarcações encomendadas ou em construção nos estaleiros brasileiros. Novos estaleiros e fornecedores nacionais estão surgindo, embalados pelo crescimento da indústria naval e do setor de petróleo e gás. Neste contexto, surge a necessidade da utilização de ferramentas de TI a fim de colaborar na eficiência da cadeia produtiva, mais especificamente, na viabilização da interação entre companhias de construção naval e fornecedores de bens e serviços.

O Catálogo Navipeças, mantido pela Organização Nacional da Indústria do Petróleo (ONIP), armazena informações qualificadas de empresas nacionais, fabricantes e prestadores de serviços, diretamente ligados à construção e reparação naval, dando visibilidade aos fornecedores cadastrados. Entretanto, o cadastro de novas empresas no catálogo é feito manualmente através de formulários, o que pode gerar determinadas inconsistências dos dados provocadas por erro humano ou pela simples desatualização. Outro problema se refere ao fato do catálogo não cobrir todas as empresas brasileiras com condições de fornecer bens e serviços aos estaleiros. Além disso, o catálogo é disponibilizado à comunidade apenas através do seu *site* na Web [ONIP 2012], em formato HTML simples, tornando difícil a integração do mesmo com outras fontes de dados do setor.

Diante dos problemas apontados, surge a necessidade da criação de um sistema de *software* que cole e analise dados oriundos da indústria naval com dois principais objetivos: (i) encontrar inconsistências nas bases de dados, principalmente quanto aos relacionamentos entre empresas, bens e serviços; (ii) integrar diversas fontes a fim de fornecer uma base de dados mais abrangente em relação à cadeia produtiva do setor naval. Este trabalho apresenta a arquitetura de uma ferramenta que satisfaz o primeiro objetivo, ou seja, mapear as inconsistências do Catálogo Navipeças.

2. Extração de Informações e Coleta de Dados na Web

Extração de Informações pode ser definida como a tarefa de identificar informação relevante para o usuário, geralmente organizada de forma estruturada, em repositórios de dados com pouca ou nenhuma estrutura, como documentos semiestruturados e texto em linguagem natural [Appelt 1999]. Para melhorar o desempenho da extração de informações da *Web*, é necessário realizar a coleta das páginas de interesse.

Coletores *Web*, conhecidos também por *crawlers* ou *spiders*, são aplicações que percorrem a *Web* de forma automática, armazenando as páginas visitadas bem como os recursos disponibilizados pelas mesmas. Os coletores buscam encontrar a maior quantidade de páginas *Web* do universo desejado com o menor custo computacional. Coletores que restringem as páginas de interesse com base no conteúdo podem ser classificados como específicos ou focados [Dong, Hussain and Chang 2008].

3. Arquitetura da Ferramenta

A Figura 1 apresenta a arquitetura da ferramenta proposta, que é dividida em quatro componentes principais. O *Coletor* recupera os dados do Catálogo Navipeças através dos arquivos HTML disponibilizados na *Web* e os armazena em um banco de dados local. O *Extrator de Entidades* analisa os arquivos coletados e extrai os dados que compõem as empresas, bens e serviços do catálogo. Também são extraídas as categorias de bens e serviços. O *Extrator de Relacionamentos* analisa os arquivos coletados em conjunto com as entidades já identificadas inferindo três tipos de relacionamentos: produção de bens, fornecimento de serviços, e categorias de bens ou serviços. Por fim, o *Gerador de Relatórios* analisa semanticamente os dados extraídos, produzindo o *Relatório de Inconsistências*.

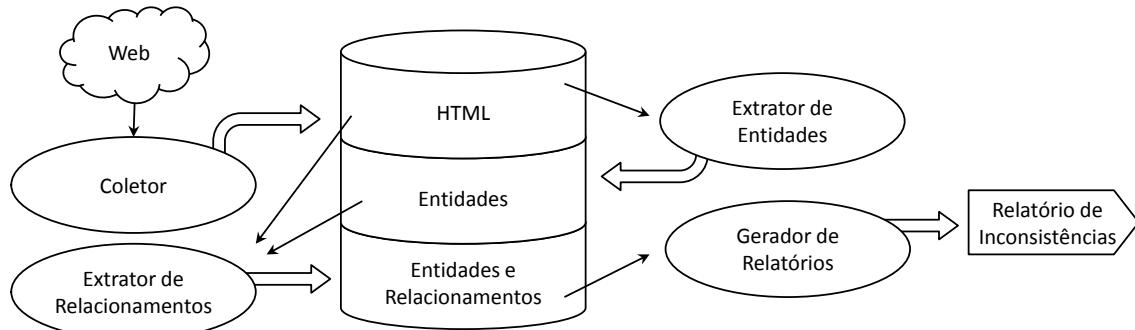


Figura 1. Arquitetura da ferramenta para análise de inconsistências

O modelo de dados – no nível lógico – do banco de dados local utilizado como entrada do *Gerador de Relatórios* pode ser visto na Figura 2. Bens e serviços são caracterizados por uma descrição, pertencem a apenas uma categoria e podem ser produzidos ou fornecidos por quaisquer empresas. Empresas são caracterizadas por diversos atributos como CNPJ, Razão Social, Nome Fantasia, etc. Os relacionamentos *Produção* e *Fornecimento* indicam quais empresas estão aptas a fornecer um determinado serviço ou produzir um determinado bem. Além dos atributos extraídos dos documentos HTML, foram adicionados às entidades outros dois: *ID*, para identificação única de cada instância e *Fonte*, que corresponde ao URL da página *Web* que contém o dado extraído. O atributo *Fonte* também foi adicionado a todos os relacionamentos.

Ainda foi adicionado o atributo *Candidato* que tem por função indicar se os dados extraídos foram confirmados através de referências em outras páginas do catálogo ou apenas definem bens, empresas e serviços em potencial, ou seja, dados candidatos a identificar as entidades.

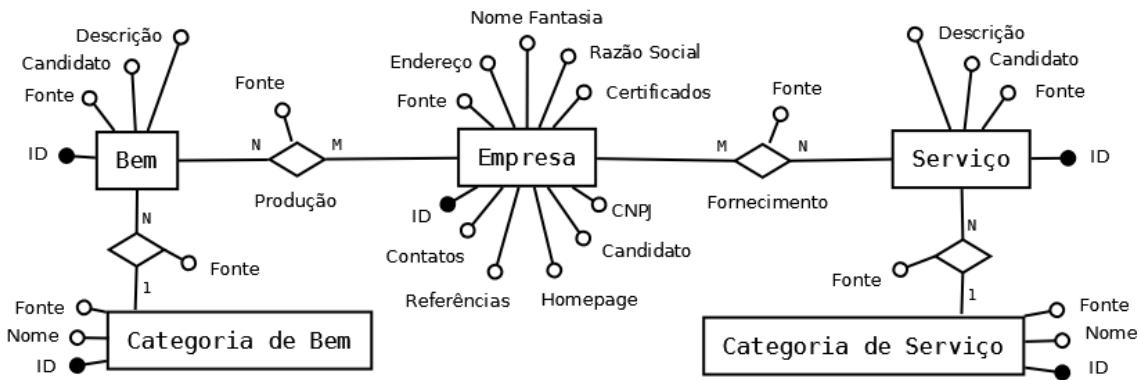


Figura 2. Modelo ER da ferramenta para análise de inconsistências

4. Implementação da Ferramenta

Esta seção descreve a construção dos componentes apresentados na arquitetura. O *Coletor* utiliza a ferramenta *Wget* [Niksic 2011], padrão para coleta de *sites Web* em ambientes Unix. Os extratores de entidades e relacionamentos foram desenvolvidos utilizando a linguagem PHP. O banco de dados foi implementado utilizando o modelo relacional no SGBD PostgreSQL. Um *script* de Shell [Cooper 2011] integra os componentes.

As páginas *Web* do catálogo Navipeças são organizadas em três seções principais. A primeira seção contém um conjunto de empresas que compõem o catálogo. As outras duas definem os bens produzidos e os serviços oferecidos pelas empresas. Uma vez coletadas as páginas, o *Extrator de Entidades* analisa a formatação HTML em busca de padrões que identificam as entidades de interesse. Além do conteúdo dos arquivos, o endereço de cada página também fornece informação. Os URL são formados por uma nomenclatura padronizada de parâmetros enviados por GET, os quais auxiliam na identificação das instâncias, sejam estas empresas, bens ou serviços.

Para cada empresa há uma página HTML com quatro grupos de informações: endereço, contatos, bens produzidos e serviços oferecidos. Estas informações estão bem definidas através de comentários únicos, o que facilita a separação de cada porção de código. As tuplas extraídas são então adicionadas à relação *Empresa* no banco de dados. Os bens são apresentados em listas não ordenadas organizadas em categorias. As informações de interesse são extraídas a partir de cada item das listas. Como não há referência explícita aos identificadores das categorias, o *Extrator de Entidades* utiliza um número inteiro com autoincremento. O *Extrator de Relacionamentos* adiciona o ID da categoria aos atributos de cada bem identificado para mapear o relacionamento entre as instâncias. As tuplas extraídas são então adicionadas às relações *Bens* e *Categoria de Bem*. Esse processo é repetido de forma muito similar para o último grupo da página HTML (serviços).

O *Extrator de Relacionamentos* reanalisa os arquivos HTML em busca das relações entre as entidades identificadas. Quando encontradas, essas relações são mapeadas para o modelo relacional. Por fim, o componente consulta o banco de dados para obter dois tipos principais de inconsistências: entidades candidatas (EC) e relacionamentos candidatos (RC). Por exemplo, empresas inexistentes na seção “Empresas”, mas referenciadas na seção “Serviços” são rotuladas como candidatas porque não têm uma página com dados de identificação como CNPJ, endereço, etc. Relacionamentos entre empresas da seção “Empresas”, referenciadas na seção “Bens”, podem ser rotulados como candidatos se não houver informação explícita da produção desses bens na página da empresa. Empresas, bens e serviços passam de candidatos para confirmados quando os relacionamentos *Produção* e *Fornecimento* têm duas fontes (seções distintas).

Os experimentos realizados são sumarizados na Tabela 1 que apresenta, para cada coleta, o número de instâncias coletadas de cada entidade, o tipo de inconsistência, a entidade envolvida e a quantidade de inconsistências identificadas.

Tabela 1. Inconsistências identificadas no Catálogo Navipeças.

Coleta	Empresas	Bens	Serviços	Tipo	Entidade	Quantidade
18/05/2011	205	746	174	EC	Bem	12
				EC	Serviço	2
05/11/2010	125	745	172	RC	Bem	315
				RC	Serviço	29

5. Conclusões

Este artigo apresentou a arquitetura de uma ferramenta que identifica inconsistências em bases de dados do setor naval. Foi utilizado como estudo de caso o Catálogo Navipeças, entretanto a ferramenta pode ser adaptada para coletar dados de outras bases do segmento naval com pouco esforço de programação. Como trabalhos futuros podem ser destacados: (i) o uso de algoritmos de mineração de dados para automatizar o reconhecimento de padrões que atualmente é realizado pelo usuário especialista. (ii) a generalização da ferramenta e a coleta de outras bases de dados do setor naval; (iii) a integração dessas bases com o Catálogo Navipeças; (iv) a identificação e remoção de registros duplicados e (v) a busca na *Web* por páginas de empresas candidatas a participar do catálogo integrado. Uma base de dados mais abrangente contribuirá significativamente para o aumento da competitividade do setor naval, garantindo ampla igualdade de oportunidades para os fornecedores nacionais.

Referências

- Appelt, D. (1999). Introduction to information extraction. In *AI Commun*,12(3):161-172.
- Cooper, Mendel (2012), Advanced Bash-Scripting Guide, version 6.3, url: <http://tldp.org/guides.html>.
- Dong, H., Hussain, F., Chang, E. (2008). A survey in semantic web technologies-inspired focused crawlers. In Proceedings of the International Conference on Digital Information Management, p. 934-936.
- Niksic, H. (2012) GNU Wget 1.12, url: <http://www.gnu.org/software/wget/manual>.
- ONIP (2012). Catálogo Navipeças, url: <http://www.onip.org.br/cadastrados/navipecas>.

Estratégias de Persistência de *Clusters* em uma Técnica de Casamento por Similaridade para *Web Forms*¹

Filipe Roberto Silva, Ronaldo dos Santos Mello

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

{filipesilva.sc, ronaldo}@inf.ufsc.br

Abstract. Most data available currently in Web are just accessible by queries in Web forms. These hidden data bases that are just revealed in response to those queries are called Deep Web. One way to search contents in the Deep Web is to provide to the user relevant Web forms to his/her interests (car dealers, airfare, etc) and the own user can formulate his/her queries. This paper presents and evaluates strategies for clusters' persistence in a Web forms' similarity search system called WF-Sim with the purpose of identifying similar Web forms efficiently and effectively.

Resumo. Grande parte dos dados disponíveis atualmente na Web só é acessível através de consultas formuladas sobre Web forms. Estes bancos de dados “escondidos” e revelados apenas como resposta a estas consultas são denominados de Deep Web. Uma das formas de busca a conteúdos na Deep Web é fornecer ao usuário Web forms relevantes para os seus interesses (revendas de automóveis, passagens aéreas, etc) e o próprio usuário realiza suas buscas. Este trabalho apresenta e avalia estratégias de persistência de clusters de afinidade em um mecanismo de casamento (matching) de Web forms por similaridade chamado WF-Sim, de forma que seja possível identificar Web forms semelhantes entre si de forma eficiente e eficaz.

1. Introdução

A Internet está presente na vida de muitas pessoas. Qualquer dúvida, pesquisa, notícia e etc, pode ser alcançada através da Internet utilizando as populares máquinas de busca, como *Google* ou *Bing*. Porém, o que poucos sabem é que essas buscas atingem apenas a superfície da Web. Grande parte do conteúdo na Web encontra-se escondido em bancos de dados que não são acessados por tais máquinas de busca. A esse conteúdo é dado o nome de *Deep Web* ou Web oculta (*Hidden Web*)². O conteúdo da Deep Web é revelado em páginas dinâmicas, geradas automaticamente conforme requisições do usuário realizadas sobre formulários na Web (*Web forms*). Assim sendo, tais páginas não podem ser indexadas pelas máquinas de busca.

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

² <http://brightplanet.com/images/uploads/12550176481-deepWebwhitepaper.pdf>

Dada a dificuldade para a descoberta e indexação de todo o conteúdo da *Deep Web*, uma abordagem possível de busca e que é o foco deste trabalho, é pesquisar somente por campos de *Web forms*, porém indexando os campos destes *Web forms* e permitindo a busca por outros *Web forms* similares, ou seja, considerando um processo de *matching* de *Web forms*. Para entender melhor esse processo, é interessante entender melhor como é formado um *Web form*.

Um *Web form* possui diversos campos (campos de texto, *check boxes*, *combo boxes* e etc), que permitem ao usuário preencher digitando ou selecionando um valor. Como é possível observar na Figura 1, os *Web forms* possuem rótulos, que são utilizados para que o usuário saiba o que colocar em cada campo. Esses campos podem conter os já citados valores para seleção, como é possível ver na Figura 1 para o campo ‘*Select what you have:*’, que possui duas possibilidades de preenchimento: ‘*I have a bike*’ e ‘*I have a car*’.

Assim, esses valores podem ser utilizados para a comparação de um *Web form* com outro, tanto os valores de rótulo quanto os valores de preenchimento.

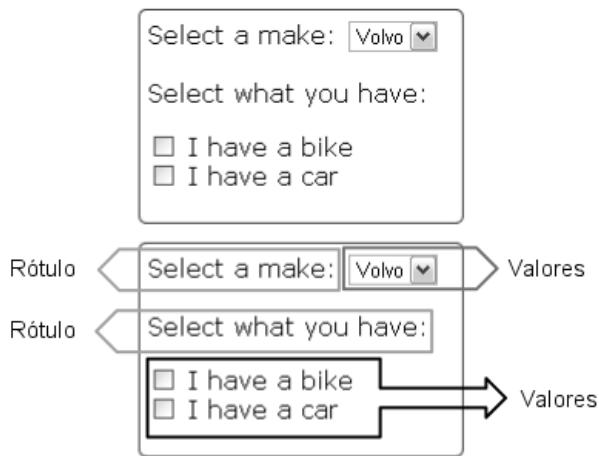


Figura 1. Web form e suas características

Este trabalho é uma continuação dos resultados obtidos com o projeto *WF-Sim*, projeto este que está em andamento e visa tornar possível buscas por similaridade sobre *Web forms*. Com o projeto já é possível agrupar por semelhança *Web forms* previamente extraídos da Web, indexar e a partir de um *Web form* de entrada, encontrar outros semelhantes. Experimentos foram realizados sobre o *WF-Sim* e o mesmo já tem resultados satisfatórios de busca em termos de acurácia (Gonçalves et. al. 2011). Entretanto, existem algumas melhorias a serem feitas e este trabalho visa implementar algumas dessas melhorias.

Especificamente, o objetivo desse trabalho é apresentar otimizações no algoritmo de clusterização de dados de *Web forms* utilizado pelo *WF-Sim* em termos de persistência desses *clusters*. O projeto, como foi implementado, realiza agrupamentos dos *Web forms* a cada vez que é executado. Assim, ele somente guarda os *clusters* em memória, não persistindo os mesmos. A contribuição deste trabalho é a implementação e teste de duas soluções para a persistência desses dados: uma em bancos de dados e outra em arquivos. Cada alternativa foi testada visando verificar a melhor solução para o problema. Resultados de experimentos preliminares são relatados, demonstrando qual das duas estratégias se mostrou mais promissora.

Os demais capítulos detalham o desenvolvimento deste trabalho. O capítulo 2 aborda o projeto *WF-Sim*, com foco na atividade de clusterização. O capítulo 3 mostra como foram

desenvolvidas as persistências dos *clusters* e o capítulo 4 descreve os experimentos preliminares. Por fim, no capítulo 5 são apresentadas as conclusões e atividades futuras.

2. WF-Sim

A descoberta de similaridades entre *Web forms* a partir de *strings* que representam valores em um campo é um grande desafio, já que muitas vezes duas palavras totalmente diferentes significam a mesma coisa, ou ainda duas palavras iguais em contextos diferentes possuem significados diferentes. Um exemplo disso seriam 2 *Web forms* com campos ‘*Manufacturer*’ e ‘*Year of Manufacture*’. Apesar dos nomes semelhantes, eles possuem significados totalmente diferentes.

O trabalho em questão é uma extensão do projeto *WF-Sim* (*Web form Similarity*), um projeto em desenvolvimento pelo Grupo de Banco de Dados da UFSC³ que visa realizar buscas por *Web forms* similares, tentando resolver parte desta problemática. A seguir são abordados alguns detalhes sobre o funcionamento do *WF-Sim*.

2.1 Funcionamento

O *WF-Sim* trata *Web forms* dividindo-os em elementos. Cada elemento representa um campo contendo um rótulo e opcionalmente um conjunto de valores. Essa divisão é exemplificada pela Figura 2 onde, no primeiro elemento, temos o rótulo “*Select a make*” e uma relação de valores iniciada por “Volvo”.

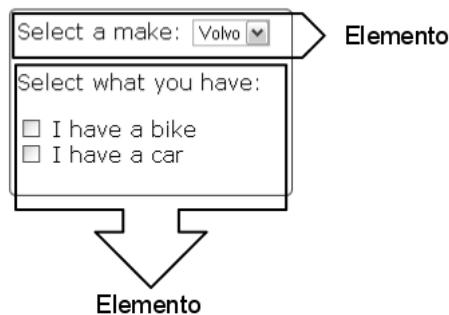


Figura 2. Divisão de um *Web form* em elementos

O projeto visa desenvolver um sistema de busca por similaridade para *Web forms* que é composto por 3 módulos principais. Estes módulos são: *clusterização*, *indexação* e *busca*. Neste sistema, os elementos são clusterizados de acordo com uma métrica de similaridade (ver Seção 2.2) e então indexados. A partir dessa indexação é possível realizar buscas, ou seja, a partir de um *Web form* de entrada são encontrados outros semelhantes. Essa busca, ao invés de comparar elemento por elemento de todos os *Web form*, compara somente os centróides, que são os elementos com maior afinidade com todos os demais elementos no *cluster* e que representam o *cluster* como um todo. Assim, a partir das comparações com os centróides, encontra-se os grupos de elementos semelhantes àquele *Web form* de entrada.

³ <http://www.gbd.inf.ufsc.br>

2.2 Cálculo da Similaridade

O *WF-Sim* utiliza, para o cálculo de similaridade dos rótulos, a métrica *TF-IDF* (Cohen et. al. 2003) e para a similaridade dos valores, a métrica *SubSetSim* (Dorneles 2004), ambas escolhidas por possuírem os melhores resultados se comparados a outras métricas. A partir do cálculo das similaridades dos rótulos e dos valores separadamente, é feita uma média ponderada desses valores, como é possível ver na Equação 1, onde l_1 e l_2 são os rótulos, vl_1 e vl_2 os valores, e *labelSim* e *valuesSim* são os cálculos de similaridade dos rótulos e valores, respectivamente. Por ser uma média ponderada, é possível aplicar pesos diferentes para cada atributo do elemento (*labelWeight* e *valuesWeight*), sendo a soma dos pesos igual a um (1). Vale lembrar que para elementos pertencentes a um mesmo *Web form* não é calculada a similaridade, visto que o objetivo é encontrar similaridades entre *Web forms* diferentes.

$$ElementSim = labelSim(l_1, l_2) * labelWeight + valuesSim(vl_1, vl_2) * valuesWeight \quad (1)$$

2.3 Clusterização

A *clusterização* dos elementos é feita utilizando uma extensão do algoritmo *Median Shift* (Jouili, Tabbone & Lacroix 2010), onde os elementos medianos são definidos como centróides dos *clusters*. Primeiramente, um raio de clusterização é determinado pelo usuário. Assim, é verificado em cada elemento, quais outros elementos estão contidos dentro desse raio. Esse processo é chamado de cálculo de vizinhança.

