Uso de Inferência na Compreensão das Modificações em Documentos Semiestruturados

Alessandreia Marta de Oliveira[1, 2], Leonardo Murta[1], Vanessa Braganholo[1]

[1]Universidade Federal Fluminense
[2] Universidade Federal de Juiz de Fora
{alessandreia, leomurta, vanessa}@ic.uff.br

Abstract. Applications nowadays are increasingly using XML to represent semistructured data and, consequently, a large amount of XML documents is available worldwide. As semistructured data evolve over time, due to, for example, some changes of technical nature, change control over XML documents becomes fundamental. Existing research on XML change control focuses on discovering syntactic changes. However, there are several applications where this syntactic information is not enough, ie, situations where it is necessary to detect the elements or attributes that have changed and it is also necessary to infer the changes reason. To solve this problem, this paper presents an inference-based XML evolution approach using Prolog that is able to detect semantic changes between two versions of an XML document. We also present an example considering an employee management system to illustrate the approach, showing the possibility of inferring the user's intent when creating the second version of a document.

Resumo. As aplicações se apoiam cada vez mais na linguagem XML para representar dados semiestruturados e, consequentemente, uma grande quantidade de documentos XML é gerada. Um problema relacionado é que os dados semiestruturados evoluem ao longo do tempo, em função, por exemplo, de modificações de cunho técnico. Desta forma, a gerência das modificações em documentos XML é uma necessidade cada vez mais crescente. Abordagens existentes relacionadas ao controle de mudanças de documentos XML têm seu foco em mudanças sintáticas. No entanto, existem diversas aplicações onde essa informação sintática não é suficiente, ou seja, situações onde não basta detectar os elementos ou atributos que mudaram, mas também é necessário inferir a razão das modificações. Diante disso, este artigo apresenta uma abordagem para a compreensão da evolução de documentos XML baseada em inferência, utilizando a linguagem Prolog, que é capaz de detectar mudanças semânticas entre duas versões de um documento XML. Um cenário considerando um sistema de informações de funcionários exemplifica a proposta e mostra a possibilidade de inferir a intenção do usuário ao criar a segunda versão de um documento.

Categories and Subject Descriptors: H. Information Systems [H.2. Database Management]: Textual Databases

Palavras Chave: Controle de modificações, inferência, Prolog, XML

1. INTRODUÇÃO

O volume e a diversidade de informações que circulam atualmente pela Web têm crescido muito. Muitas dessas informações encontram-se organizadas em documentos XML [Bray *et al.* 2008] e provêm de várias fontes. Um problema relacionado é que tais documentos evoluem ao longo do tempo, em função, por exemplo, de modificações de cunho técnico. Além disto, os usuários podem não se interessar somente pelos valores atuais dos documentos, mas também pelo monitoramento e controle das modificações entre as suas versões. O usuário pode querer monitorar modificações ou precisar de informações de versões antigas de um documento. Por exemplo, num cenário onde um documento XML representa o cadastro de funcionários de uma empresa, o usuário pode querer verificar quando determinados funcionários foram contratados ou demitidos, ou então verificar o cargo de um funcionário num dado momento no passado.

Desta forma, percebe-se que o controle das modificações em documentos semiestruturados, em especial documentos XML, é uma necessidade cada vez mais crescente. Na literatura, estes problemas vêm sendo estudados já há algum tempo [Cobena *et al.* 2002; Wang *et al.* 2003; Zhao *et al.* 2004]. No

entanto, o foco destas abordagens está relacionado somente às modificações sintáticas nos documentos, ou seja, estes algoritmos realizam comparações entre os documentos baseadas nas suas estruturas, sem considerar a semântica associada. Sendo assim, no cenário do exemplo de cadastro de funcionários, essas abordagens conseguem detectar que o valor do salário de um funcionário mudou, mas somente com essa informação não é possível saber se ele teve um aumento anual ou se foi promovido, por exemplo.