Pelo fato dos elementos não estarem contidos em um espaço vetorial, é necessário calcular todas as similaridades entre os elementos e transformar estes valores em distâncias. Como os valores de similaridade estão no intervalo de ‘0’ a ‘1’, sendo que o valor ‘0’ significa ser diferente e o valor ‘1’ ser igual, a distância é calculada utilizando a Equação 2, sendo que para valores de similaridade igual a ‘1’ é aplicada diretamente a distância zero.

$$Distância = \frac{1}{Similaridade} \quad (2)$$

Com isso, são calculadas as distâncias entre os elementos, estes são agrupados e os centróides definidos.

3. Persistência dos Clusters

Pelo fato do *WF-Sim* ainda estar em desenvolvimento, suas clusterizações não são persistidas, ou seja, a cada execução, todos os formulários são novamente processados e clusterizados. Isso foi feito devido à necessidade inicial de analisar os resultados obtidos, tendo mais relevância a qualidade dos resultados do que a velocidade para obtê-los.

O cálculo dos *clusters* é algo que demanda muito processamento e por isso é demorado. Portanto, uma das contribuições deste trabalho é a extensão do *WF-Sim* para permitir o armazenamento dos *clusters* calculados e, assim, executar o agrupamento somente quando novas *Web forms* forem consideradas.

Duas estratégias de persistência dos *clusters* foram implementadas: a persistência em diretórios de arquivos e a persistência em um banco de dados. As próximas seções detalham estas estratégias.

3.1 Persistência em arquivos

Uma das estratégias para guardar os dados dos *clusters* foi através de arquivos gerenciados diretamente pelo *WF-Sim*. Como existem alguns parâmetros que podem variar na criação dos *clusters*, cada nova clusterização foi armazenada em pastas de acordo com os parâmetros de entrada definidos. Com isso é possível armazenar várias clusterizações com diferentes configurações.

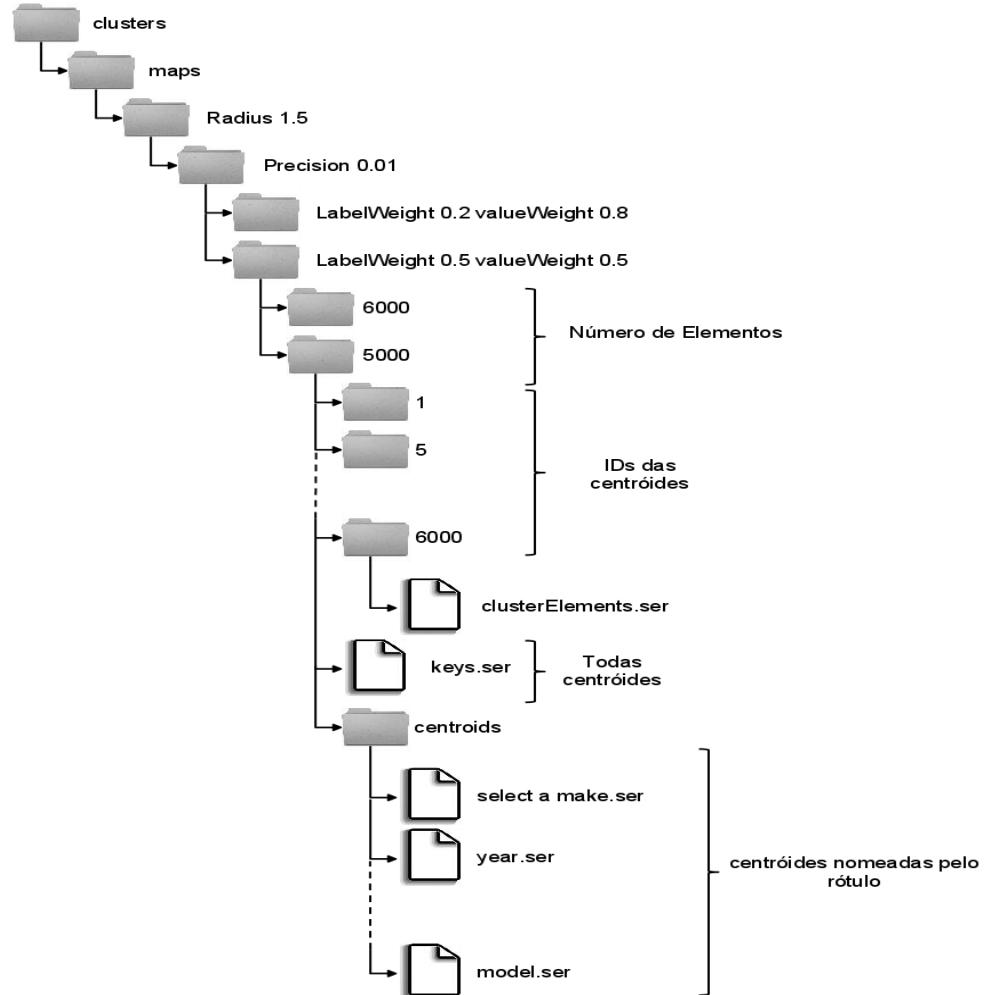


Figura 3. Organização dos *clusters* em pastas e arquivos

Como é possível ver na Figura 3, existem configurações de peso para rótulos e valores, precisão e raio de clusterização. Além disso, as clusterizações estão divididas pelo número de elementos utilizados na clusterização. Esta divisão considera outro parâmetro de configuração do *WF-Sim*, que é o número de elementos a ser carregado da base de dados de *Web forms*.

Cada clusterização gera vários *clusters* e, como visto anteriormente, cada *cluster* possui um centróide. Assim, os elementos de cada *cluster* foram armazenados em pastas nomeadas pelo identificador do centróide e os centróides foram armazenados separadamente em duas formas. A primeira forma foi um arquivo contendo todos os centróides identificados com um ID gerado pelo sistema. Desta forma, mantém-se em memória somente as informações dos centróides e se

carrega os *clusters* desejados (mantidos em pastas específicas identificadas pelos IDs dos centróides), conforme a busca do usuário. A segunda forma foi armazenar cada centróide em um arquivo separado nomeado pelo próprio rótulo. Com isso, é possível fazer buscas acessando diretamente as pastas dos *clusters* desejados.

Para possibilitar a criação de arquivos nomeados pelos rótulos dos centróides, foi necessário um pré-processamento que retira caracteres especiais que não são aceitos pelo sistema de arquivos. Ainda, alguns rótulos possuíam muitos caracteres e por isso foram filtrados rótulos com mais de 250 caracteres.

3.2 Persistência em BD

Além da persistência em arquivos, também se decidiu testar o armazenamento dos *clusters* em um banco de dados relacional. Para isso, foi necessário o projeto do banco de dados. A modelagem conceitual definida pode ser vista na Figura 4.

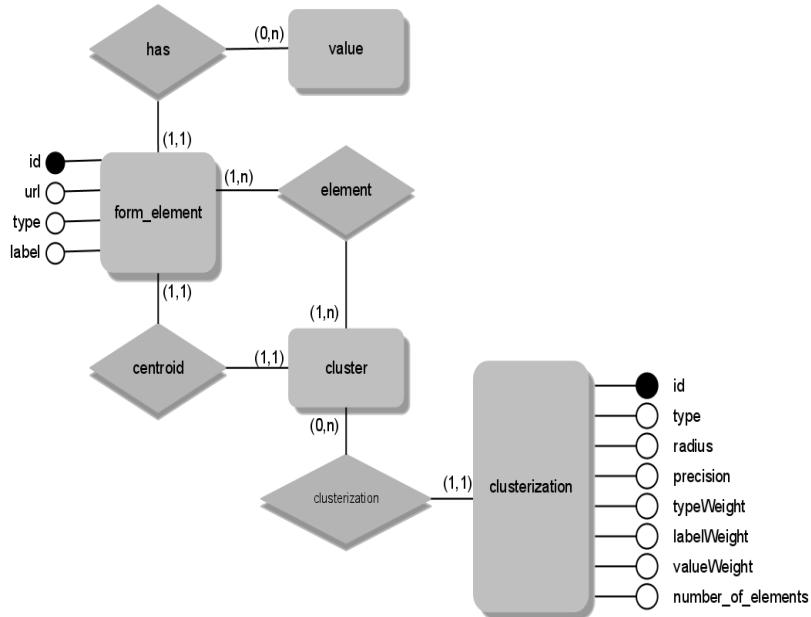


Figura 4. Modelagem conceitual da persistência dos *clusters*

Assim como no armazenamento em arquivos, encontrou-se o problema de como separar as clusterizações de acordo com cada configuração. Para resolver isso, uma entidade “*clusterization*” foi criada para manter estas configurações. Assim, cada *cluster* está relacionado a uma clusterização, podendo haver *clusters* iguais com configurações diferentes.

Como os centróides também são elementos, optou-se por não criar uma entidade centróides. Ao invés disso, a entidade ‘*cluster*’ possui uma relação ‘*centroid*’ com ‘*form_element*’ e outra relação ‘*element*’ também com ‘*form_element*’.

4. Experimentos

Para confirmar a melhoria de desempenho das estratégias propostas e compará-las, foram realizados alguns experimentos preliminares de busca sobre as clusterizações persistidas. Esses testes foram realizados em um computador com Intel Core2 Duo CPU T5800, 2 GHz, 3GB

RAM, Windows XP SP3, sendo 6157 o número de elementos clusterizados, divididos em 910 *clusters*.

Para a realização dos testes foi criado um formulário base (*template*) no domínio de automóveis com 10 elementos. Este é um formulário típico para busca de automóveis para venda ou aluguel. Este domínio foi escolhido pois é o que existe em maior quantidade na base de dados de *Web forms*. A Tabela 1 mostra os elementos considerados no *template* com suas restrições de valor, quando existentes.

Tabela 1. Web Form template considerado nos testes

ID	Rótulo	Valor	ID	Rótulo	Valor
1	select make	ford hyundai nissan Toyota	6	model	any model blazer caravan century civic cobalt enclave
2	min price	20,000 30,000 40,000 50,000 60,000 70,000 80,000	7	miliage	20,000 30,000 40,000 50,000 60,000 70,000 80,000
3	choose year	2000 2001 2002 2003 2004 2005 2006	8	used car	acura alfa romeo audi bmw fiat chevrolet cadillac
4	max price	200,000 300,000 400,000 500,000 600,000 700,000 800,000	9	sort order	by price by make by date by year
5	include	all vehicles new vehicles used vehicles certified vehicles	10	modified	

Para variar a quantidade de *clusters* retornados na busca, em alguns testes foram retirados elementos do *template* sendo que os mantidos foram aqueles que retornavam mais dados. Assim, testes foram feitos com 1, 5 e 10 elementos, para fins de avaliação de desempenho de um número crescente de elementos a ser considerado nas buscas.

Para encontrar os centróides desejados, tanto no banco de dados quanto nos arquivos, dividiu-se o *template* em seus elementos e, para cada elemento, foram encontrados os centróides cujos rótulos possuísem os mesmos termos do elemento de entrada. Encontrados esses centróides, foi aplicado o cálculo de similaridade do *WF-Sim* sobre eles. Os centróides com similaridade abaixo de 0,7 foram desconsiderados para manter no resultado apenas dados com alta similaridade.

Além da busca retornando os *clusters* mais similares, também foram realizados testes utilizando o mecanismo de ranqueamento (*ranking*) do *WF-Sim*. Esse ranqueamento deve ser executado durante o processamento de cada consulta para verificar quais *clusters* são mais semelhantes à consulta. O *WF-Sim* originalmente utilizava todos os *clusters* calculados na clusterização para o ranqueamento, produzindo um *overhead* na geração do resultado da consulta. Para reduzir este *overhead*, o mecanismo de ranqueamento foi modificado neste trabalho para considerar somente os *clusters* recuperados das buscas em arquivos e no banco de dados.

Nos experimentos realizados, o foco foi a avaliação de desempenho em termos de tempo das duas estratégias. A avaliação da relevância dos *clusters* retornados não foi tratada nesse trabalho, visto que os resultados das clusterizações já são validados pelo *WF-sim*.

A Tabela 2 apresenta os resultados dos experimentos. Os tempos mostrados são a média de 3 execuções para cada busca. As consultas foram *disjuntivas*, ou seja, são retornados os *clusters* que casam com pelo menos um (1) dos elementos do *template*. Por isso, quanto mais elementos são considerados como entrada para a busca, maior é a quantidade de *clusters* recuperados.

Tabela 2. Resultados dos Experimentos

Tipo de teste	#Elementos	#Clusters retornados	Tempo (seg.)
Arquivos s/ ranking	1	9	0,968
Arquivos c/ ranking	1	9	1,135
BD s/ ranking	1	9	1,588
BD c/ ranking	1	9	1,098
Arquivos s/ ranking	5	38	2,162
Arquivos c/ ranking	5	38	2,338
BD s/ ranking	5	38	4,052
BD c/ ranking	5	38	5,031
Arquivos s/ ranking	10	65	2,807
Arquivos c/ ranking	10	65	3,135
BD s/ ranking	10	65	5,531
BD c/ ranking	10	65	5,766

Os testes com 1 elemento consideraram buscas envolvendo o elemento com ID 1 do *template* na Tabela 1. Os testes com 5 elementos consideraram os elementos com ID de 1 a 5 e os testes com 10 elementos consideraram todos os elementos do *template*.

De acordo com a Tabela 2, a estratégia de persistência em arquivos sem considerar ranqueamento do resultado foi a mais rápida, tornando-se 2,2 vezes mais lenta com 5 vezes mais elementos considerados na busca e 2,8 vezes mais lenta com 10 vezes mais elementos considerados. Isso mostra uma tendência de redução do crescimento do *overhead* à medida que o número de elementos de entrada aumenta. A estratégia de arquivos com ranqueamento ficou, em média, 1,1 vez mais lenta, representando um *overhead* pouco significativo.

A estratégia de persistência em banco de dados sem ranqueamento tornou-se 2,5 vezes mais lenta com 5 vezes mais elementos considerados na busca e 3,4 vezes mais lenta com 10 vezes mais elementos considerados. Há uma tendência semelhante à da estratégia de persistência de arquivos, apesar de haver um *overhead* maior. A estratégia de banco de dados com ranqueamento ficou, em média, para 5 e 10 elementos, 1,1 vez mais lenta, representando também um *overhead* pouco significativo. Ela só obteve desempenho superior à estratégia sem ranqueamento para a busca com 1 elemento de entrada, sendo isso provavelmente uma pequena anomalia em termos de acesso ao banco de dados.

Comparando as estratégias arquivo e banco de dados, ambas com ranqueamento, observa-se um desempenho praticamente equivalente para buscas com 1 elemento. Já para buscas com 5 e 10 elementos, a estratégia de banco de dados mostra-se, em média, 2 vezes mais lenta.

5. Conclusão

Este trabalho descreve duas estratégias para armazenamento de *clusters* de dados de *Web forms*, bem como uma avaliação de desempenho de acesso para ambas no contexto do projeto *WF-Sim*, que realiza buscas por similaridade em *Web forms*. Comparado com trabalhos relacionados (Chang and Cheng 2007; Silva 2007; Hao et. al. 2010; Hong et. al. 2010; Nguyen et. al. 2010), verifica-se que os mesmos não apresentam detalhes sobre a persistência da clusterização ou utilizam apenas bancos de dados para armazenar os *clusters* gerados. Além disso, nenhum desses trabalhos utiliza o algoritmo de clusterização proposto pelo *WF-Sim* para o contexto de *Web forms*. Este algoritmo foi escolhido por ser dirigido a dados representados na forma de grafo (estrutura de representação dos atributos das *Web forms* no *WF-Sim*) e por gerar automaticamente centróides de cada *cluster*. Essa facilidade se mostrou útil para fins de indexação, pois apenas os centróides são indexados e servem como base para buscas, ao invés de se indexar todos os elementos do *cluster*.

Analisando os resultados dos experimentos, verificou-se um bom desempenho das duas estratégias, considerando a quantidade de *clusters* a ser acessada e filtrada. Entretanto, para uma maior quantidade *clusters* retornados, a persistência em arquivos obteve melhor desempenho. Mesmo assim, é necessário avaliar melhor a escalabilidade das estratégias com um volume maior de dados, dado que existem milhões de *Web forms* disponíveis atualmente.

Quanto à utilização ou não de ranqueamento, não se notou grande diferença de desempenho, ou seja, os testes com e sem ranqueamento obtiveram tempos próximos. Conclui-se, assim, que organizar o resultado da busca não prejudica tanto o desempenho. Esse resultado

era de certa forma esperado, pois o ranqueamento trabalha com um universo de *clusters* reduzido. Como trabalho futuro, pretende-se realizar esses testes sobre uma base de dados maior e verificar se esse desempenho permanece aceitável.

Pelo fato de não existir persistência anteriormente, a velocidade de busca do *WF-Sim* melhorou consideravelmente. Com a adição dessa melhoria, o tempo de busca chegou a ter redução de horas (pois a clusterização era re-executada a cada sessão de utilização do sistema) para apenas alguns segundos, sendo tais resultados decisivos para tornar o *WF-Sim* um sistema de busca viável.

Vale lembrar que os testes realizados se referem apenas a atividades de busca. A clusterização no *WF-Sim* ainda é um processo demorado, porém, não é executado frequentemente. Mesmo assim, a persistência do resultado desse processamento melhorou a desempenho das buscas no *WF-Sim*. Conforme mencionado anteriormente, pretende-se melhorar o desempenho da clusterização e também avaliar o desempenho com uma implementação alternativa utilizando o *K-Means* (MacQueen 1967), um algoritmo popular para clusterização de dados.

Referências

- Chang, K. C.-C and Cheng, T. (2007) "Entity Search Engine: Towards Agile Best-Effort Information Integration over the Web". In: Proceedings of the 2nd Conference on Innovative Data Systems Research (CIDR). p.108-113.
- Cohen, W., Ravikumar, P. and Fienberg, S. (2003) "A Comparison of String Distance Metrics for Name-Matching Tasks". In: Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03). p.73-78.
- Dorneles, C. F. et. al. (2004) "Measuring Similarity between Collection of Values". In: Proceedings of 6th ACM CIKM International Workshop on Web Information and Data Management (WIDM 2004). p.56-63.
- Gonçalves, R. et. al. (2011). "A Similarity Search Approach for *Web forms*". In: Proceedings of the IADIS International Conference IADIS WWW/Internet.
- Hao, L.; Wan-Li,Z. and Fei, R., (2010) "Describing the Semantic Relation of the Deep Web Query Interfaces Using Ontology Extended LAV". Journal of Software, v. 5, n. 1. p.89-98.
- Hong, J., He, Z., and Bell, D. A. (2010). "An evidential approach to query interface matching on the Deep Web". Information Systems, v.35, n.2. p.140-148.
- Jouili, S., Tabbone, S., and Lacroix, V. (2010). "Median graph shift: A new clustering algorithm for graph domain". In: Proceedings of the 20th International Conference on Pattern Recognition. p. 950-953.
- MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations". In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. p. 281–297.
- Nguyen, T., Nguyen, H. and Freire, J. (2010). "PruSM: A Prudent Schema Matching Approach for Web Forms". In: Proceedings of 19th ACM Conference on Information and Knowledge Management (CIKM). p.1385-1388.
- Silva, A., Freire, J., and Barbosa, L. (2007). "Organizing Hidden-Web Databases by Clustering Visible Web Documents". In: Proceedings of the International Conference on Data Engineering (ICDE). p.326-335.

Detectando similaridade entre diferentes representações de entidades usando Classificadores Bayesianos¹

Daniel S. de Oliveira, Carina F. Dorneles

Dpto. de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.049-900 – Florianópolis – SC – Brazil

{oliveirads, dorneles}@inf.ufsc.br

Resumo. Este artigo apresenta os resultados iniciais de um experimento realizado sobre o Apache Mahout com o objetivo de detectar similaridade entre bases relacionais, utilizando Classificadores Bayesianos. Foram realizados testes com os algoritmos disponíveis – Bayes e CBayes – para classificar os registros de duas bases de dados com conteúdo semelhante.

1. Introdução

Na Web, as entidades do mundo real podem estar representadas de formas diferentes, tornando a integração de dados uma tarefa árdua. Muitas vezes se deseja encontrar uma entidade e o ideal seria encontrar as representações mais significativas, de modo a enriquecer a consulta. Dada uma representação de uma entidade, por exemplo, “Universidade Federal de Santa Catarina”, afirmar que outra representação, por exemplo, “UFSC”, diz respeito à mesma entidade pode ser um problema de classificação binária: ou a segunda representação diz respeito à primeira, ou não. Dentre as técnicas de classificação binária, destacam-se os Classificadores Bayesianos, por sua simplicidade e eficiência. [Rennie ET AL 2003] sugerem correções que tornam o Classificador Textual Naive Bayes praticamente tão eficiente quanto o Support Vector Machine (SVM), um classificador muito usado pela confiabilidade nos resultados oferecidos [H and Thor 2010, Bilenko and Mooney 2003, Christen 2008].

Neste trabalho, são analisados experimentos de classificação de representações de entidades usando os Classificadores Bayesianos do Apache Mahout. Os algoritmos são treinados com entidades de uma, ou mais classes. Em seguida, são inseridas entidades diversas para serem classificadas. Por fim, a ferramenta aponta o percentual de classificações corretas. Dessa forma, pode-se afirmar quais são as entidades mais similares entre si. A Seção 2 deste artigo fala sobre os Classificadores Bayesianos e a implementação destes pelo Apache Mahout. A Seção 3 mostra os experimentos realizados. Os trabalhos futuros são apresentados na Seção 4.

2. Apache Mahout – Classificadores Bayesianos

Os Classificadores Bayesianos são capazes de determinar a classe de uma entidade, uma vez fornecidos dados de treino com entidades rotuladas. Suas limitações compreendem basicamente a necessidade de quantidade semelhante de registros nas classes de treino e

¹ Este trabalho é parcialmente financiado pelo CNPQ (Bolsa PQ-Nível 2 - Nro. processo: 307992/2010-1) e pela Capes (Bolsa do Programa REUNI).

a representatividade destes. O Mahout possui dois Classificadores Bayesianos: Naive Bayes (Bayes) e Complementary Naive Bayes (CBayes). O primeiro já possui algumas correções que levam em conta a quantidade de palavras que aparecem em um documento, representando uma classe e o somatório de ocorrências da classe ao longo do corpo de documentos. O segundo corrige erros sistêmicos do primeiro, que causam classificações errôneas, como dados de treino desbalanceados e classes com maior peso do que outras, causado pela assunção de independência entre as características. Além de modelar melhor o texto, aplicando técnicas de processamento de texto, como a métrica de similaridade Tf-Idf. Ambos possuem um parâmetro para ajuste fino manual dos pesos, α_i , conhecido como Suavizador de Laplace [Mitchell 2010], que serve para evitar superajustes que invalidam o modelo.

O modelo é gerado com base nos dados de treino, onde são apresentadas classes contendo entidades rotuladas. Depois são inseridos os documentos contendo representações de entidades e o Classificador, baseado no modelo gerado, aloca as representações nas classes com maior verossimilhança. A ferramenta dispõe de um Testador que verifica se as classificações foram realizadas corretamente. Como os dados de teste também são rotulados, é possível checar se uma representação foi alocada na classe de mesmo rótulo. Observa-se que este rótulo não é utilizado durante uma classificação normal, pois esta ocorre apenas com o processamento do Classificador sobre o modelo. A Figura 1 mostra a arquitetura do Classificador Bayesiano do Mahout.

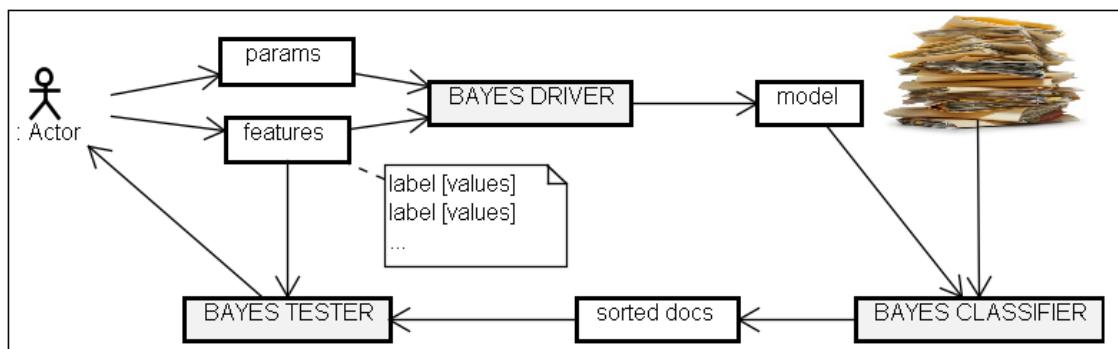


Figura 1. Arquitetura do Classificador Bayesiano do Apache Mahout

3. Experimentos

O objetivo dos experimentos é comprovar a eficácia dos Classificadores Bayesianos do Mahout quando aplicados sobre dados tabelados, a fim de indicar similaridade entre valores de atributos de representações de entidades simples – sem relacionamentos.

3.1. Configuração

Os experimentos foram realizados utilizando uma base de dados de ruas, contendo aproximadamente 3500 registros; e uma base de dados de inscrição em concurso, contendo aproximadamente 6500 registros: RUAS(ender), CONCURSO(chave, dtnasc, sexo, unidade, ender, comple, bairro, cidade, uf). A ferramenta foi treinada com metade (3250) dos registros da base de concursos, considerando três fatores: algoritmo (Bayes e CBayes), parâmetro de sensibilidade dos pesos – α_i – e conjunto de atributos (Completo e Seletivo). Este terceiro fator é empregado suprimindo atributos de códigos

julgados pouco significativos para representar a entidade real, neste caso “chave” e “unidade”, seguindo orientações do manual [Owen ET AL 2011]. O Testador foi utilizado após cada treino para checar a eficácia da classificação executada sobre a outra metade da base de concursos (3250 registros) e sobre a base de ruas (3500 registros).

3.2. Resultados

A Figura 2 mostra os gráficos dos resultados dos testes realizados com treinamento completo. No primeiro conjunto de testes, são classificadas 3250 representações da entidade Concurso, utilizando os dois algoritmos para comparar todos os atributos desta base com todos os da base de treino. No segundo conjunto de testes, são classificadas 3500 representações da entidade Rua, também utilizando os dois algoritmos para comparar todos os atributos das duas bases. A Figura 3 mostra os gráficos dos resultados dos testes realizados com treinamento seletivo, onde foram excluídos os atributos de códigos. O teste do terceiro conjunto é similar ao primeiro, assim como o quarto conjunto com relação ao segundo. O percentual de classificações corretas se refere aos atributos que foram alocados na classe esperada. No caso de Concurso sobre Concurso, cada atributo deve corresponder a si próprio. No caso de Rua sobre Concurso, o atributo “Rua.ender” deve corresponder ao atributo “Concurso.ender”.

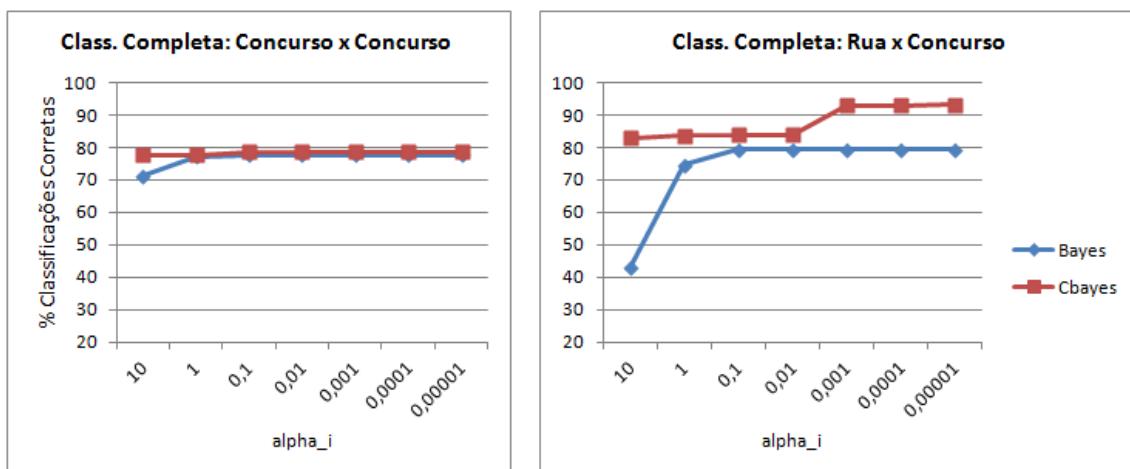


Figura 2. Classificação com todos os atributos

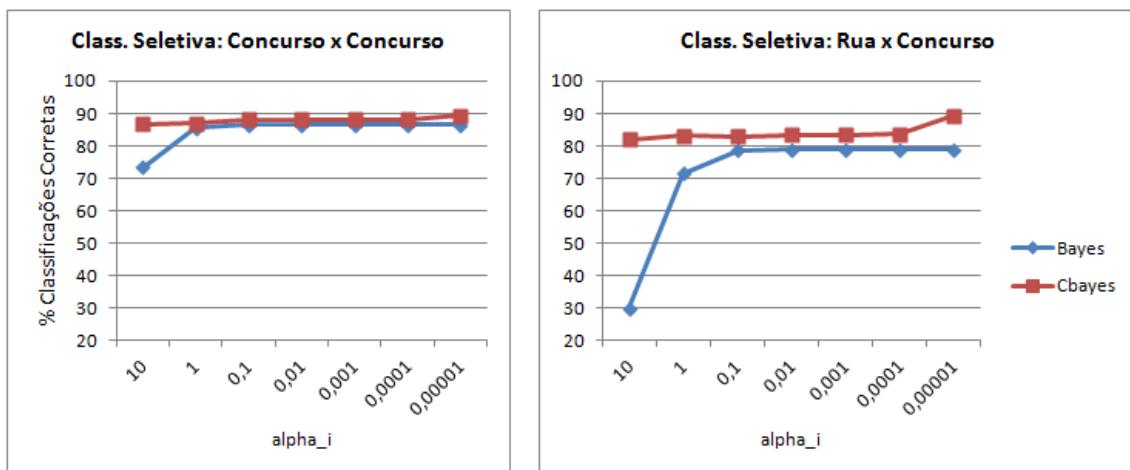


Figura 3. Classificação apenas com atributos representativos

Percebe-se que, em geral, o percentual de acerto é maior na classificação seletiva, pois os algoritmos não são confundidos pelos atributos de código. Percebe-se também que o algoritmo CBayes tem rendimento diferenciado nos dois casos em que as bases de treino e teste são diferentes, pois, nos outros dois casos, é mais fácil afirmar que a entidade testada é similar às entidades de treino, então o Bayes também apresenta resultado satisfatório. Ainda, percebe-se que o percentual de acerto aumenta à medida que o alpha_i diminui, pois ele não interfere tanto no processamento dos algoritmos. As classificações mal sucedidas se devem, basicamente, à semelhança entre valores de atributos distintos, como por exemplo, endereços completos que contêm o nome do bairro. Importante salientar que os resultados positivos na comparação entre as bases Rua e Concurso não indicam similaridade entre elas, mas sim entre os atributos comuns a ambas. Com base nessa capacidade de reconhecer atributos similares, pode-se proceder à indicação de entidades similares.

4. Conclusões e Trabalhos Futuros

Após estudo teórico e análise dos experimentos, conclui-se que os Classificadores Bayesianos do Apache Mahout são eficazes para detectar similaridade entre diferentes representações de entidades, tomando como base a similaridade entre seus atributos. Para aperfeiçoar os resultados gerados pelo Mahout, os algoritmos de Classificação Bayesiana serão adaptados, fazendo uso de métricas de similaridade já conhecidas no campo de recuperação de texto [Dorneles ET AL 2011], de modo a indicar a melhor função, ou conjunto de funções, para comparar cada atributo. Ainda, pretende-se automatizar a tarefa de testes, a fim de descobrir as melhores combinações de algoritmo e parâmetros de sensibilidade para uma determinada base de dados.

Referências

- Bilenko, M. and Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.
- Christen, P. (2008). Febrl: a freely available record linkage system with a graphical user interface. In *Proceedings of the second Australasian workshop on Health data and knowledge management-Volume 80*. Australian Computer Society, Inc.
- Dorneles, C. F., Gonçalvez, R., Mello, R. S. (2011). *Approximate data instance matching: a survey*. Knowledge and Information Systems, v. 27, i. 1.
- H and Thor, A. (2010). Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*, v. 3, n. 1, p. 484-493.
- Mitchell, T. M. (2010). Chapter 1 Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression Learning Classifiers based on Bayes Rule. *Machine Learning*. p. 1-17.
- Owen, S., Anil, R., Dunning, T. and Friedman, E. (2011). *Mahout in Action*. Manning Publications.
- Rennie, J. D., Shih, L., Teevan, J. and Karger, D. (2003). Tackling the poor assumptions of naive bayes text classifiers. In *Machine Learning-International Workshop Then Conference*.

Um sistema para análise de redes de pesquisa baseado na Plataforma Lattes

Lucas R. de Farias, André P. Vargas, Eduardo N. Borges

¹Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brazil

{lfarias, andre.prisco, eduardoborges}@furg.br

Abstract. This paper describes a system that extracts the scientific production of a set of researchers from their Lattes curriculums, analyzes the data searching for replicated references, and generates a graphical visualization of the collaboration networks among the researchers. You can view the list of publications in common to any two researchers and evaluate the quality of replica identification. The analysis of the collaboration networks allows better understanding how the interactions among researchers behave.

Resumo. Este artigo descreve um sistema que extrai a produção científica de um conjunto de pesquisadores, a partir dos seus currículos Lattes, analisa os dados extraídos em busca de referências bibliográficas replicadas, e gera uma visualização gráfica das redes de colaboração entre os pesquisadores. É possível visualizar a lista de publicações em comum a dois pesquisadores quaisquer e avaliar a qualidade da identificação de réplicas. A análise das redes de colaboração permite entender melhor como se comportam as interações entre pesquisadores.

1. Introdução

Atualmente, analisar a produção científica de uma instituição de ensino como um todo é uma tarefa bastante difícil. Além do elevado número de publicações dos servidores da universidade, as conferências e revistas científicas que contêm estas publicações, nacionais ou internacionais, podem não estar indexadas por ferramentas de busca como SciELO¹, Scopus², Google Scholar Citations³. Também podem existir múltiplas fontes de dados contendo citações para as mesmas publicações mas que diferem na forma com que foram escritas e representadas, o que poderia ser identificado como publicações distintas.

A Plataforma Lattes do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) é uma base de dados que contém, entre outras informações, os currículos da maior parte dos pesquisadores que atuam no Brasil. Grande parte dos editais de financiamento de projetos feitos por instituições de amparo à pesquisa, como o próprio CNPq, utilizam os currículos Lattes dos pesquisadores como uma das formas de avaliação das propostas. Este fato motiva os pesquisadores a manter seus currículos com informações corretas e atualizadas, tornando a Plataforma Lattes uma fonte adequada para análise da produção científica brasileira.

¹<http://www.scielo.org>

²<http://www.scopus.com>

³<http://scholar.google.com/citations>

O projeto Rede de Pesquisa da FURG tem como objetivo sintetizar informações sobre a produção científica da universidade a partir dos dados individuais de cada pesquisador. Este artigo apresenta a principal atividade do projeto que é a especificação e o desenvolvimento de um componente de *software* capaz de extrair a produção científica dos currículos Lattes, analisar os dados extraídos e visualizar graficamente as redes de colaboração entre os pesquisadores.

O restante do texto está organizado da seguinte forma. A seção 2 apresenta alguns trabalhos relacionados na área de redes sociais. A seção 3 apresenta a arquitetura do sistema proposto. Na seção 4 são descritos detalhes de implementação dos componentes de coleta, extração, análise e processamento da rede social. Também é especificado o esquema do banco de dados utilizado para representar a rede e os detalhes das publicações analisadas. A seção 5 mostra a interface do protótipo implementado e descreve suas principais funcionalidades. Por fim, na seção 6 são apresentadas as conclusões e alguns trabalhos futuros.

2. Trabalhos Relacionados

O sistema ArnetMiner [Tang et al. 2008] fornece serviços de busca *online* e mineração de dados para redes sociais de pesquisadores. Ele constrói perfis para os pesquisadores integrando informação acadêmica e bibliográfica de múltiplas fontes de dados. Os relacionamentos entre os pesquisadores são baseados na coautoria da produção bibliográfica. Entre os principais recursos destacam-se a visualização de subgrafos, a busca por especialistas em um determinado tema de pesquisa e os *rankings* acadêmicos baseado em diversas métricas.

O portal CiênciaBrasil [Laender et al. 2011] permite analisar e visualizar informação sobre pesquisadores brasileiros que participam dos Institutos Nacionais de Ciência e Tecnologia (INCT). Entre os principais recursos oferecidos pelo portal estão as visualizações das redes de colaboração entre os pesquisadores de cada instituto. Elas destacam as interações criadas e intensificadas após a formação de um INCT. As redes sociais são construídas de forma automática a partir de dados coletados da Plataforma Lattes e das páginas *Web* dos INCT. CiênciaBrasil identifica os relacionamentos de coautoria através de um algoritmo específico para deduplicação de citações bibliográficas [Borges et al. 2011b].

O scriptLattes [Mena-Chalco and Cesar Junior 2009] é uma ferramenta de código-fonte aberto que extrai dados de um conjunto de currículos Lattes e gera relatórios a partir deles. Estes relatórios contêm uma lista de todas as publicações do conjunto, com tratamento das publicações replicadas, gráficos da produção científica e um grafo das redes de colaboração entre os pesquisadores.

Os grafos gerados pelo portal CiênciaBrasil e pela ferramenta scriptLattes possuem alguns problemas. Primeiramente, as imagens geradas são estáticas, não sendo possível reposicionar vértices e arestas do grafo. Em redes complexas, com um número elevado de vértices e arestas, isto pode ser um grande empecilho para a visualização e entendimento das colaborações entre os pesquisadores. Além disso, nenhum dos sistemas estudados permite visualizar quais são as produções bibliográficas que dois pesquisadores publicaram em conjunto, ou seja, que artigos/livros/trabalhos geraram a aresta entre esses dois pesquisadores. Também não é possível visualizar como cada referência bibliográfica

está representada em sua forma original em cada fonte de dados de onde o relacionamento foi extraído.

Um dos objetivos do sistema proposto neste artigo é solucionar os problemas descritos anteriormente permitindo ao usuário alterar a disposição dos vértices e arestas, listar as publicações em comum a dois pesquisadores, visualizar as múltiplas representações de cada publicação e, por consequência, avaliar a identificação de referências bibliográficas duplicadas.

3. Arquitetura do Sistema

O sistema desenvolvido é composto por três componentes de processamento principais e um banco de dados. A partir de um conjunto de currículos Lattes, é gerado uma visualização gráfica de um grafo da rede de colaboração (Figura 1).

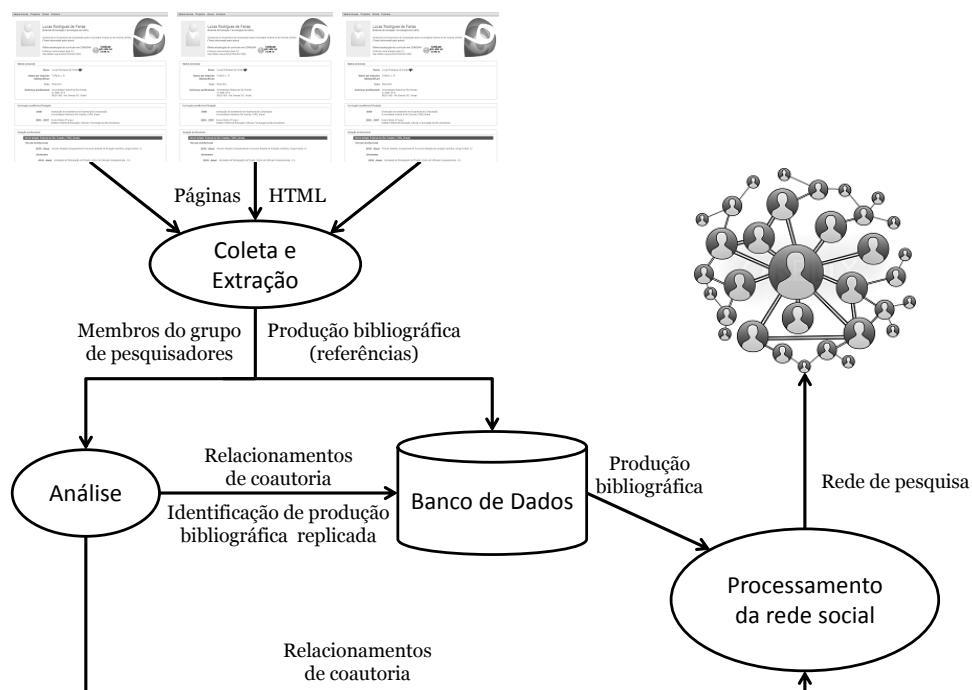


Figura 1. Arquitetura do sistema desenvolvido evidenciando entradas e saídas dos principais componentes.

O primeiro componente é denominado *Coleta e Extração*. Ele recebe como parâmetro um documento contendo uma lista de identificadores dos currículos Lattes dos pesquisadores que compõem o conjunto a ser analisado. O identificador de um currículo é obtido a partir dos últimos 16 dígitos de sua URL. O componente então faz o *download* das páginas HTML dos currículos Lattes e faz um processo de análise sintática, mais conhecido como *parsing*, que extrai dessas páginas os dados dos pesquisadores, como suas informações pessoais e publicações.

Os dados dos membros do conjunto de pesquisadores e as referências das produções bibliográficas são enviados para o componente *Análise*. Este componente analisa os dados extraídos, fazendo a identificação de produções bibliográficas duplicadas. A partir desta identificação, o componente gera os relacionamentos de coautoria entre os pesqui-

sadores do grupo. Estes resultados são armazenados em um banco de dados, assim como os dados recebidos do componente *Coleta e Extração*.

Os relacionamentos de coautoria previamente identificados também são enviados para o componente *Processamento da Rede Social*, que é responsável por gerar uma visualização gráfica da rede de pesquisa envolvendo os pesquisadores analisados. Este componente recupera do banco de dados as informações sobre as publicações dos pesquisadores, permitindo que usuários do sistema visualizem quais são as publicações que dois pesquisadores possuem em comum, selecionando os dois pesquisadores desejados na rede de pesquisa. Além disso, é possível ver como uma produção científica comum a dois pesquisadores está referenciada no currículo Lattes de cada um.

A Figura 2 apresenta um exemplo de rede de colaboração gerada pelo sistema para um determinado subgrupo de pesquisadores do Centro de Ciências Computacionais (C3), uma das 13 unidades acadêmicas Universidade Federal do Rio Grande (FURG). Muitos dos seus professores foram contratados recentemente, durante o programa REUNI, e ainda estão fortemente ligados aos grupos de pesquisa em que atuaram durante sua pós-graduação. Entretanto, novas interações entre os docentes do C3 estão surgindo. Analisar a produção científica em comum dos docentes permite identificar se estas interações poderiam formar grupos formais de pesquisa. Essa é uma necessidade atual do C3 e de outras novas unidades acadêmicas da universidade.

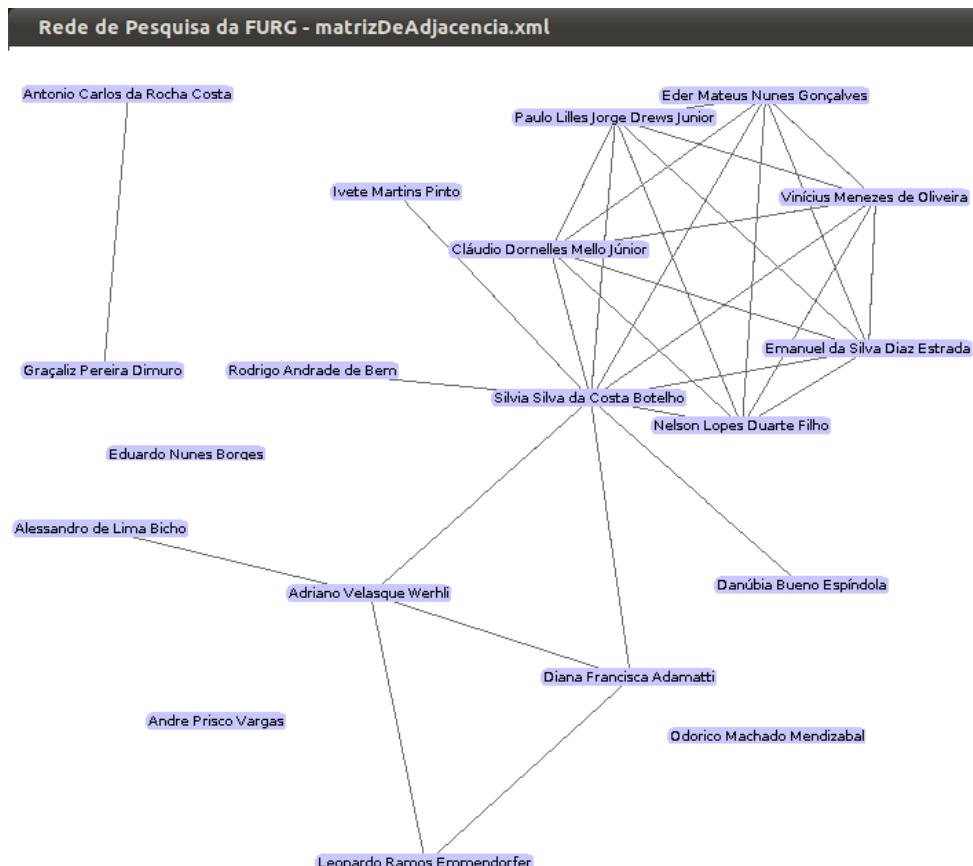


Figura 2. Exemplo de rede de pesquisa composta por parte dos docentes do C3.

Na Figura 2, nota-se a organização de três grupos principais. O primeiro grupo, localizado no canto superior direito, é formado pelos pesquisadores Cláudio, Éder, Emanuel, Nelson, Paulo, Sílvia e Vinícius. A característica de ser um grafo completo indica que este grupo possui forte cooperação interna. Este comportamento se deve ao fato deste grupo de pesquisa existir formalmente e de forma consolidada.

O segundo grupo, formado pelos pesquisadores Adriano, Alessandro, Danúbia, Diana, Ivete, Leonardo, Rodrigo e Sílvia, caracteriza-se por um grafo com um número consideravelmente menor de colaborações. Observe que os pesquisadores Danúbia, Ivete e Rodrigo estão ligados apenas à Sílvia Botelho. Este comportamento se deve ao fato da relação entre os pesquisadores ser de orientação de graduação ou pós-graduação. Também é possível perceber que a pesquisadora Sílvia coopera com ambos os grupos analisados.

Além destes, há um terceiro grupo de pesquisa formado por apenas dois pesquisadores. Antônio Rocha Costa e Graçaliz Dimuro atuam em parceria há muito tempo antes de ingressarem no C3 ano passado. Existem ainda outros pesquisadores isolados, ou seja, que não colaboraram com nenhum outro pesquisador do conjunto no período de tempo analisado. Colaborações com grupos de pesquisa externos ao C3 não foram processadas.

4. Implementação

Esta seção apresenta os detalhes de implementação dos componentes de coleta, extração, análise e processamento da rede social. O banco de dados utilizado para representar os relacionamentos de coautoria e os metadados das referências bibliográficas é especificado em um esquema relacional.

4.1. Coleta e Extração

O scriptLattes é um *script* GNU-GPL desenvolvido para a extração e compilação automática de produções acadêmicas de um conjunto de pesquisadores cadastrados na Plataforma Lattes. A partir dos dados coletados, o *software* gera uma rede de colaboração entre os pesquisadores e relatórios que descrevem o conjunto analisado [Mena-Chalco and Cesar Junior 2009].

O *software* faz o *download* dos currículos Lattes de um grupo de pessoas de interesse e compila listas de produções, tratando as produções duplicadas e similares. São geradas páginas HTML com listas de produções e orientações separadas por tipo e colocadas em ordem cronológica invertida. Adicionalmente, são criados automaticamente vários grafos (redes) de coautoria entre os membros do grupo de interesse e um mapa de geolocalização dos membros e alunos de pós-graduação com orientação concluída. Os relatórios gerados permitem avaliar, analisar ou documentar a produção de grupos de pesquisa.

Para o desenvolvimento do trabalho apresentado neste artigo, foram utilizados apenas os módulos de *seleção* e *pré-processamento* de dados do scriptLattes. O código-fonte foi adaptado para comunicar-se com o SGBD PostgreSQL⁴. Com o banco de dados criado, foi possível armazenar as informações de interesse sobre os pesquisadores e suas publicações, as relações entre autores e as informações sobre referências duplicadas (indisponíveis no *software* original). Foram consideradas relevantes apenas as publicações

⁴<http://www.postgresql.org>

dos tipos artigos completos publicados em periódicos, artigos aceitos para publicação, livros publicados ou organizados, capítulos de livros publicados e trabalhos completos publicados em anais de congressos.

4.2. Análise

A identificação de duplicatas utiliza a solução já implementada pelo scriptLattes. O sistema utiliza como base do processo de deduplicação a métrica distância de edição [Levenshtein 1966]. O algoritmo determina a similaridade entre duas cadeias de caracteres a partir do número mínimo de edições necessárias para transformar uma cadeia na outra. Quanto menor este número, maior a similaridade.

Duas publicações são identificadas como réplicas quando a similaridade entre os títulos for maior que um limiar de similaridade preestabelecido. Além disso, somente publicações do mesmo ano são comparadas. Todas as publicações do conjunto de pesquisadores são colocadas em uma lista ordenada. Cada publicação é comparada com as publicações subsequentes da lista, até encontrar uma que seja identificada como sua duplicata.

O algoritmo do scriptLattes assume uma propriedade transitiva da função de similaridade adotada. Por exemplo, se a referência bibliográfica A é similar à B e B é similar à C , então é assumido que A é similar à C . Todas as referências similares entre si, ou seja, detectadas como réplicas de uma mesma publicação, são organizadas em um mesmo agrupamento (*cluster*).

A seguir, são gerados os relacionamentos de coautoria. Para cada pesquisador p é extraída a lista de seus colaboradores a partir dos outros autores do seu conjunto de publicações únicas, ou seja, do conjunto de referências bibliográficas em todos os seus agrupamentos. O número de publicações compartilhadas entre p e um pesquisador q é obtido a partir da contagem do número de vezes que q aparece na lista. O mesmo processo é feito com todos os outros pesquisadores do grupo.

4.3. Banco de Dados

A Figura 3 apresenta o diagrama relacional do banco de dados implementado. O nome dos pesquisadores que pertencem ao conjunto analisado bem como os identificadores dos currículos Lattes são armazenados na tabela *membros*. A tabela *adjacencia* armazena os relacionamentos de coautoria entre cada par de membros da rede de pesquisa e o peso da colaboração. Este peso representa o número de publicações em comum entre dois pesquisadores. Na aplicação podem ser calculados outros pesos derivados como a razão entre o número de publicações em comum e o total de publicações de um pesquisador, tornando os relacionamentos bidirecionais (grafo dirigido).

A tabela *publicacao* armazena os principais metadados das citações bibliográficas: título, autores, ano de publicação e número de páginas. Note que estes campos estão presentes em qualquer tipo de publicação analisada. Cada publicação pertence a apenas um membro porque foi extraída de seu currículo. Para cada tipo de publicação foi criada uma tabela auxiliar que armazena os metadados específicos que não se aplicam aos demais tipos. Por exemplo, capítulos de livros publicados (tabela *capitulo*) possuem número de edição, enquanto artigos aceitos para publicação (tabela *revista_aceito*) não possuem.

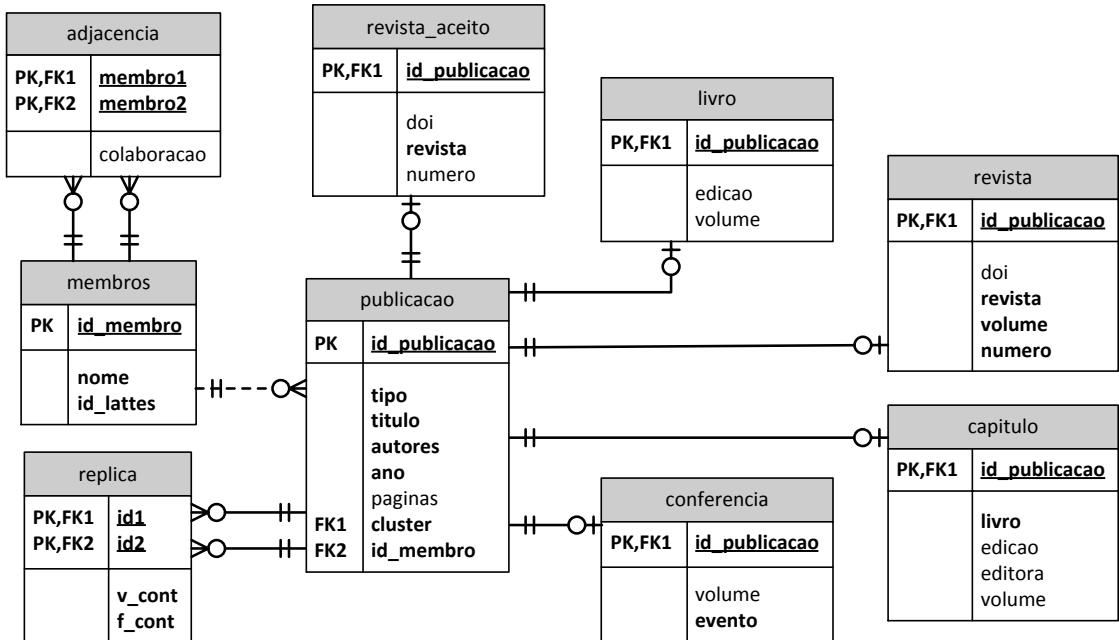


Figura 3. Projeto relacional do banco de dados desenvolvido.

Os relacionamentos entre a tabela *publicacao* e estas tabelas auxiliares podem ser vistos como mapeamentos de herança de tipos.

As instâncias da tabela *publicacao* representam todas as referências bibliográficas extraídas de todos os currículos, sem o processo de identificação de duplicatas. Pares de referências identificadas como réplicas da mesma publicação são armazenados na tabela *replica*. Além do par de identificadores das publicações duplicadas, esta tabela contém os campos *v_cont* e *f_cont*, responsáveis por controlar o *feedback* da deduplicação. Cada vez que um usuário da aplicação avalia a detecção como correta, *v_cont* é incrementado. Caso contrário, *f_cont* é incrementado. A diferença entre os valores destes dois campos é usada como uma boa estimativa da qualidade, independente do processo de deduplicação utilizado. Considerando a propriedade transitiva da função de similaridade (seção 4.2), cada instância da tabela *publicacao* ainda referencia um agrupamento de referências bibliográficas replicadas através do campo *cluster*. Assim, é possível verificar a forma como uma publicação está representada em cada um dos currículos de seus autores.

4.4. Processamento da Rede Social

O componente *Processamento da rede social* recebe do banco de dados a produção bibliográfica individual dos pesquisadores e as múltiplas representações das referências bibliográficas replicadas. Os relacionamentos de coautoria identificados na etapa de análise também são recebidos como entrada. A partir destes relacionamentos o componente monta uma visualização gráfica da rede de colaboração a partir de um grafo.

5. Protótipo

O processamento da rede social foi implementado em Java utilizando as bibliotecas Prefuse [Heer et al. 2005] e Swing [Hoy et al. 2002]. Prefuse foi utilizada para visualização gráfica do grafo de coautoria e Swing para o desenvolvimento da interface gráfica do usuário (GUI). Uma das vantagens da biblioteca Prefuse é o suporte ao formato de arquivo GraphML exportado pelo scriptLattes. GraphML é uma sublinguagem da XML exclusiva para a descrição de grafos [Brandes et al. 2002]. Outra vantagem é o suporte à construção de grafos interativos.

O sistema permite ao usuário mover os vértices, configurando uma nova disposição para os elementos do grafo. Este recurso permite entender melhor os relacionamentos entre os pesquisadores e os grupos de pesquisa dos quais fazem parte.

A Figura 4 apresenta uma captura de tela destacando a seleção de dois vértices que representam os pesquisadores Diana e Leonardo. Quando o segundo vértice é selecionado, é exibida ao usuário uma janela contendo a lista de publicações em comum aos pesquisadores selecionados. Cada publicação é apresentada apenas uma vez.

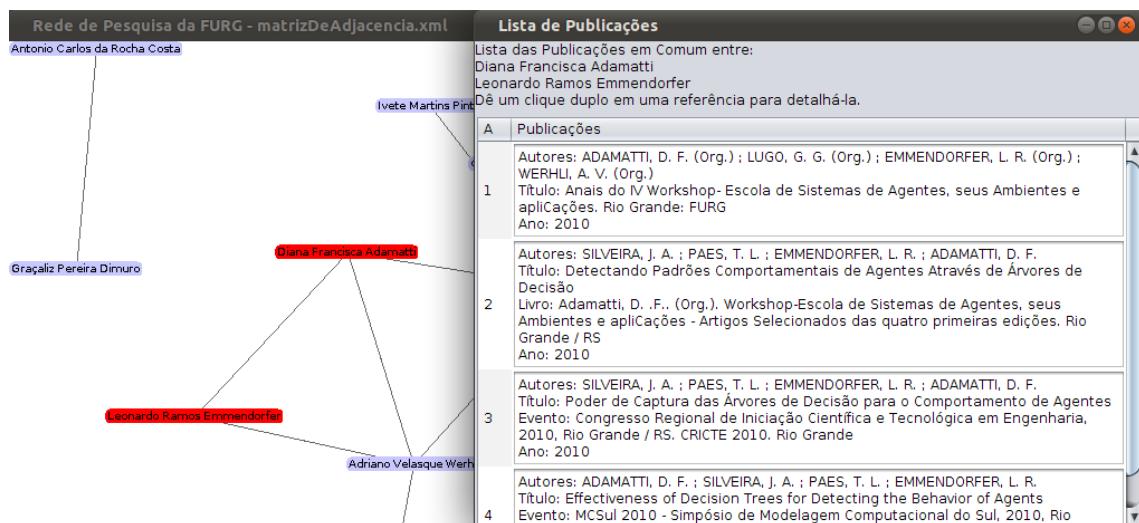


Figura 4. Lista de publicações em comum aos pesquisadores selecionados.

Um dos recursos mais interessantes é a visualização das múltiplas representações de uma publicação e a validação do processo de deduplicação. A Figura 5 apresenta uma nova janela que é aberta quando uma das publicações da lista anterior é selecionada. Nesta janela são apresentadas as descrições das referências bibliográficas na sua forma original contida nos currículos Lattes dos pesquisadores. Assim, é possível verificar se esta publicação realmente é de autoria de ambos os pesquisadores, ou seja, se a identificação de duplicatas realmente está funcionando. O usuário pode validar a deduplicação selecionando os botões *Sim* ou *Não*, incrementando os valores dos contadores da tabela *replica* (seção 4.3). Um equívoco na validação pode ser corrigido clicando no botão *Cancelar* para retificar a sua seleção.

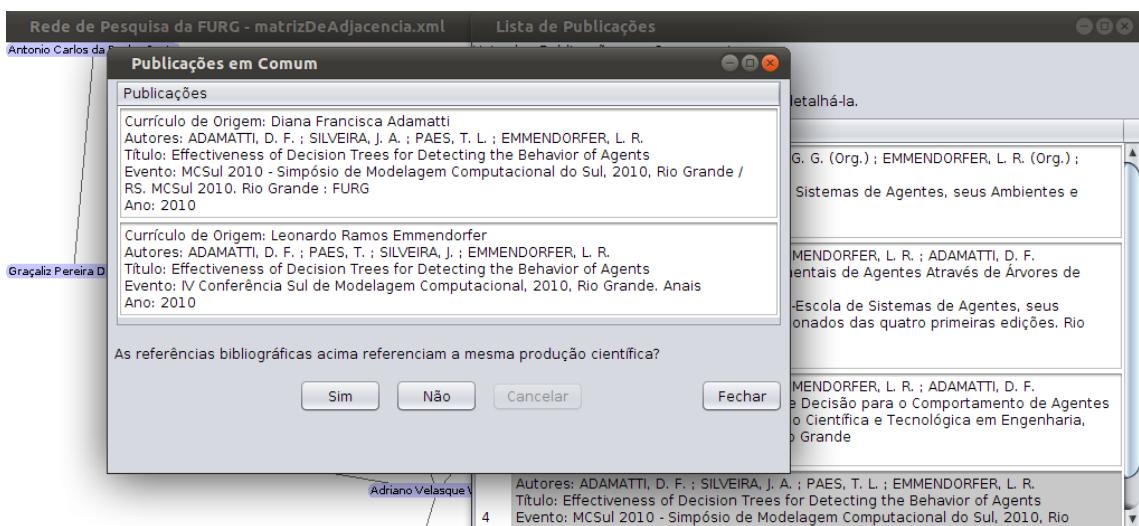


Figura 5. Avaliação da identificação de réplicas através da análise das múltiplas representações da mesma publicação.

6. Conclusão

Este trabalho apresenta o desenvolvimento de um sistema para visualizar graficamente as redes de colaboração entre um conjunto de pesquisadores. A representação gráfica de uma rede de colaboração de pesquisadores permite que uma grande quantidade de informação seja analisada rapidamente.

Ao invés do usuário ter que identificar as interações e os relacionamentos entre os membros do conjunto analisado de forma explícita, como nas redes sociais tradicionais, o sistema proposto extrai estes relacionamentos de forma automática. A base de dados utilizada para a extração foi a Plataforma Lattes. A produção científica dos currículos Lattes é analisada para inferir os relacionamentos de coautoria entre os pesquisadores.

Facilitar o entendimento das interações entre os pesquisadores como uma rede social é uma contribuição significativa. É possível identificar grupos de pesquisa emergentes bem como analisar a produção de grupos bem estabelecidos. O sistema proposto também pode auxiliar o processo de constituição de novos grupos de pesquisa e o gerenciamento de grupos existentes.

Outra característica importante é que o sistema permite ao usuário avaliar qualquer técnica de identificação de duplicatas usada para inferir relacionamentos de coautoria. Portanto, outros algoritmos de detecção de referências bibliográficas replicadas [Borges et al. 2011a] podem ser utilizados.

Como trabalhos futuros destacam-se a inclusão dos pesos nas arestas, o desenvolvimento de um filtro temporal, a configuração de detalhes na interface gráfica, o desenvolvimento de um *applet Web*, e o desenvolvimento de um sistema de recomendação com o objetivo de sugerir novas colaborações entre pesquisadores. Com os novos recursos implementados, a análise das redes de colaboração permitirá entender melhor como se comportam as interações entre pesquisadores da universidade ao longo do tempo.

Referências

- Borges, E. N., Becker, K., Heuser, C. A., and Galante, R. (2011a). A classification-based approach for bibliographic metadata deduplication. In *Proceedings of the IADIS International Conference WWW/Internet*, pages 221–228.
- Borges, E. N., de Carvalho, M. G., Galante, R., Gonçalves, M. A., and Laender, A. H. F. (2011b). An unsupervised heuristic-based approach for bibliographic metadata deduplication. *Information Processing and Management*, 47(5):706–718.
- Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., and Marshall, M. (2002). *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, chapter GraphML Progress Report Structural Layer Proposal, pages 109–112. Springer.
- Heer, J., Card, S. K., and Landay, J. A. (2005). prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 421–430, New York. ACM.
- Hoy, M., Wood, D., Loy, M., Elliot, J., and Eckstein, R. (2002). *Java Swing*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2 edition.
- Laender, A. H. F., Moro, M. M., Silva, A. S., Davis Jr., C. A., Gonçalves, M. A., Galante, R., Silva, A. J. C., Bigonha, C. A. S., Dalip, D. H., Barbosa, E. M., Borges, E. N., Cortez, E., Procópio Jr., P., de Alencar, R. O., Cardoso, T. N. C., and Salles, T. (2011). Ciênciabrasil - the brazilian portal of science and technology. In *Seminário Integrado de Software e Hardware, Anais do Congresso da Sociedade Brasileira de Computação*, pages 1366–1379.
- Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.
- Mena-Chalco, J. A. P. and Cesar Junior, R. M. (2009). ScriptLattes: an open-source knowledge extraction system from the Lattes platform. *Journal of the Brazilian Computer Society*, 15:31 – 39.
- Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. (2008). Arnetminer: extraction and mining of academic social networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 990–998, New York. ACM.

Pré-processamento de tabelas Web heterogêneas para execução do algoritmo de junção *merge*.

Larissa R. Lautert¹, Carina F. Dorneles¹

¹Curso de Ciência da Computação – Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brasil

{llautert, dorneles}@inf.ufsc.br

Abstract. *The Web is the largest repository of data available, with over 150 million high-quality tables. Several works have combined their efforts to allow queries on them, but there are still challenges. One challenge, in particular, involves the various different types of structures found on the Web, limiting the amount of tables in the appropriate format for the application of operators used in traditional databases. One of the fundamentally important operations for allowing data to be queried from different sources is the merge join. In this work, the merge join algorithm was adapted to two types of structures easily found on the Web. We proposed three algorithms adaptions, all aimed at performance. According to the tests performed, one of them offers considerably higher performance than the others.*

Resumo. *A Web é o maior repositório de dados disponível, contando com mais de 150 milhões de tabelas no estilo relacional. Muitos trabalhos têm unido esforços a fim de permitir consultas sobre elas, porém ainda existem desafios. Um deles diz respeito aos diversos tipos de estrutura encontrados na Web, limitando a quantidade de tabelas na formatação apropriada para aplicação dos operadores utilizados em bancos de dados tradicionais. Uma das operações de fundamental importância para permitir consulta dados de diferentes fontes é a junção merge. Neste trabalho, o algoritmo de junção merge foi adaptado para dois tipos de estrutura facilmente encontrados na Web. São propostas três adaptações distintas, todas visando desempenho. De acordo com os testes realizados, uma delas apresenta desempenho consideravelmente maior do que as outras.*

1. Introdução

A Web oferece mais de 150 milhões de tabelas HTML com dados dos mais diversos domínios no formato relacional [Cafarella et al. 2008]. Como os motores de busca existentes consideram apenas a proximidade entre palavras-chave, não tiram proveito destas estruturas para melhorar seus resultados. A fim de permitir a exploração de uma quantidade maior de dados, pode-se utilizar conceitos tradicionais de Bancos de Dados Relacionais, como a operação de junção. Dessa forma, é possível estabelecer ligações entre tabelas HTML de páginas diferentes, com dados complementares, para se obter resultados mais completos.

Efetuar junção entre tabelas Web tem sua principal dificuldade na heterogeneidade de formatações encontradas. Devem ser consideradas tabelas com rótulos na primeira linha, na primeira coluna e até mesmo em seu interior, muitas vezes se referindo a

mais de uma coluna. Além destes casos, existem tabelas aninhadas com conteúdo semi-estruturado em algumas células.

Com o objetivo de permitir a execução de consultas sobre dados de tabela HTML, vários trabalhos da literatura unem esforços para extraí-los apropriadamente. Nesse contexto, precisa-se considerar o problema de rótulos mal formulados nos cabeçalhos, ou seja, títulos inadequados para o conteúdo das colunas das tabelas. Essa questão já foi abordada por Venetis et al. [Venetis et al. 2011] e Silva et al. [da Silva et al. 2007]. De modo a auxiliar na junção, existe outro trabalho que propõe uma técnica para descobrir novas entidades a partir de *links* existentes em tabelas, sendo capaz de obter mais informações sobre determinado campo [Lin et al. 2010]. No contexto de execução de consultas estruturadas, o protótipo *Structured Web Search Engine* utiliza tabelas HTML como base de dados[Mergen et al. 2010]. Dos trabalhos analisados, nenhum aborda consultas com a operação de junção entre tabelas com formatação diferente da relacional.

Neste trabalho, são propostas três adaptações no algoritmo de junção *merge* tradicional, considerando tabelas com rótulos na primeira linha (caso tradicional) e rótulos na primeira coluna (tabelas transpostas). A Figura 1 ilustra as duas situações. Para detectar se a tabela encontra-se transposta, utilizou-se o trabalho de Silva et al. [da Silva et al. 2007], que identifica se determinado rótulo candidato descreve adequadamente o conteúdo de uma coluna. Após, é aplicado o algoritmo de junção *merge* adaptado para suportar este caso especial. Os testes de desempenho mostraram que o aumento no tempo de execução

Title	Language	Nominations	Awards	Director
12 Angry Men	English	6	13	Sidney Lumet
Pulp Fiction	English Spanish	45	46	Quentin Tarantino
The Divide	English	3	2	Xavier Gens
The Godfather	English Italian	24	17	Francis Ford Coppola

(a)

Title	12 Angry Men	Pulp Fiction	The Divide	The Godfather
Language	English	English Spanish	English	English Italian
Nominations	6	45	3	24
Awards	13	46	2	17
Director	Sidney Lumet	Quentin Tarantino	Xavier Gens	Francis Ford Coppola

(b)

Figura 1. Dois tipos de estruturas facilmente encontrados na Web: (a) tabela tradicional e (b) tabela transposta.

da junção quando se tem tabelas transpostas é pequeno, de forma que o método proposto é considerado viável, dada a escala da Web.

O artigo está estruturado da seguinte forma. Na Seção 2, listam-se trabalhos relacionados. Na Seção 3, apresenta-se a formalização do algoritmo de junção *merge* e descreve-se brevemente as abordagens utilizadas. A Seção 4, central deste artigo, explica as mudanças feitas no algoritmo tradicional para adequação ao novo caso. Na Seção 5, mostram-se os experimentos realizados para avaliar o desempenho dos protótipos desenvolvidos. Por fim, a Seção 6 lista trabalhos futuros e conclui o artigo.

2. Trabalhos relacionados

O problema de junção entre tabelas HTML com estruturas heterogêneas não foi diretamente abordado por nenhum outro trabalho. Entretanto, dada a importância e relevância do conteúdo encontrado nestas estruturas [Cafarella et al. 2008] vários trabalhos tratam a questão de extração de dados no formato relacional de páginas Web, seja de tabelas ou de listas HTML [Elmeleegy et al. 2009]. Soluções já foram propostas para geração de rótulos apropriados [da Silva et al. 2007] [Venetis et al. 2011], descoberta de novas entidades em tabelas através de *links* [Lin et al. 2010] e execução de consultas utilizando tabelas HTML como base de dados [Mergen et al. 2010].

Os rótulos presentes nos cabeçalhos de tabelas extraídas da Web são um problema, pois raramente refletem o conteúdo exato da coluna em questão [Venetis et al. 2011]. Com o intuito de resolver essa questão, Altigran et al. utilizaram um método para rotular colunas combinando um conjunto pré-definido de candidatos e resultados de motores de busca [da Silva et al. 2007]. O algoritmo utiliza um modelo probabilístico que leva em conta a afinidade entre os valores de determinada coluna da tabela e os rótulos em potencial, previamente definidos. A probabilidade é estimada pela contagem do número de respostas a consultas do tipo “*<rótulo><valor>*” feitas diretamente em motores de busca para todos os valores das colunas. Verificou-se que quanto mais ocorrências são retornadas para determinada consulta, melhor o rótulo utilizado se aplica ao conjunto de dados da coluna.

Outra abordagem usada na solução do problema de extrair corretamente os rótulos das colunas utiliza uma base de dados com o objetivo de inferir automaticamente a semântica de determinada tabela [Venetis et al. 2011]. Especificamente, associam-se anotações descrevendo os nomes de atributos presentes e relações binárias entre estes. A principal contribuição da abordagem é utilizar fatos extraídos de textos na Web para interpretar tabelas, pois assim garante-se que a técnica tem uma vasta cobertura de domínios. Para isso, consideram-se bancos de dados do tipo *isA*, cujas entradas são pares do tipo (*instância, classe*), e um banco de dados de relações com triplas do tipo (*argumento1, predicado, argumento2*). Uma coluna *A* é rotulada com a classe *C* se uma fração significativa de células *A* é rotulada com a classe *C* no banco de dados *isA*. De forma similar, um relacionamento entre duas colunas *A* e *B* é rotulado como *R* se um número significativo de pares de valores *A* e *B* ocorre na forma (*a, R, b*) no banco de dados de relações.

Além das tabelas, as estruturas de lista também são frequentes na Web e, algumas vezes, podem ser convertidas para o estilo relacional. Dessa forma, podem ser utilizadas como base de dados para consultas. Com essa proposta, foi desenvolvido o *ListExtract* [Elmeleegy et al. 2009] capaz de transformar listas em tabelas. O algoritmo proposto,

que é independente de domínio, busca pela melhor tabela possível na qual a lista seria segmentada. Primeiramente consideram-se tipos de dados, sintaxe, delimitadores, uma base de dados que armazena a pontuação para a ocorrência de determinados termos e um conjunto de tabelas extraídas automaticamente. Os dois últimos recursos auxiliam na identificação de termos que não devem ser separados. Após essa fase inicial, a tabela é analisada como um todo a fim de identificar possíveis erros na separação, considerando a coesão dos valores das colunas.

Outro trabalho analisado tem o objetivo de encontrar novos atributos para uma tabela através de *links* presentes nesta [Lin et al. 2010]. Dada uma coluna onde todos os valores possuem *links*, a solução proposta analisa todas as páginas destes *links* em busca de trechos com informações em comum. Por exemplo, no caso de uma tabela sobre professores de determinado departamento, poderia haver uma coluna com os *links* de suas *homepages*. Se na página de cada professor fosse encontrado seu endereço de *e-mail*, a coluna *e-mail* seria incorporada à tabela inicial.

Para a realização de consultas sobre tabelas HTML, a abordagem *Structured Web Search Engine* (SWSE) [Mergen et al. 2010] foi projetada tendo em vista a escala e a natureza dinâmica das fontes de dados Web. Para evitar criação e manutenção de mapeamentos entre as fontes de dados, estes são derivados automaticamente através do casamento entre os atributos da consulta e das tabelas. Essa abordagem, entretanto, não garante que todas as respostas corretas serão retornadas nem que as respostas retornadas estão, de fato, corretas. As consultas devem ser expressas em linguagem SQL, com suporte às operações de *seleção*, *projeção* e *junção*. Como resposta, o usuário obtém uma lista de possíveis reescritas, ou seja, diferentes formas de responder à consulta. A limitação desta abordagem consiste nas tabelas utilizadas como base de dados. São consideradas apenas as formatadas como uma tabela relacional, sem levar em conta outros tipos de estruturas heterogêneas facilmente encontradas na Web.

Neste trabalho o algoritmo de junção *merge* foi adaptado para a execução sobre tabelas HTML. Como mencionado, foram consideradas tabelas tradicionais, com rótulos na primeira linha, e tabelas transpostas, com rótulos na primeira coluna. Para identificar a formatação, foi utilizado o método proposto por Altigran et al., que infere a afinidade de valores com um rótulo candidato através de um motor de busca [da Silva et al. 2007]. Para a realização da junção, três abordagens diferentes foram implementadas, em busca do melhor desempenho.

3. Algoritmo de junção *merge*

O algoritmo original de junção *merge* pode ser usado para a implementação de junção natural e equi-junção. A seguir, ele é descrito para as relações $r(R)$ e $s(S)$, onde $R \cap S$ representa seus atributos em comum. Por questões de desempenho, o algoritmo supõe que ambas as relações encontram-se ordenadas de acordo com os atributos de $R \cap S$. No algoritmo de junção *merge* mostrado na Figura 2, *AtribJunção* (linha 12) refere-se aos atributos em $R \cap S$, e $t_r \bowtie t_s$, em que t_r e t_s são tuplas que possuem os mesmos valores para *AtribJunção*, denota a concatenação dos atributos das tuplas. O algoritmo começa associando um ponteiro à primeira tupla de cada relação. Após, armazena as tuplas de s com o mesmo valor para *AtribJunção* e lê as correspondentes na relação r . É importante que as tuplas de s caibam na memória principal para evitar transferência de

blocos, pois tornaria o processo mais oneroso. Como as duas relações estão ordenadas e o algoritmo as percorre sequencialmente, cada tupla é acessada uma única vez. Assim, o número de acesso a blocos é igual à soma do número de blocos das duas relações. [Silberschatz et al. 2006]

```

1 pr := endereço da primeira tupla de r;
2 ps := endereço da primeira tupla de s;
3 while (ps ≠ nulo and pr ≠ nulo) do
4     begin
5         ts := tupla para qual ps aponta;
6         Ss := {ts};
7         configure ps para apontar para a próxima tupla de s;
8         acabou := falso;
9         while (not acabou and ps ≠ nulo) do
10            begin
11                ts' := tupla para qual ps aponta;
12                if (ts'[AtribJunção] = ts[AtribJunção])
13                    then begin
14                        Ss := Ss ∪ {ts'};
15                        configure ps para apontar para a próxima tupla de s;
16                    end
17                else acabou := verdadeiro;
18            end
19            tr := tupla para qual pr aponta;
20            while (pr ≠ nulo and tr[AtribJunção] < ts[AtribJunção]) do
21                begin
22                    configure pr para apontar para a próxima tupla de r;
23                    tr := tupla para qual pr aponta;
24                end
25            while (pr ≠ nulo and tr[AtribJunção] = ts[AtribJunção]) do
26                begin
27                    for each ts in Ss do
28                        begin
29                            adicione ts▷◁ tr ao resultado;
30                        end
31                        configure pr para apontar para a próxima tupla de r;
32                        tr := tupla para qual pr aponta;
33                    end
34    end

```

Figura 2. Junção merge.

4. Modificações no algoritmo de junção *merge* para execução com tabelas heterogêneas

No presente trabalho, o algoritmo da Figura 2 foi implementado com adaptações para suportar tabelas transpostas. Como entrada, recebe arquivos com tabelas HTML que são extraídas e salvas em *LinkedList<LinkedList<String>>*, que é uma estrutura ordenada com duas dimensões de *Strings*. A extração e a forma como o algoritmo de junção é executado diferenciam as estratégias entre si. A primeira abordagem consistiu em extrair a tabela do formato HTML sem alterar sua estrutura, para após executar um método que a transpõe. Assim, a tabela estará no formato tradicional e o algoritmo de junção *merge* pode ser aplicado sem alterações. Como o custo da transposição é elevado, implementou-se outra abordagem que, diferente da anterior, trabalha com a tabela em seu formato original. Para isso, o algoritmo de junção *merge* foi adaptado para interpretar linha como coluna e vice-versa. Visando um desempenho melhor, a terceira abordagem realiza a transposição da tabela durante sua extração do arquivo HTML.

Para identificar situações onde os rótulos das tabelas encontram-se na primeira coluna, e não na primeira linha, utilizou-se um método que infere a afinidade de um conjunto

de valores com um possível rótulo através de motores de busca [da Silva et al. 2007]. Primeiramente obtém-se as pontuações dadas à afinidade dos valores de cada coluna com seu respectivo valor na primeira linha. De forma análoga, o mesmo é realizado com os valores de cada linha, considerando como rótulo candidato o valor presente na primeira coluna de cada. Compara-se, então, as médias das pontuações. Se os valores possuem maior afinidade com os rótulos da primeira linha, a tabela é considerada *tradicional*; caso contrário, é dita *transposta*.

4.1. Abordagem 1: Transposição de tabela após extração

Na primeira solução implementada, as tabelas tradicionais e transpostas são extraídas da mesma forma, conforme o algoritmo presente na Figura 3. O arquivo com a tabela HTML é percorrido linha a linha e a cada *tag* de linha encontrada (*<tr>*), uma nova *LinkedList<String>* é adicionada à estrutura principal. Na sequência, a cada *tag* de coluna encontrada (*<td>*), uma nova *String* é adicionada à estrutura referente à linha em questão. Dessa forma, não se altera a estrutura das tabelas originais. Para realizar a junção *merge*, primeiramente a tabela que não se encontra no formato tradicional é transposta conforme o algoritmo da Figura 4 de complexidade $O(n^2)$.

```

1 entrada := primeira linha do arquivo HTML;
2 i := 0;
3 while (entrada ≠ nulo) do
4     begin
5         if (entrada > "<tr>")
6             then begin
7                 linhaTmp := nulo;
8                 fimDeLinha := falso;
9                 j := 0;
10                while (not fimDeLinha) do
11                    begin
12                        if (entrada > "<td>")
13                            then begin
14                                linhaTmp[j] := valor em entrada
entre "<td>" e "</td>";
15                                j := j + 1;
16                            end
17                            entrada := próxima linha do arquivo HTML;
18                            if (entrada > "</tr>")
19                                then begin
20                                    fimDeLinha := verdadeiro;
21                                    tabela[i] := linhaTmp;
22                                end
23                                end
24                                i := i + 1;
25                            end
26                            entrada := próxima linha do arquivo HTML;
27 end

```

Figura 3. Algoritmo para extração da tabela HTML.

4.2. Abordagem 2: Adaptação do algoritmo de junção *merge*

A segunda abordagem extrai as tabelas da mesma forma descrita anteriormente, porém não realiza a transposição. Para suportar tabelas transpostas, foi implementado um método que retorna determinada coluna como se fosse linha, conforme a Figura 5. Este método é executado durante o algoritmo de junção *merge* a cada vez que *pr* ou *ps*, dependendo de qual tabela está transposta, são configuradas para apontar para a próxima

```

1 numColunas := número de colunas da tabela invertida;
2 numLinhas := número de linhas da tabela invertida;
3 for i := 0 until numColunas do
4     begin
5         for j := 0 until numLinhas do
6             begin
7                 linhaTmp[j] := tabelaInvertida[j][i];
8                 j := j + 1;
9             end
10            tabelaTradicional[i] := linhaTmp;
11            i := i + 1;
12        end

```

Figura 4. Algoritmo de transposição de tabela.

tupla (linhas 1, 2, 7, 15, 22 e 31 da Figura 2). O método possui complexidade $O(n)$ e é executado uma vez para cada coluna da tabela. Dessa forma, o aumento de complexidade total em relação à junção de tabelas tradicionais é $O(n^2)$.

```

1 numLinhas := número total de linhas da tabela;
2 j := índice da coluna que se deseja transpor;
3 for i := 0 until numLinhas do
4     begin
5         colunaTransposta[i] := tabelaInvertida[i][j];
6         i := i + 1;
7     end

```

Figura 5. Algoritmo que obtém determinada coluna como linha.

4.3. Abordagem 3: Transposição de tabela durante extração

Na terceira estratégia adotada, a diferença está na extração das tabelas transpostas. O algoritmo está exposto na Figura 6. Antes de alocar a estrutura para armazenamento do conteúdo da tabela, conta-se quantas *tags* de linha e de coluna o arquivo possui. Este trecho difere a abordagem das anteriores e possui complexidade $O(n)$. Após, percorre-se o arquivo para extrair seu conteúdo, com uma iteração a cada linha. O primeiro valor é salvo na primeira posição da primeira $\langle\text{LinkedList}<\text{String}\rangle$; o segundo, na primeira posição da segunda $\langle\text{LinkedList}<\text{String}\rangle$ e assim por diante, até percorrer toda a tabela do arquivo. Dessa forma, as tabelas transpostas são salvas com a mesma estrutura de uma tabela tradicional e o algoritmo de junção *merge* pode ser executado sem alterações.

5. Experimentos

5.1. Configuração

Foram realizados experimentos com as três abordagens para verificar sua escalabilidade para a Web. O conjunto de dados para os testes consistiu em 20 tabelas HTML sobre o domínio de filmes extraídas da Web. A Figura 7 mostra duas delas. Para testar a aplicação do algoritmo no pior caso, onde toda tupla encontra correspondente para junção, as tabelas foram alteradas e tiveram seu número de linhas aumentado. Dessa forma, as abordagens foram testadas com tabelas que possuíam entre 1, 250, 500 ou 1000 tuplas com valores iguais para o atributo utilizado na junção (“*Title*”).

```

1 numLinhas := 0;
2 numColunas := 0;
3 entrada := primeira linha do arquivo HTML;
4 while (entrada ≠ nulo) do
5   begin
6     if (entrada > "<tr>")
7       then begin
8         fimDeLinha := falso;
9         while (numLinhas = 0 and not fimDeLinha) do
10           begin
11             if (entrada > "<td>")
12               then begin
13                 numColunas := numColunas + 1;
14               end
15             entrada := próxima linha do arquivo HTML;
16             if (entrada > "</tr>")
17               then begin
18                 fimDeLinha := verdadeiro;
19                 numLinhas := numLinhas + 1;
20               end
21             end
22           end
23         entrada := próxima linha do arquivo HTML;
24       end
25     entrada := primeira linha do arquivo HTML;
26   i := 0;
27   while (entrada ≠ nulo) do
28     begin
29       if (entrada > "<tr>")
30         then begin
31           fimDeLinha := falso;
32           j := 0;
33           while (not fimDeLinha) do
34             begin
35               if (entrada > "<td>")
36                 then begin
37                   linha := nulo;
38                   for k := 0 until numLinhas do
39                     begin
40                       linha[k] := nulo;
41                     end
42                   tabela[j] := linha;
43                   coluna := valor entre "<td>" e "</td>"
44                 em entrada;
45                 tabela[j][i] := coluna;
46                 j := j + 1;
47               end
48             entrada := próxima linha do arquivo HTML;
49             if (entrada > "</tr>")
50               then begin
51                 fimDeLinha := verdadeiro;
52               end
53             i := i + 1;
54           end
55         end

```

Figura 6. Algoritmo que transpõe a tabela durante sua extração.

Os testes foram realizados em ambiente Windows, com processador AMD Athlon II 245 Dual-Core com 4GB de memória principal no IDE NetBeans 7.0.1. Foram testados casos onde nenhuma, uma e as duas tabelas encontravam-se transpostas, sempre com 50000 repetições para observar diferenças significativas no desempenho.

5.2. Resultados

A Figura 8 apresenta os gráficos com o tempo de execução, medido em segundos, em função do número de tuplas com valor equivalente no atributo considerado para a junção.

Title	Albatross	Contraband	Tangled (I)
Language	English	English	English
Release date	14 October 2011	11 January 2012	7 January 2011
Production Company	Isle of Man Film	Universal Pictures	Walt Disney Pictures

Title	Genre	Written by	Aspect Ratio	Budget
Albatross	Drama	Tamzin Rafn	2.35 : 1	\$33,000,000
Beauty and the Beast	Animation Fantasy	Linda Woolverton	1.66 : 1	\$25,000,000
Contraband	Action, crime	Aaron Guzikowski	2.35 : 1	\$41,000,000
Joyful Noise	Comedy	Todd Graff	2.35 : 1	\$18,000,000
The Iron Lady	Drama Biography	Abi Morgan	2.35 : 1	\$13,000,000

Figura 7. Algumas tabelas do conjunto de teste.

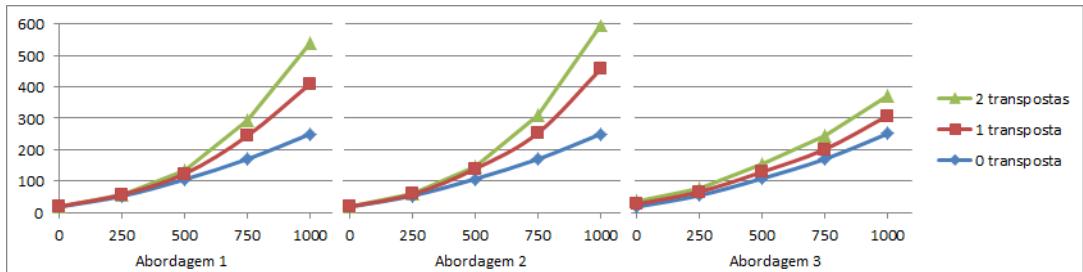


Figura 8. Gráficos com o tempo de execução em função do número de linhas das tabelas utilizadas para as três abordagens.

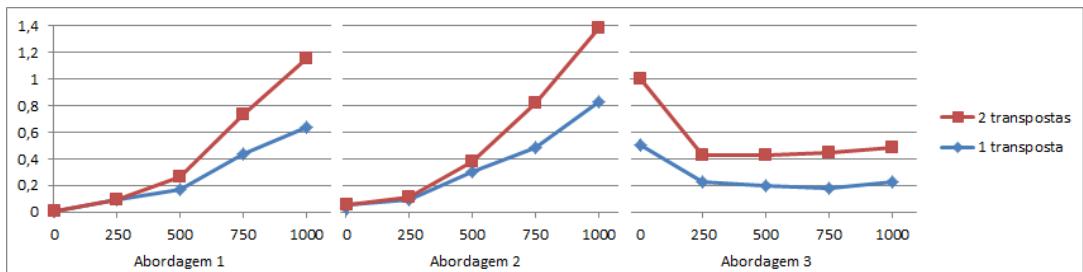


Figura 9. Gráficos comparando o desempenho de 1 e 2 tabelas transpostas em relação a execução do algoritmo com 2 tabelas tradicionais.

Percebe-se aumento maior nas abordagens 1 e 2, que adicionam complexidade quadrática ao caso tradicional (nenhuma tabela transposta), e praticamente linear na abordagem 3, que adiciona a complexidade de $O(n)$ ao caso tradicional. Pelos gráficos da Figura 9 percebe-se que para tabelas com elevado número de linhas, as abordagens 1 e 2 possuem desempenho muito mais lento quanto a junção envolve tabela transposta. No caso onde

é feita a transposição, o tempo chega a ser 64% mais lento quando apenas uma tabela é tradicional e 115% quando nenhuma é. Os resultados mostraram que a abordagem de ler a coluna como se fosse uma linha também não foi satisfatória para tabelas transpostas. O aumento do tempo chega a 83% para uma tabela transposta e 139% para duas. Já a estratégia de transpor a tabela durante sua extração, com complexidade adicional linear, mostrou-se viável para tabelas com elevado número de linhas. O aumento do tempo estabilizou em cerca de 20% a mais do que o caso tradicional para uma tabela transposta e cerca de 45% para duas. Assim, pode-se afirmar que esta abordagem é a mais adequada para a Web, tendo em vista sua enorme quantidade de dados.

6. Conclusão e trabalhos futuros

Este artigo descreve a proposta de adaptação do algoritmo de junção *merge* para suportar tabelas HTML transpostas, onde os rótulos encontram-se na primeira coluna, e não na primeira linha. Foram utilizadas três estratégias diferentes na implementação, a fim de encontrar o melhor desempenho na operação. De acordo com os experimentos, observou-se que o melhor método proposto chega a ser 2,86 vezes melhor do que o pior em questão de tempo de desempenho.

Como trabalhos futuros, destacam-se: (i) adaptar do algoritmo a mais tipos de tabelas HTML encontradas na Web; (ii) considerar a similaridade para realizar a operação de junção; (iii) utilizar um dicionário de sinônimos para detectar rótulos equivalentes; e (iv) utilizar um *crawler* focado para aumentar o conjunto de testes.

Referências

- Cafarella, M. J., Halevy, A., Wang, D. Z., Wu, E., and Zhang, Y. (2008). Webtables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1:538–549.
- da Silva, A. S., Barbosa, D., Cavalcanti, J. M. B., and Sevalho, M. A. S. (2007). Labeling data extracted from the web. In *Proceedings of the 6th International Conference on Ontologies, DataBases, and Applications of Semantics*, pages 1099–1116. Springer.
- Elmeleegy, H., Madhavan, J., and Halevy, A. (2009). Harvesting relational tables from lists on the web. *Proc. VLDB Endow.*, 2:1078–1089.
- Lin, C. X., Zhao, B., Weninger, T., Han, J., and Liu, B. (2010). Entity relation discovery from web tables and links. In Rappa, M., Jones, P., Freire, J., and Chakrabarti, S., editors, *WWW*, pages 1145–1146. ACM.
- Mergen, S. L. S., Freire, J., and Heuser, C. A. (2010). Querying structured information sources on the web. *IJMSO*, 5(3):208–221.
- Silberschatz, A., Korth, H., and Sudarshan, S. (2006). *Sistema de Banco de Dados*. Editora Campus, Rio de Janeiro, 5 edition.
- Venetis, P., Halevy, A., Madhavan, J., Pasca, M., Shen, W., Wu, F., Miao, G., and Wu, C. (2011). Recovering semantics of tables on the web. *PVLDB*, 4(9):528–538.

Abordagens de Particionamento de Dados para Redes Sociais

Bruno S. Velasco¹, Carmem S. Hara¹

¹Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – 81.531-980 – Curitiba – PR – Brasil

Abstract. *Online Social Networks (OSN) have never been so popular as it is nowadays. Due to its fast growth, designing a suitable infrastructure demands time and resources. Thus, a common approach adopted by several OSN is to set them on the Cloud. Although data gets spread throughout servers, a good balancing of data is barely achieved. To overcome this, several methods have been developed towards a good partitioning. This paper presents the main techniques for OSNs partitioning: general approaches, such as Full Replication and Distributed Hashing, and more specific methods for OSN such as Partitioning and Replication and Time-dependent Partitioning. Also, a comparative study is presented highlighting which approaches are best for each possible context.*

Resumo. *Redes Sociais Online (RSO) nunca foram tão populares quanto são hoje. Devido ao seu rápido crescimento, especificar uma boa infraestrutura demanda tempo e recursos. Por esse motivo é comum hospedá-las na nuvem. Mesmo com os dados distribuídos, um bom balanceamento dificilmente é contemplado. Assim, diversas técnicas tem sido desenvolvidas para alcançar um bom particionamento. Este artigo apresenta algumas técnicas para o particionamento de RSO. Elas são classificadas como de uso geral, como Replicação total e Hash distribuído, e métodos específicos para RSO, como o SPAR e Particionamento dependente de tempo. Ao final, um estudo comparativo é apresentando destacando quais abordagens são mais apropriadas para cada cenário.*

1. Introdução

Uma Rede Social Online (RSO) é um serviço que permite a interação entre usuários através de meios bem conhecidos, como troca de texto ou mídia. Usuários desta rede podem estabelecer ligações e a quantidade de usuários pode variar de dezenas até milhares. Cada ligação feita pode ser entendida como uma “amizade” que ocorre entre os participantes. Atualmente, há mais de 200 RSO conhecidas cuja maioria tem mais de 1 milhão de usuários [Wikipedia 2011]. O Facebook lidera o topo desta lista com cerca de 750 milhões de usuários registrados [Facebook 2011]. Esta é uma característica marcante das RSO: os usuários se registram de forma acelerada. O Twitter, por exemplo, cresceu cerca de 1382% entre Fevereiro e Março de 2009 [Cnet 2011, Carrasco et al. 2011]. Esse rápido crescimento compromete o desempenho do sistema caso ele não tenha sido desenvolvido para dar suporte a tal carga de trabalho. Uma solução comum é atualizar o hardware existente. Essa abordagem é conhecida como *escalabilidade vertical*, porque a capacidade é incrementada sobre os mesmos recursos. Esta decisão não só é financeiramente cara como ineficiente, a curto prazo, já que o hardware tende tornar-se obsoleto em pouco tempo. Além do mais, não é uma solução escalável [Pujol et al. 2010, Pujol et al. 2009].

Esse rápido crescimento resulta em um dilema entre os desenvolvedores de RSO: “É preferível alocar recursos humanos no desenvolvimento de novas funcionalidades ou

focar na construção de um sistema altamente escalável, capaz de dar suporte a um grande volume de dados?” [Pujol et al. 2010, Pujol et al. 2009]. A primeira escolha é a mais comum e claramente a mais fácil. Por outro lado, tal escolha pode resultar em uma situação conhecida como “morte-pelo-sucesso”, no qual usuários são atraídos pelas funcionalidades do sistema e este, por sua vez, não é capaz de comportar tal demanda. Uma situação semelhante ocorreu com a RSO *Friendstar* [Scalability 2011].

No entanto, a segunda opção, que consiste em criar uma base sólida para acomodar uma carga de trabalho que pode nunca vir a acontecer, dispõe um grande investimento em termos de recursos humanos e financeiros. Tais esforços podem não ser recompensados justamente por não se saber se esta RSO terá o retorno esperado. Porém, expansões futuras tornam-se mais simples e baratas pois novos servidores de baixo desempenho podem ser utilizados. Esta abordagem é conhecida como *escalabilidade horizontal* [Pujol et al. 2010, Pujol et al. 2009].

Tornar uma RSO escalável não é uma tarefa trivial. Consequentemente, o seu particionamento torna-se complexo [Carrasco et al. 2011] [Viswanath et al. 2009] [Curino et al. 2010] [Pujol et al. 2010]. Como ela é uma comunidade, uma possível representação computacional de uma RSO é um grafo cujos nodos representam os usuários, e as arestas, as ligações (amizades) entre os participantes. Esse grafo é conhecido como *grafo de amizades*. Entretanto, tal grafo não contém nenhuma informações sobre a coesão das ligações. Por isso, para um determinado intervalo de tempo, a *rede de atividade* [Viswanath et al. 2009] pode ser considerada, consiste no subgrafo do grafo de amizades originado da interação entre os pares com peso nas arestas, que representa o volume de interações entre o par.

Apesar dos desafios que uma solução escalável apresenta, ela é necessária para comportar um alto tráfego de dados. Um destes desafios é o particionamento de dados. O problema em questão é o quanto eficiente e “bom” um particionamento pode ser, ou seja, fazer com que “dados afins” (usuários) sejam armazenados nos mesmos servidores, evitando sobrecarga da rede. A definição de “dados afins” não é precisa e pode variar de meramente ser uma ligação de amizade, para a frequência na qual um par de usuários interage em um dado período de tempo. O objetivo do particionamento é explorar a localidade espacial entre os dados de um mesmo nodo de rede, evitando a comunicação desnecessária entre os servidores.

Um bom particionamento de uma RSO depende diretamente de como os desenvolvedores definem “dados afins”, ou seja, sob quais critérios os dados serão particionados. Atualmente, existem as seguintes técnicas de distribuição de dados para RSO:

1. **Abordagens gerais:** são técnicas agnósticas do conceito de dados afins. Dentre elas podem ser citadas as Tabelas de Dispersão Distribuídas (*Distributed Hash Tables - DHT*) e Replicação Total.
2. **Particionamento baseado no grafo de amizade:** são técnicas que utilizam a estrutura do grafo de amizades para determinar o particionamento de usuários e replicação dos dados. Dentre elas, existe o SPAR [Pujol et al. 2010], que é uma plataforma de particionamento e replicação capaz de escalar, transparentemente, a estrutura de grafo para atingir localidade de dados, ao mesmo tempo que minimiza as replicações.
3. **Particionamento dependente de tempo:** são técnicas que utilizam o grafo de

amizades, em conjunto com o espaço temporal para definir o conceito de dados afins, que constitui a rede de atividade. Uma das técnicas que segue essa abordagem foi proposta em [Carrasco et al. 2011], que busca agrupar pares de usuários com maior taxa de interação nos mesmos servidores.

O restante desse trabalho está organizado como segue. A Seção 2 descreve o comportamento de uma RSO de acordo com Viswanath et al. A Seção 3 detalha cada uma das abordagens principais de particionamento. A Seção 4 contém um breve estudo comparativo entre as abordagens de particionamento e a Seção 5 traz o fechamento desse trabalho.

2. Características de uma RSO

Alguns estudos tem sido desenvolvidos para melhor compreender o comportamento e particularidades de uma RSO [Viswanath et al. 2009, Benevenuto et al. 2009]. Embora diversos estudos sejam baseados no grafo de amizades e na rede de atividades, poucos consideram sua modificação ao longo do tempo, ou seja, se as ligações entre os usuários tornam-se mais fortes ou fracas. Viswanath et al. propuseram analisar essas modificações sob dois aspectos: microscópico e macroscópico. O primeiro nível atem-se em estudar o comportamento para cada par de usuários, enquanto o nível macro avalia o quanto usuários individuais alteram o comportamento e propriedades da rede.

Os dados empregados para a avaliação foram baseados no Facebook. A forma de interação escolhida para o estudo foi postagens no mural de recados que cada usuário tem em seu perfil. Esse método de interação pode apresentar algumas limitações se comparado às outras maneiras de interatividade providas pelo Facebook como mensagens, fotos, vídeos, e conversa [Viswanath et al. 2009]. Porém, o mural foi escolhido por conter o *timestamp* que é utilizado para monitorar as ações dos usuários ao longo do tempo.

Uma característica de alto nível observada foi que a rede de atividade representa cerca de 12,2% do grafo de amizades. Isto confirma um estudo anterior que afirma que o grau de um nodo de uma rede de atividade (ou seja, o número de amigos com os quais usuários interage na média) é significamente menor que o grau de um nodo da rede de amigos [Viswanath et al. 2009].

2.1. Interação de um par de usuários

O primeiro comportamento interessante que foi observado remete à distribuição do número de postagens feitas no mural dos usuários. Esse comportamento é mostrado na Figura 1, que tem no eixo *y* a distribuição cumulativa e no eixo *x* o número de postagens para cada par. A mediana do número de postagens por par de usuários é 2, sendo que 81% dos pares não trocaram mais que 5 postagens [Viswanath et al. 2009].

Essa distribuição sugere dois níveis de usuários: com alta e baixa taxa de interação. Os autores consideraram um usuário de baixa interatividade se esse mandou não mais que 5 postagens no mural, caso contrário, tal usuário é considerado de alta interatividade. Para os de baixa interatividade, foi constatado que as postagens ocorreram, ou por causa da primeira mensagem, ou devido ao aplicativo do Facebook que lembra do aniversário de seus amigos [Viswanath et al. 2009]. Na verdade, 54% dos pares do grupo de baixa interatividade tiveram ao menos uma postagem relacionada a felicitações de aniversário.

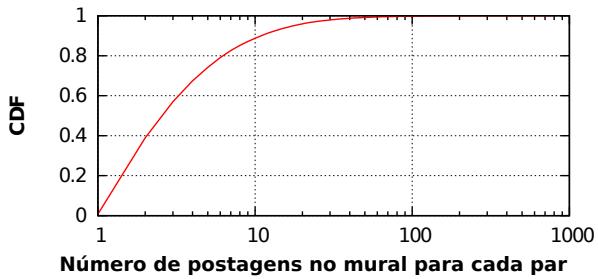


Figura 1. Distribuição do número de postagens no mural entre cada par [Viswanath et al. 2009].

O grupo de alta interatividade teve um comportamento diferente. A maioria destes usuários trocou a sua primeira mensagem tão logo a ligação foi efetuada. Além disso, a interação decaiu ao longo do tempo. A Figura 2 ilustra a taxa de interatividade no tempo, baseada nas postagens enviadas ao mural desde a criação da ligação. As postagens são agrupadas por mês.

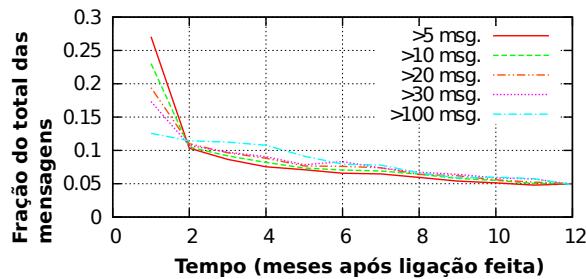


Figura 2. Fração das mensagens enviadas ao longo do tempo [Viswanath et al. 2009].

Após observar a redução constante, os autores voltaram suas atenções para quanto longa uma ligação entre os pares permanece na rede de atividade. No total, apenas 23% dos pares do grupo de alta interatividade se mantiveram durante o intervalo de 12 meses. Viswanath et al. concluíram que para ambos os grupos, de alta e baixa interatividade, usuários tendem a interagir menos com o passar do tempo.

2.2. Evolução da rede ao longo do tempo

Após ter estudado o comportamento entre os pares de usuários, os autores pesquisaram o impacto na rede que esse comportamento ocasiona. Para tanto, 9 *snapshots* foram realizadas na rede, com 90 dias de intervalo cada. Eles estavam interessados em quantos dos nodos permaneciam na rede ao longo do tempo. Para isso, o conceito de *semelhança* foi empregado, que consiste em tomar 2 *snapshots* consecutivas e ver quantos dos nodos residem em ambas as amostras. A Figura 3 apresenta a análise para todas as amostras. É possível observar que as duas primeiras *snapshots* possuem semelhança de 60% aproximadamente, e essa taxa reduz levemente ao longo do tempo. Apesar desta mudança, cerca de 45% das ligações permaneceram ativas durante o período.

Acerca das propriedades do grafo resultante da rede de atividade, os autores utilizaram algumas métricas para estudo de grafos e, interessantemente, todos os testes man-

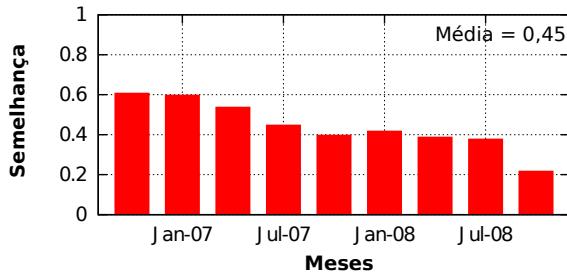


Figura 3. Semelhança da rede de atividade [Viswanath et al. 2009].

tiveram as métricas estáveis para todas as amostras, evidenciando que grandes variações entre as ligações individuais não afetam a rede como um todo.

3. Abordagens para distribuição de dados de uma RSO

Nesta seção são apresentadas algumas técnicas propostas na literatura para o particionamento e replicação de dados de uma RSO. Dentre as abordagens gerais são descritas as técnicas de Tabela de Dispersão Distribuída (**DHT**) e **Replicação total**. Essas estratégias não consideram a afinidade entre os usuários para distribuir os dados como suas ligações ou interações. Assim sendo, elas não garantem a localidade dos dados. Como representante da abordagem baseada no grafo de amizade é apresentado o **SPAR** [Pujol et al. 2010]. Além disso, é apresentada a proposta de [Carrasco et al. 2011], que segue a abordagem de particionamento dependente de tempo. Ambas consideram a afinidade dos usuários.

3.1. Abordagens gerais

As técnicas de DHT e Replicação total são de uso geral e também podem ser aplicadas para RSO. A Replicação total, largamente utilizada por sua simplicidade [Pujol et al. 2010], consiste simplesmente em replicar todos os nodos (usuários) do grafo por todos os servidores. Essa operação requer grande capacidade de armazenamento e possui um alto custo computacional já que cada réplica precisa estar sincronizada. Sua vantagem, porém, é que a localidade espacial é total, uma vez que todo nodo tem sua réplica em cada servidor. A DHT, por sua vez, agrupa usuários mediante uma chave. Essa chave pode ser o nome de usuário ou um valor aleatório. Opcionalmente, replicação pode ser empregada para oferecer uma melhor localidade espacial. A DHT é amplamente utilizada para a distribuição de dados por ser um método escalável [Pujol et al. 2010], sem controle centralizado e tolerante a falhas.

No topo da Figura 4 é ilustrada uma RSO com 10 usuários e 15 ligações (amizade bidirecional). Imediatamente abaixo, são ilustradas as abordagens de Replicação total (a), DHT (b) e DHT com replicação (c), juntamente com as métricas para comparação, que são o *tráfego gerado na leitura* de um usuário, *tráfego gerado na escrita* de um usuário e o espaço em memória utilizado. Em (a), Replicação total atinge 100% de localidade espacial (*tráfego na leitura* = 0), porém requer alta capacidade de armazenamento (*memória* = 10) e sincronização constante (*tráfego na escrita* = 10). Em (b), a DHT é empregada para distribuir os usuários sem considerar a ligação entre os nodos. Apesar de exigir menos espaço de armazenamento e dispensar sincronização, não é oferecida localidade de

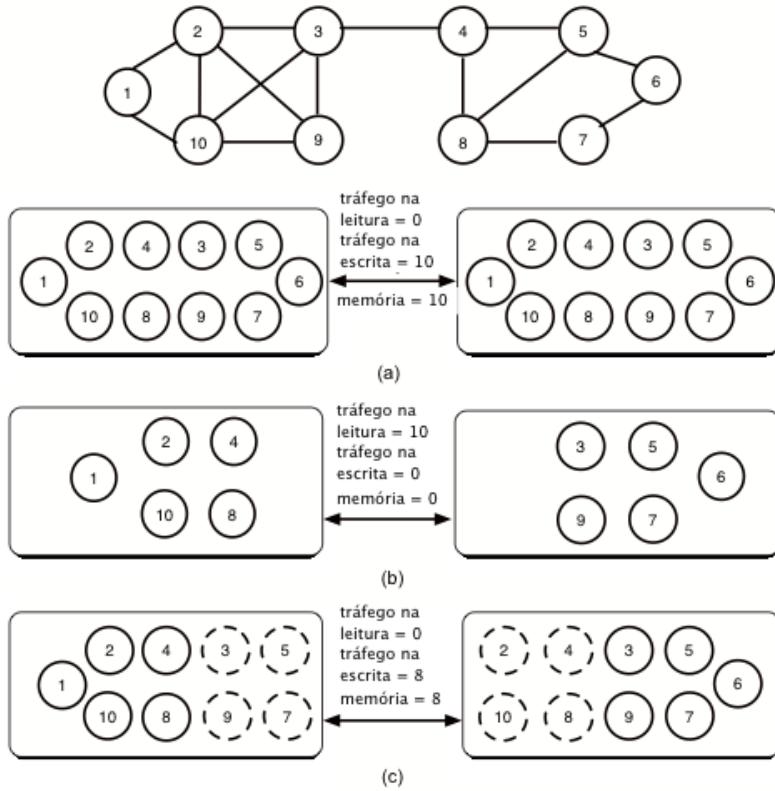


Figura 4. Parte de uma RSO sendo particionando em 2 servidores usando (a) Replicação total, (b) DHT, (c) DHT com replicação dos vizinhos [Pujol et al. 2010].

dados. Assim, para amenizar o custo da localidade, em (c) é empregada replicação dos dados, que exige armazenamento e sincronização extras.

3.2. Particionamento e replicação (SPAR)

Essa abordagem foi introduzida por Pujol et al., sendo um dos primeiros métodos de particionamento para RSO a considerar a ligação entre os usuários, ao contrário das abordagens gerais. O SPAR atua como um *middleware*, permitindo que o usuário escolha entre um sistema de armazenamento relacional ou chave-valor. A distribuição dos dados é feita de forma automática, particionando e replicando os dados conforme necessário durante a execução, sem a necessidade de interrupção do serviço. Quando alguma alteração no grafo é realizada, seja pela adição ou remoção de nodos e ligações, ele é capaz de decidir quais ações tomar [Pujol et al. 2010].



Figura 5. Particionamento usando SPAR [Pujol et al. 2010].

O particionamento resultante do SPAR garante que todos os usuários da RSO e seus vizinhos diretos (conhecidos como *vizinho a um-salto*) estão armazenados no mesmo servidor, atingindo melhor localidade espacial para os dados mais importantes (amigos,

seguidores, etc.) [Pujol et al. 2010]. A Figura 5 mostra como esse método particiona a RSO referente à Figura 4; note que somente os vizinhos a um-salto relevantes são replicados em ambos os servidores (*tráfego na leitura = 0*). Ao final, os custos de armazenamento, leitura e escrita na RSO, são inferiores comparadas às abordagens de propósito geral.

Para avaliar o desempenho do sistema, os autores realizaram experimentos focados na sobrecarga envolvida para a replicação, considerando que o gargalo está diretamente relacionado com a replicação de usuários. A fonte de dados utilizada envolveu dados reais do Orkut, Twitter e Facebook, coletados em diferentes períodos.

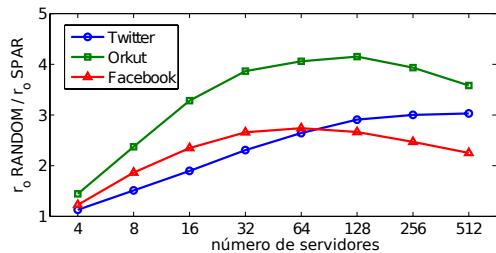


Figura 6. SPAR versus DHT [Pujol et al. 2010].

A Figura 6 ilustra um dos experimentos realizados, a fim de verificar o quanto inferior é a sobrecarga para o particionamento do SPAR se comparado com a DHT. A notação (r_o) expressa o número total de réplicas necessárias para manter a localidade semântica. No eixo y do gráfico está plotada a relação de sobrecarga referente à DHT comparada com o SPAR, enquanto o eixo x representa o número de servidores. Por exemplo, para 128 servidores, o SPAR utilizou 4 vezes menos réplicas que a DHT para distribuir dados provenientes do Orkut.

À medida que novas ligações (arestas) são adicionadas, algumas ações ocorrem para manter o balanceamento do grafo a fim de manter a localidade. A Figura 7 mostra essas ações, evidenciando que na maior parte do tempo o SPAR permanece em espera, apesar de algumas ações ocorrerem.

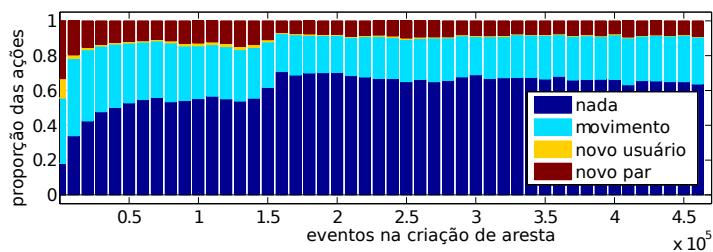


Figura 7. Percentual de cada ação promovida pelo SPAR ao longo da criação de ligações [Pujol et al. 2010].

Pujol et al. concluem seus resultados demonstrando que o SPAR é capaz de alcançar 100% de localidade com uma sobrecarga de operação mínima, apresentando um desempenho muito superior à DHT.

3.3. Particionamento dependente de tempo

O método proposto por [Carrasco et al. 2011] visa partitionar uma RSO baseada na interação dos usuários participantes. O conceito de usuários afins envolve a dimensão tempo, ou seja, a frequência com que eles interagem.

Baseado no trabalho de Viswanath et al., apresentado na Seção 2, a rede de atividade é construída sobre o grafo de amizades correspondente considerando a dimensão tempo. Uma vantagem desta abordagem é que a rede de atividade é consideravelmente menor que o grafo de amizades em si, (12,2% como observado na Seção 2).

A Figura 8 mostra uma pequena RSO com 7 nodos. As mensagens trocadas entre os usuários são representadas por um número entre colchetes sobre a ligação, como pode ser visto na Figura 8(a). Por exemplo, no período 1, os usuários 1 e 2 trocaram 100 mensagens enquanto os usuários 2 e 3 não interagiram; no período 2, usuários 1 e 2 não interagiram, ao contrário dos usuários 2 e 3 que trocaram 100 mensagens. Quando a replicação é utilizada para garantir a localidade de vizinhança a um-salto, tal como SPAR, todas as 200 mensagens são replicadas, como na Figura 8(b). Porém, quando a rede de atividade é partitionada em períodos diferentes, por existir menos mensagens entre usuários em um intervalo de tempo, nenhuma replicação é executada, como na Figura 8(c).

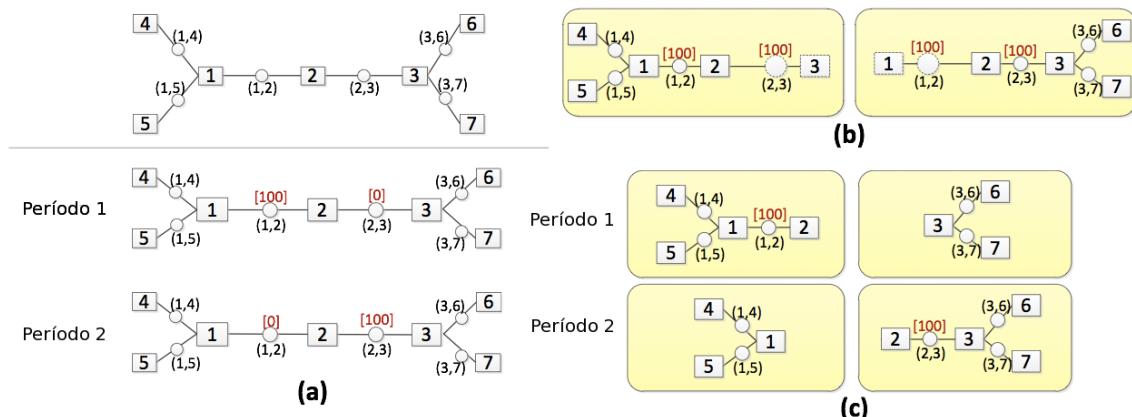


Figura 8. Exemplo de uma pequena RSO sob alguns cenários de particionamento [Carrasco et al. 2011].

A avaliação desse modelo buscou evidenciar quanto da localidade espacial foi corretamente contemplada. Para tal, o mesmo conjunto de dados do Facebook da Seção 2 foi utilizado. A consulta consiste nas 6 mensagens mais recentes dos amigos dos amigos (conhecido como *vizinhança a dois-saltos*). Ao todo, 3 métodos de partitionamento foram comparados: o próprio algoritmo, DHT *hash_p1* (todas mensagens geradas pelo mesmo usuário são agrupadas em uma partição) e outra modalidade de DHT *hash_p1p2*, com a diferença que todas as mensagens trocadas entre um par de usuários são agrupadas na mesma partição [Carrasco et al. 2011].

A Figura 9 evidencia que para todos os números de partições, mais de 80% das consultas acessaram no máximo 3 partições com o algoritmo proposto. As autoras concluem que esse método é consideravelmente melhor para a localidade dos dados se comparado com os outros dois algoritmos.

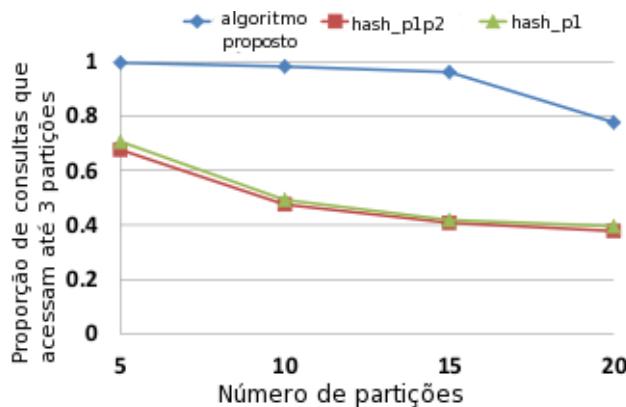


Figura 9. Proporção de consultas que acessam no máximo 3 partições [Carrasco et al. 2011].

4. Discussão

A importância de um bom particionamento de dados pode ser traduzida através da localidade espacial dos dados, que consiste em agrupar dados afins nas mesmas partições, evitando tráfego de rede. Quanto mais usuários “afins” pertencem a um mesmo nodo de rede, melhor foi o método de particionamento.

Existem dois tipos de escolha para particionar uma RSO: abordagens gerais e específicas. A primeira opção traz a vantagem da simplicidade, atingindo bons resultados se comparados às abordagens de escalabilidade vertical. A Replicação total oferece uma melhor localidade dos dados, apesar desse requerer mais espaço de armazenamento para comportar as réplicas. A DHT, por outro lado, exige menos espaço de armazenamento porém oferece menor localidade dos dados, mesmo com réplicas.

A segunda opção oferece algoritmos mais robustos, uma vez que eles consideram a propriedade de semelhança e coesão dos usuários. O SPAR atinge altos níveis de localidade, chegando a 100% quando apenas vizinhança a um-salto é considerada. Essa abordagem mostrou ter baixa latência e maior vazão quando comparado com Cassandra [Pujol et al. 2010], o sistema de armazenamento chave-valor utilizado pelo Facebook que faz distribuição de dados por DHT. Ele também é capaz de servir 3 vezes mais requisições se comparado com algoritmos comerciais baseados em DHT. Sua desvantagem, porém, é que exige muitas réplicas: para 512 partições, no Facebook, uma média de 7 réplicas são usadas enquanto que para alguns usuários suas réplicas existem em todos os servidores. O algoritmo baseado no tempo, por outro lado, possui a vantagem de trabalhar com a rede de atividade, cujo grafo é menor e mais representativo. Apesar de sua localidade ser ligeiramente menor que a solução SPAR, ele requer muito menos réplicas.

A escolha é necessária entre um método com uma perfeita predição sobre a localidade de dados ao preço de grande espaço de armazenamento, e o uso de um algoritmo com considerável localidade ao preço de armazenamento bastante inferior.

5. Conclusão

O presente trabalho apresentou alguns métodos para particionamento de Redes Sociais Online, incluindo abordagens gerais como **Replicação total** e **DHT**, e métodos específicos tais como **Particionamento e replicação (SPAR)** e **Particionamento depen-**

dente de tempo. Foi observado que ambos os métodos específicos tem um melhor desempenho se comparados aos métodos tradicionais e comerciais utilizados para esse propósito. Apesar do SPAR alcançar 100% de localidade de dados para operações de vizinhança a um-salto, ele requer muito mais espaço de armazenamento que o dependente de tempo, que se baseia na frequência da interação entre os usuários. A análise desta frequência evidencia uma rede conhecida como rede de atividade, que contém propriedades interessantes que são utilizadas por esse último método, sendo capaz de atingir altos níveis de localidade ao preço de poucas réplicas.

Após destacar as vantagens e desvantagens de cada abordagem, uma breve comparação foi realizada concluindo que existe uma escolha a se fazer entre capacidade de armazenamento e precisão na localidade dos dados.

Referências

- Benevenuto, F., Rodrigues, T., Cha, M., and Almeida, V. (2009). Characterizing user behavior in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 49–62. ACM.
- Carrasco, B., Lu, Y., and da Trindade, J. (2011). Partitioning social networks for time-dependent queries. In *Proceedings of the 4th Workshop on Social Network Systems (SNS'11)*. New York, NY, USA: ACM, pages 503–509.
- Cnet (2011). Nielsen: Twitter's growing really, really, really, really fast. http://news.cnet.com/8301-13577_3-10200161-36.html.
- Curino, C., Jones, E., and Madden, S. (2010). Schism : a workload-driven approach to database replication and partitioning. *Proceedings of the VLDB Endowment*, 3(1-2):48–57.
- Facebook (2011). Facebook statistics. <http://www.facebook.com/press/info.php?statistics>.
- Pujol, J., Erramilli, V., Siganos, G., Yang, X., Laoutaris, N., Chhabra, P., and Rodriguez, P. (2010). The little engine (s) that could: Scaling online social networks. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 375–386. ACM.
- Pujol, J., Siganos, G., Erramilli, V., and Rodriguez, P. (2009). Scaling online social networks without pains. In *Proc of NETDB*.
- Scalability, H. (2011). Friendster lost lead because of a failure to scale. <http://highscalability.com/blog/2007/11/13/friendster-lost-lead-because-of-a-failure-to-scale.html>.
- Viswanath, B., Mislove, A., Cha, M., and Gummadi, K. P. (2009). On the evolution of user interaction in Facebook. *Proceedings of the 2nd ACM workshop on Online social networks - WOSN '09*, page 37.
- Wikipedia (2011). List of social networking websites. http://en.wikipedia.org/wiki/List_of_social_networking_websites.

Usando informações do contexto para melhoria da precisão nas buscas por similaridade¹

Karine B. de Oliveira, Carina F. Dorneles

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

Campus Universitário Trindade, Florianópolis, SC, Brazil

{karineb,dorneles}@inf.ufsc.br

Abstract. *Most of the existing data sources have some form of representation for real-world objects classes, which can be represented in different ways. This article describes the proposal of a similarity technique that is suitable for searching for representations of real-world objects classes. A search approach is presented, based on the class context and attributes similarity. The main goal is to use any internal information from the document that can be representative to characterize the class itself and adjust the similarity function.*

Resumo. *A grande maioria das fontes de dados existentes possui alguma forma de representação de classes de objetos do mundo real, que podem ser representadas de maneiras distintas. Este artigo descreve a proposta de uma técnica de similaridade adequada para buscas por representações de classes de objetos do mundo real. É apresentado o projeto de uma abordagem de busca, que além da similaridade de atributos também é baseada no contexto onde a classe está inserida. A idéia principal é a utilização de qualquer informação interna ao documento que possa ser usada para caracterizar a classe propriamente dita e ajustar a função de similaridade através de informações do contexto.*

1. Introdução

A crescente possibilidade de publicação de conteúdo em diversas fontes de dados tem aumentado a quantidade de informações heterogêneas. As diferenças na representação podem aparecer nos valores (instâncias) dos dados armazenados, que representam entidades do mundo real, ou na sua estrutura (esquema) de armazenamento, que é o foco deste trabalho. A grande maioria das fontes de dados existentes, por exemplo, *web forms*, documentos XML, tabelas em um banco de dados relacional, *web tables*, entre outras, possui alguma forma de representação de classes de objetos do mundo real. Uma mesma classe de objetos do mundo real pode ser representada de maneiras distintas em diferentes fontes de dados.

Toda fonte de dados que possui alguma representação de classe pode também possuir outras informações além desta representação, que pode ser chamada de **contexto**. Conforme [Stefanidis et al. 2011], contexto é qualquer informação interna a um documento ou Banco de Dados, que pode ser usada para caracterizar o dado propriamente dito. No caso de *web forms*, por exemplo, que estão inseridos em uma

¹ Trabalho parcialmente financiado pelo CNPq - Bolsa PQ-Nível 2 (Nro. processo: 307992/2010-1) e do Universal - WF-Sim (Nro. processo: 481569/2010-3)

página web, sempre existem outras informações relevantes, além da estrutura do formulário. A Figura 1 apresenta o *web form* que está inserido na fonte de pesquisas *jobsearch.co.uk* representando a classe *job*. Nesta mesma fonte de dados, há também informações sobre propagandas de cursos para a carreira profissional, além de links para artigos e notícias sobre o mercado de trabalho. Essas informações podem ser entendidas como o contexto da classe e podem auxiliar na caracterização da mesma.

The screenshot shows the homepage of JobSearch.co.uk. At the top, there is a navigation bar with links: Home, Search, Recruiters A to Z, My Alerts, My Profile, My Jobs, Career Advice, Training, Help & FAQs, and Log in. Below the navigation bar, there is a banner for 'Commercial Solicitor' in Leeds Neg. To the right of the banner are three other job ads: 'Personal Injury Solicitor' in Leeds (£50,000 per year), 'Corporate Solicitor NO - 2 years PQE' in Leeds (£34,000 to £37,000), and another partially visible ad. Below the banner is a search form with fields for 'keyword(s)' (e.g. sales executive) and 'location' (e.g. London). There are also checkboxes for 'job term': any, full-time, part-time, permanent, contract/temporary, and a link to 'advanced search'. To the right of the search form are four promotional boxes: 'Job @Alerts', 'CV', 'news letters', and 'jobs of the week'. Below the search form, there is a summary: £836.3m worth of new jobs available now on site, 34,847 jobs from 951 employers, and 23,008 new jobseekers registered this month. A red arrow points to the search form area.

Figura 1: Representação de classe inserida no contexto

A diferença na representação de classes de objetos do mundo real é observada em muitas situações. Em formulários de pesquisa web, por exemplo, a classe “*job*” na fonte de pesquisa *jobsearchusa.org* (Figura 2) está representada com os seguintes atributos: “*state*”, “*city*”, “*industry*”, “*career field*” e “*job title*”. Na fonte de pesquisas *jobsearch.co.uk* (Figura 3) a classe “*job*” está representada com os seguintes atributos: “*keyword(s)*”, “*location*” e “*job term*”. Observa-se que comparando apenas duas fontes de dados, é possível identificar algumas diferenças nas representações da mesma classe de objetos do mundo real. A diferença na representação de classes pode tornar-se um problema quando se deseja buscar classes de objetos do mundo real, por exemplo, busca de *web forms* para pesquisas na *DeepWeb* [Madhavan et al. 2009] e *matching* de esquemas.

The screenshot shows the 'Quick Job Search' form on jobsearchusa.org. It consists of several dropdown menus labeled 'Step 1 - Select a State', 'Step 2 - Select a City', 'Step 3 - Select an Industry', and 'Career Field (Optional)'. Below these is a field for 'Job Title Search (Optional)' with a text input and a 'Find Jobs' button.

Figura 2: Web Form 1

The screenshot shows the 'Search Jobs' form on jobsearch.co.uk. It has fields for 'keyword(s)' (e.g. sales executive) and 'location' (e.g. London). Below these are checkboxes for 'job term': any, full-time, part-time, permanent, and contract/temporary, along with a link to 'advanced search'.

Figura 3: Web Form 2

A diferença de representações de dados similares em diversas fontes tem motivado o desenvolvimento de técnicas de recuperação dessas informações. Buscando maior abrangência dos resultados de pesquisas, alguns trabalhos propõem técnicas de

busca por similaridade [Levin e Heuser 2010, Borges e Dorneles 2006, Kaster et al. 2011]. Essas técnicas podem ser adequadas e aplicadas aos mais diversos contextos, por exemplo, desambiguação de entidades [Levin e Heuser 2010], busca por imagens [Kaster et al. 2011], ou até mesmo algoritmos de busca por similaridade em SGBDs (Sistemas de Gerenciamento de Bancos de Dados) comerciais [Borges e Dorneles 2006]. Um exemplo de similaridade de objetos complexos é Flint [Blanco et al. 2008], uma técnica de busca de entidades similares que coleta e indexa páginas web que contenham dados de instâncias de uma entidade conceitual. No entanto há uma carência no desenvolvimento de técnicas para busca de classes de objetos aplicadas a situações em que se tem apenas o conhecimento conceitual da classe e não dos objetos propriamente ditos. Nesta situação, técnicas de *matching* de esquemas não podem ser usadas, pois nem sempre as classes estão relacionadas com outras classes, como em *web forms*, por exemplo. Neste caso a classe está isolada, representada apenas pelo formulário.

A solução proposta neste trabalho para o problema de múltiplas representações de classes de objetos tem como objetivo o desenvolvimento de uma técnica de similaridade adequada para a busca por classes similares, usando informações de contexto. Visando alcançar este objetivo, é proposta uma abordagem de busca por similaridade baseada no contexto, ou seja, qualquer informação interna ao documento onde a classe está inserida que pode ser usada para caracterizar a classe propriamente dita [Stefanidis et al. 2011]. Além de utilizar a similaridade dos atributos identificados na estrutura da classe, também é utilizado o contexto da classe dentro da sua fonte de dados como forma de reajustar os escores de similaridade, de modo a diminuir a quantidade de falsos positivos e falsos negativos.

Os experimentos preliminares mostram que as classes analisadas apresentam contexto com conteúdo relevante, indicando que o reajuste dos escores de similaridade baseado no contexto pode atingir resultados mais precisos para as buscas. Desta forma as principais contribuições do trabalho são: a) a proposta de um método de busca por similaridade específico para pesquisas de representações de classes de objetos do mundo real; b) a utilização de informações do contexto no ajuste dos escores de similaridade entre as classes pesquisadas.

Este artigo é organizado como segue. A Seção 2 apresenta alguns trabalhos relacionados. A Seção 3 apresenta alguns conceitos de processo de matching, funções de similaridade e contexto. A Seção 4 apresenta a proposta. A Seção 5 apresenta os experimentos preliminares. Na Seção 6 são apresentados os desafios futuros e as considerações finais.

2. Trabalhos Relacionados

Alguns trabalhos têm sido desenvolvidos utilizando similaridade entre objetos complexos armazenados nas mais diversas formas na web [Elmagarmid et al. 2007]. Neste contexto, um dos problemas que envolve similaridade de objetos complexos é a resolução de entidades, processo que identifica registros que referem-se à mesma entidade do mundo real e que combina as diferentes representações em uma única [Wang et al. 2011, Whang et al. 2009, Ploch 2011]. Outro problema que envolve similaridade de objetos complexos é a busca por objetos ou instâncias da mesma entidade do mundo real [Blanco et al. 2008, Kantorski e Guimarães 2010].

Em [Wang et al. 2011], a resolução de entidades é feita baseando-se em uma tabela com um conjunto de registros e em regras de casamento de registros com funções

de similaridade e *thresholds* desconhecidos, o objetivo é encontrar as melhores funções de similaridade e *thresholds*, para combinar as entidades. Como entrada, o usuário informa exemplos positivos (ex.: registros que representam a mesma entidade) e exemplos negativos (ex.: registros que não representam a mesma entidade). Esses exemplos são usados para identificar as melhores funções de similaridade, utilizando-as para eliminar redundâncias.

Outra abordagem utilizada na solução do problema da resolução de entidades é a de pares genéricos [Whang et al. 2009]. Neste caso, para determinar se dois registros representam a mesma entidade do mundo real, são usadas duas funções, uma de *match* e outra de *merge*. Um especialista do domínio escreve essas funções, que também podem ser desenvolvidas através de técnicas de aprendizado de máquina. A função de *match* avalia se um par de registros representa a mesma entidade do mundo real. Se a função *match* retornar uma resposta verdadeira, então a função de *merge* é usada para criar um registro composto pelos dois anteriormente analisados. Depois de combinados os registros, a função de *match* pode ser aplicada novamente a fim de comparar com um terceiro registro, por exemplo.

Considerando a complexidade do problema da resolução de entidades, [Ploch 2011] explora os relacionamentos entre entidades. O processo envolve a construção de uma base de conhecimento a partir das informações coletadas dos artigos da Wikipedia, como por exemplo, título e links da página, de onde são extraídos e armazenados nomes de entidades. A base de conhecimento serve para rotular corretamente as formas diferentes de se referir à mesma entidade e também formas iguais de se referir a entidades diferentes da Wikipedia. Depois de identificadas as formas de referência a entidades dentro de um artigo. Através de uma estrutura de grafo criada com base nos links de cada página, é possível identificar a quais entidades elas referem-se, auxiliando na desambiguação.

O foco do Flint [Blanco et al. 2008], além da resolução de entidades, é a busca de entidades. O objetivo é pesquisar, coletar e indexar automaticamente páginas web que contenham representações de instâncias de uma determinada entidade do mundo real. A busca é feita através da entrada de algumas páginas web rotuladas, que são usadas como sementes, de onde o sistema infere a descrição conceitual da entidade e pesquisa por outras páginas contendo dados que representem instâncias dessa entidade.

A solução proposta difere das soluções apresentadas pois propõe a utilização do contexto para fazer o ajuste de escores e tornar o processo mais exato. A solução proposta também pode ser usada para melhorar processos de *matching* de esquemas que não levam em consideração o contexto das representações de classes para o seu cálculo de similaridade. Existem diferentes propostas para *matching* de esquemas [Shvaiko e Euzenat 2005] em que o objetivo é bastante semelhante ao objetivo desta proposta, no entanto algumas delas apresentam soluções que utilizam características inerentes a um esquema completo, como a utilização dos relacionamentos entre classes, o que não existe em *web forms* por exemplo, onde a representação da classe aparece sozinha. Em contrapartida, na maioria dos casos de *matching* de esquemas existe alguma informação que pode ser considerada o contexto da classe, como por exemplo, as representações de classes relacionadas em um esquema.

3. Conceitos básicos

A seguir serão apresentados alguns conceitos importantes para a compreensão da proposta: **processo de matching, funções de similaridade e contexto**.

Processo de matching: o processo de casamento de dados visa definir se dois ou mais dados representam o mesmo objeto do mundo real [Dorneles et al. 2011]. O casamento de dados é utilizado em integração de dados, consultas aproximadas, pesquisa por similaridade, ou seja, sempre que se pretende analisar dados provenientes de uma ou várias fontes de dados para determinar se representam ou não o mesmo objeto do mundo real. O processo de casamento de dados nem sempre é trivial, visto que os dados podem apresentar heterogeneidade de representação, tanto na estrutura quanto em seus valores. Portanto esse processo deve ser capaz de analisar estrutura e valor para atingir com maior precisão seus objetivos [Dorneles et al. 2011].

Funções de similaridade: quando os dados analisados pelo processo de *matching* são atômicos, como por exemplo (strings, valores numéricos, datas, etc.) o problema pode ser resolvido através do uso de uma função de similaridade apropriada para o tipo de dado. Em casos onde os valores são complexos, como tuplas em bancos de dados relacionais, *web forms* e documentos XML, podem ser adotadas abordagens que combinam as funções de similaridade para dados atômicos, com outras técnicas mais sofisticadas, como aprendizado de máquina por exemplo [Dorneles et al. 2011].

Existem várias funções de similaridade para strings, como Levenshtein [Levenshtein 1966], Jaro-Winkler [Jaro 1976], N-Grams [Elgaramid et al. 2007], entre outras. As diferentes funções de similaridade resultam valores diferentes para as mesmas entradas. A escolha do melhor método vai depender do domínio em que o cálculo de similaridade será aplicado [Levin e Heuser 2010].

Contexto: de maneira geral, o “contexto” pode ser entendido como um conhecimento que ajuda a identificar o que é ou não relevante em um dado momento e lugar. O contexto em sistemas é usado para fornecer serviços ou informações mais relevantes para os usuários na realização das suas tarefas. Segundo [Dey 2001], contexto é qualquer informação que caracteriza a situação de uma entidade (pessoa, lugar ou objeto) considerada relevante para a interação entre uma pessoa e uma aplicação.

Trazendo este conceito para Bancos de Dados, contexto é qualquer informação externa ao Banco de Dados que pode ser usada para caracterizar a situação de um usuário ou qualquer informação interna ao Banco de Dados que pode ser usada para caracterizar o dado propriamente dito [Stefanidis et al. 2011]. O contexto em Banco de Dados pode ser usado tanto para personalizar informações armazenadas, quanto para recuperar informações mais precisas para as consultas.

4. ContextSim

Esta seção apresenta uma visão geral da proposta, o detalhamento do motor de buscas e o ajuste dos escores de similaridade baseado no contexto.

4.1 Visão Geral

A Figura 4 apresenta a arquitetura do ContextSim (*Context Similarity*). Para que seja feito o processamento das consultas informadas pelo usuário, o conjunto de documentos consultados deve passar por uma etapa de indexação. Nesta etapa o conjunto de dados é preparado através de índices para facilitar as consultas pelo motor de buscas. Na etapa de processamento de consultas, os índices preparados anteriormente são consultados, e

os resultados são retornados ao usuário em forma de *ranking* pelo escore ajustado de similaridade com a classe pesquisada.

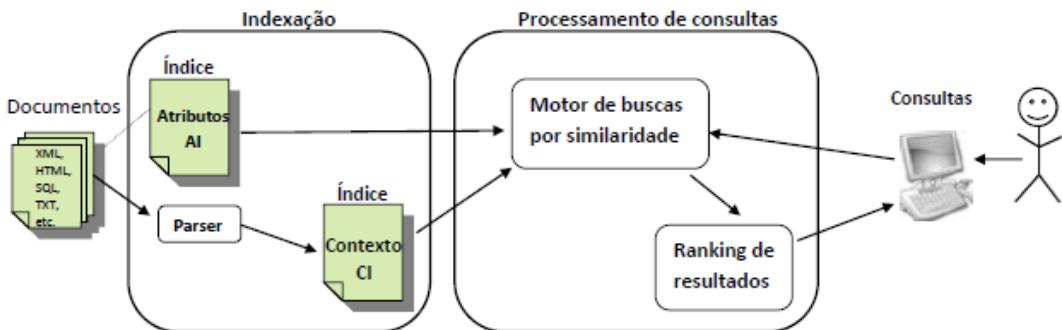


Figura 4: Visão geral do ContextSim

Na fase de indexação do conjunto de documentos, são construídas duas estruturas de índices: índice de atributos (AI) e índice de contexto (CI). O primeiro armazena os atributos das classes contidas nos documentos. Esta tarefa não está no escopo deste trabalho e pode ser realizada manualmente ou com a utilização de ferramentas para extração automática de atributos. No entanto, deve ser seguida uma estrutura específica para o índice, criando os campos referentes ao *identificador do documento*, *identificador do atributo* e *nome do atributo*.

O CI é criado com o auxílio da ferramenta *Lucene*² e armazena as informações contextuais da classe dentro do documento, exceto aquelas que dizem respeito à estrutura da classe propriamente dita (ou seja, o índice de contexto armazena termos importantes do documento, exceto atributos da classe, já indexados). Para a criação deste índice são percorridos todos os documentos e retiradas as informações da classe, como seus atributos, além de *stop words* e *tags HTML*, que são consideradas irrelevantes para o contexto. As demais informações são indexadas com a seguinte estrutura: escore, campo criado automaticamente pela ferramenta² que indica a relevância de um determinado termo para cada documento onde ele aparece, *identificador do documento*, *arquivo*, que neste caso optou-se pelo caminho absoluto do arquivo e *conteúdo*, que armazena o termo propriamente dito.

4.2 Busca por similaridade

Na etapa de processamento de consultas, o usuário informa como consulta a estrutura de uma classe, com nome e atributos. O motor de busca, por sua vez, utiliza os índices criados para pesquisar por classes similares à informada pelo usuário. A Figura 5 mostra o detalhamento do motor de buscas. Inicialmente é feito o cálculo de similaridade entre atributos, resultando em um *ranking* com o escore de cada classe do índice com a classe pesquisada. Para fazer o ajuste de escore, aumentando o escore de falsos negativos e diminuindo o escore de falsos positivos é utilizado o índice de contexto. Ao final do processo, é feito um *ranking* com os novos valores de escore ajustados.

² <http://lucene.apache.org/java/docs/index.html>

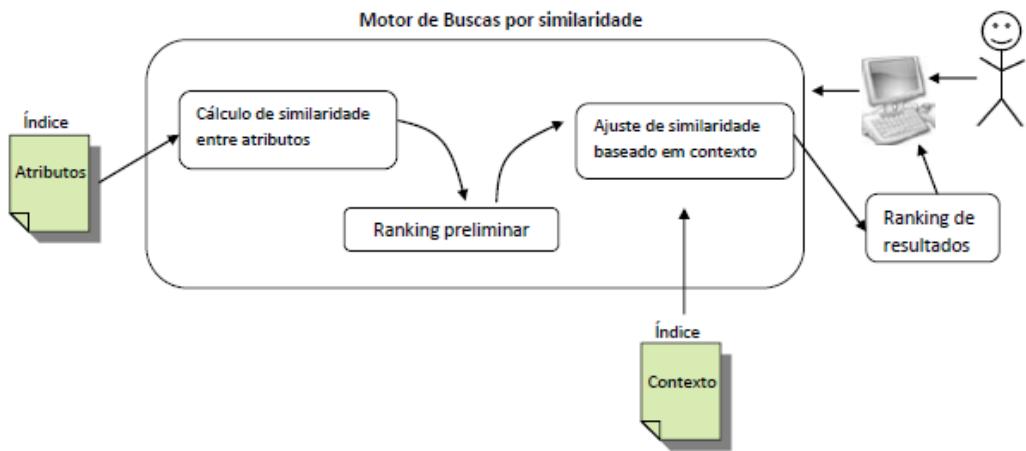


Figura 5: Detalhamento do motor de buscas por similaridade

O cálculo de similaridade entre atributos é feito comparando cada um dos atributos da classe pesquisada com cada um dos atributos das classes armazenadas através da função N-Grams [Elgaramid et al. 2007]. Caso o escore de similaridade seja inferior a 1.0, o atributo é comparado com seus sinônimos através da ferramenta WordNet⁴. A Figura 6 mostra a fórmula do cálculo da similaridade geral dos atributos para cada classe, onde n representa o número de atributos da classe pesquisada, S representa o maior escore de similaridade entre o k -ésimo atributo da classe pesquisada em comparação com os atributos da classe comparada. N representa o maior número de atributos, que pode ser, ou da classe pesquisada ou da classe comparada. Esta fórmula dá origem a um escore de similaridade de atributos para cada uma das classes presentes no índice, esses escores são organizados em um *ranking* em ordem decrescente.

$$sim(C_i, C_j) = \frac{\sum_{k=1}^n S_{k-1}}{N}$$

Figura 6: Fórmula para o cálculo de similaridade entre atributos

Considerando que o *ranking* resultante do escore de similaridade entre atributos pode resultar em falsos positivos e falsos negativos, a proposta é fazer o ajuste desses valores utilizando o contexto. Este ajuste é feito baseado na idéia de [Silva et al. 2007], porém com propósitos diferentes. O nome da classe ou do domínio é pesquisado dentro do próprio índice de contexto junto com cada termo do contexto indexado para determinada classe, quanto maior o número de resultados obtidos para uma pesquisa, maior a relação entre o termo e a classe, ou seja, eles aparecem com maior frequência juntos.

4.3 Ajuste de similaridade baseado em contexto

A partir da avaliação dos resultados das buscas dentro do índice de contexto, propõe-se o estabelecimento de pesos para os termos do contexto em relação à cada classe. Para isso cria-se uma estrutura de grafo que permite avaliar os termos mais relevantes no contexto de cada classe, podendo ajustar o escore de similaridade da classe de acordo com os seus termos do contexto.

Na abordagem proposta para a criação do grafo, as arestas ligam as classes aos termos de seu contexto e os termos às demais classes onde aparecem. Para cada aresta é

⁴ <http://wordnet.princeton.edu/>

estabelecido um peso do termo em relação ao documento onde a classe à qual se relaciona aparece. Este peso é obtido pesquisando-se no índice de contexto o termo junto com o nome da classe à qual se relaciona. Por exemplo, utilizando como palavras-chave para a busca, *book* (classe) e *history* (termo) foram obtidos 88 resultados, o que determina o peso da aresta entre esses dois nodos. A Figura 7 mostra um exemplo de parte da estrutura de grafo que poderia ser criada com termos do contexto onde as classes estão representadas pelos nodos preenchidos e os termos pelos nodos sem preenchimento. É importante ressaltar que para cada aresta, que tem dois nodos (origem e destino), um nodo representa a classe e outro nodo representa um termo do contexto, ou seja, duas classes, ou dois termos, nunca estão relacionados.

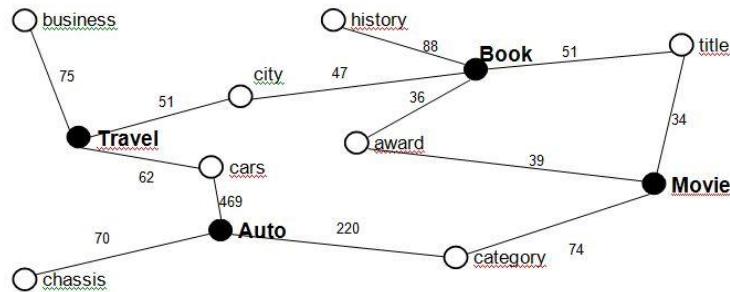


Figura 7: Grafo de contexto

Os pesos das arestas podem ser avaliados em relação a outras classes onde também aparecem e a partir desses valores, o escore geral de similaridade pode ser ajustado para mais ou para menos, dependendo dos termos que aparecem em seu contexto e no quanto esses termos são representativos.

5. Experimentos preliminares

Alguns experimentos preliminares com *web forms* foram realizados com os seguintes objetivos: I) Avaliar a qualidade dos termos indexados no CI; II) estimar a empregabilidade destes termos no ajuste dos escores de similaridade. Os experimentos validam a primeira etapa da ferramenta, a indexação. Foram indexadas 3200 páginas web que contém *web forms* utilizando o índice de atributos da base de dados do sistema *DeepPeep* [Barbosa et al. 2010] e criando o índice de contexto de cada uma. A seguir foram realizadas pesquisas com atributos correspondentes a classes de quatro domínios diferentes: *book*, *travel*, *auto* e *movie*. Para cada pesquisa realizada, retornou-se o *ranking* com o escore de similaridade de atributos e o número de resultados das pesquisas do nome do domínio junto com cada termo no próprio índice de contexto. A partir dos resultados destas buscas, é possível inferir que aqueles termos que aparecem mais vezes junto com o nome do domínio podem ser considerados pertencentes ao contexto da classe que se está pesquisando.

De acordo com os experimentos feitos até o momento, este método de avaliação do contexto mostrou-se eficaz. Por exemplo, para a pesquisa (*book*, *history*), sendo *book* o nome da classe, são obtidos 88 resultados, já para a pesquisa (*book*, *migration*) é obtido apenas 1 resultado, indicando que o termo *history* está mais relacionado à classe *book* do que o termo *migration*. Além da pesquisa com o nome da classe também pretende-se fazer pesquisas com alguns de seus sinônimos para ampliar a avaliação do contexto.

A Tabela 1 mostra os resultados obtidos a partir das buscas realizadas. Foi feita uma busca para cada domínio e selecionados os três resultados com maior escore em similaridade de atributos para a pesquisa de contexto. A média de termos pesquisados é o valor médio de termos indexados para cada uma das três classes selecionadas. A média de termos do contexto, é a média dos termos que obtiveram mais resultados em buscas no índice junto do nome da classe pesquisada e são considerados mais significativos. Também é apresentado o total de termos do contexto em percentual. Para que o percentual de termos significativos atinja valores maiores é necessário o refinamento do processo de indexação do contexto.

Tabela 1: Quantidade de termos contextuais obtidos nos experimentos

Domínio	Média de termos pesquisados	Média de termos do contexto	Média de termos do contexto (%)
Book	565,33	321,3	56,83 %
Travel	378	208	55,03 %
Auto	265,75	152,88	57,52 %
Movie	157,5	96	60,95 %

6. Considerações finais

Este artigo descreve a proposta de uma abordagem de busca por similaridade de representações de classes de objetos do mundo real que emprega o conceito de contexto da classe. O processo proposto para as buscas está dividido em duas etapas: a) indexação, em que o ambiente é preparado para as buscas através da criação dos índices de atributos e de contexto; b) processamento de consultas, etapa em que as consultas do usuário devem ser processadas pelo motor de buscas a partir da estrutura de índices criada na etapa de pré-processamento.

Alguns experimentos preliminares com o índice de contexto mostram que os documentos onde as classes estão inseridas realmente possuem um número significativo de termos que são representativos do seu contexto. A partir dos resultados preliminares pode-se concluir que fazendo a utilização dos termos do contexto é possível fazer o ajuste dos escores de similaridade entre atributos visando maior precisão para as buscas e diminuindo falsos positivos e falsos negativos.

Como desafios futuros, está o refinamento da proposta para a implementação do protótipo de buscas. Além disso, pretende-se validar da proposta através da realização de um conjunto maior de experimentos para validar o impacto da utilização do contexto na redução de falsos positivos e falsos negativos nas buscas por classes similares

Referências

- Barbosa, L., Nguyen, H., Nguyen, T., Pinnamaneni, R. e Freire, J. (2010). Creating and exploring *web form* repositories. In: SIGMOD'10, Indianapolis, Indiana, USA.
- Blanco, L., Crescenzi, V., Merialdo, P. e Papotti, P. (2008). Flint: Google-Basing the Web. In: EDBT - International Conference on Extending Database Technology.

- Borges, E., N. e Dorneles, C., F. (2006). PgSimilar: Uma ferramenta open source para suporte a consultas por similaridade no PostgreSQL. In: Anais da III Sessão de Demos, em conjunto com o XXI SBBD, Florianópolis, p. 1-6.
- Dey, A. (2001). Understanding and Using Context. In: Personal and Ubiquitous Computing, Vol. 5, p. 4-7, Springer London.
- Dorneles, C., F., Gonçalvez, R. e Mello, R., S. (2011). Approximate data instance matching: a survey. In: Knowledge and Information Systems, Volume 27.
- Elgaramid, A., K., Ipeirotis, P., G. e Verykios, V., S. (2007) Duplicate Record Detection: A Survey. IEEE Transactions on Knowledge and Data Engineering, [S.l.], Vol. 19, p. 1-16.
- Elmagarmid, A., K., Ipeirotis, P., G. e Verykios V. S. (2007). Duplicate Record Detection: A Survey. In. The IEEE Transations on knowledge and Data Engineering (TKDE) Vol. 19 No. 1, pp. 1-16.
- Jaro, M., A. (1976). Unimatch: A Record Linkage System: User's Manual. Technical report, US Bureau of the Census, Washington, D.C., 1976.
- Kantorski, G., Z. e Guimarães, R. (2010). Similaridade entre Objetos Localizados em Fontes de Dados Heterogêneas. In: VI ERBD, Joinville-SC.
- Kaster, D., S., Bugatti, P., H., Ponciano-Silva, M., Traina, A. J. M., Azevedo-Marques, P., M., Santos, A., C. e Traina Jr., C. (2011). MedFMI-SiR: A Powerful DBMS Solution for Large-Scale Medical Image Retrieval. In: ITBAM, 16-30
- Levenshtein, V., I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady, Vol. 10, p.707, URSS.
- Levin, F. e Heuser, A., C. (2010). Using Genetic Programming to Evaluate the Impact of Social Network Analysis in Author Name Disambiguation. In AMW.
- Madhavan, J., Afanasiev, L., Antova, L. e Halevy, A. (2009). Harnessing the Deep Web: Present and Future. In 4th Biennial Conf. on Innovative Data Systems Research (CIDR).
- Ploch, D. (2011) Exploring Entity Relations for Named Entity Disambiguation. In: ACL-HLT. Student Session, pages 18–23,Portland, OR, USA 19-24
- Shvaiko, P. e Euzenat, J. (2005). A Survey of Schema-based Matching Approaches. In: Journal on Data Semantics, Vol. 3730, p.146-171.
- Silva, A., S., Barbosa, D., Cavalcanti, J., M., B. e Sevalho, M., A., S. (2007). Labeling Data Extracted from the Web. In: OTM, Vol. Part I.
- Stefanidis, K., Koutrika, G. e Pitoura, E. (2011). A Survey on Representation, Composition and Application of Preferences in Database Systems. In: ACM Transactions on Database Systems. Vol. 36, 3, 45 p.
- Wang, J., Li, G., Yu, J. e Feng, J. (2011) Entity Matching: How Similar Is Similar. In: VLDB, Vol. 4, No. 10.
- Whang, S., E., Benjelloun O. e Garcia-Molina, H. (2009). Generic entity resolution with negative rules. In: VLDB, Vol. 18, No. 6, p1261-1277