Diante disso, este artigo apresenta uma abordagem que visa apoiar a compreensão da evolução de documentos XML. Essa proposta faz uso de um mecanismo de inferência, baseado na linguagem Prolog, para prover semântica na comparação entre duas versões de um documento XML. O contexto esperado de aplicação da abordagem proposta não assume o uso de rótulos temporais, como proposto em Chien *et al.* [2001] e Rizzolo e Vaisman [2008], por exemplo. Desta forma, esta proposta atua em cenários mais críticos, onde são considerados como entrada somente duas versões de um documento XML, sem nenhum tipo de metadado relacionando essas versões. A abordagem consiste na compreensão das modificações entre versões de um documento XML, que são transformadas em fatos Prolog e que, a partir de um conjunto de regras de inferência, possibilita deduzir a intenção do usuário ao criar a segunda versão. Em suma, o propósito desse trabalho é viabilizar a identificação da razão real das modificações em documentos XML, tomando como base a análise das modificações sintáticas granulares em atributos e elementos.

O restante do artigo está organizado em outras cinco seções. A Seção 2 apresenta um exemplo motivacional para fundamentar a abordagem proposta na Seção 3. A Seção 4 apresenta um exemplo de utilização e a Seção 5 descreve alguns trabalhos relacionados. Para finalizar, a Seção 6 conclui o artigo e descreve algumas propostas de trabalhos futuros.

2. EXEMPLO MOTIVACIONAL

Esta seção apresenta o exemplo utilizado ao longo do artigo com o objetivo de ilustrar a abordagem proposta. Suponha um sistema que gerencia o cadastro de funcionários de uma empresa, lidando com informações tais como CPF, nome, salário, cargo, departamento ao qual está vinculado e filial onde está alocado. Essas informações foram descritas em um documento XML, versão 1 (v1), onde foram definidos alguns funcionários, como, por exemplo, João, que é analista de sistemas da empresa (Figura 1.a). Suponha ainda que, posteriormente, João teve seu salário e cargo alterados e o funcionário Pedro mudou de filial, como mostra o trecho em destaque na Figura 1.b.

Tais modificações textuais são facilmente identificadas após uma análise das duas versões deste fragmento de documento XML. No caso de uma empresa com um número de funcionários considerável, esta tarefa já não seria trivial. Para tanto, existem algoritmos genéricos para detectar as modificações em arquivos texto, sem suporte para a sintaxe específica do arquivo, como, por exemplo, diff3 [MacKenzie *et al.* 2011]. O problema é que eles não consideram o formato específico dos arquivos XML, tratando-os como arquivos comuns de texto. Desta forma, caso uma modificação ocorra, por exemplo, na ordem de dois atributos, tais algoritmos acusariam que as duas versões do documento são diferentes. No entanto, em se tratando do modelo XML, a ordem dos atributos é irrelevante.

Para controlar versões de documentos XML, é necessária uma forma de detectar exatamente quais são as diferenças entre duas versões do documento, ou seja, o que foi modificado de uma versão para a outra em termos das estruturas que compõem um documento XML (elementos e atributos). Existem alguns algoritmos dedicados à detecção de diferenças em documentos XML, tais como X-Diff [Wang *et al.* 2003] e XyDiff [Cobena *et al.* 2002]. No entanto, apesar destes algoritmos levarem em consideração a estrutura do documento, sendo assim mais adequados que os algoritmos genéricos, eles se prendem somente à sintaxe do documento XML, não possibilitando a identificação da semântica das modificações (i.e., razão por trás das ações de adição, remoção e alteração dos elementos e atributos do documento). Existem diversas aplicações onde essa informação sintática não é suficiente, ou seja, situações onde não

basta detectar os elementos ou atributos que mudaram, mas também é necessário inferir a razão das modificações. Neste caso, por exemplo, após uma análise do novo cenário (v2), é importante perceber que João recebeu um aumento de salário e, além disso, mudou de cargo, o que significa que ele foi promovido. Em outras palavras, um conjunto de modificações na estrutura do documento pode corresponder a uma modificação semântica, que deveria ser automaticamente identificada.

Documento XML versão 1

```
<?xml version="1.0" encoding="UTF-8"?>
<empresa>
  <funcionario>
    <cpf>12365444410</cpf>
    <nome>Joao</nome>
    <salario>1600</salario>
    <cargo>Analista de Sistemas</cargo>
    <depto>Tecnologia da Informacao</depto>
    <filial>Juiz de Fora</filial>
  </funcionario>
  <funcionario>
    <cpf>56798012314</cpf>
    <nome>Pedro</nome>
    <salario>1900</salario>
    <cargo>Suporte Help Desk</cargo>
    <depto>Tecnologia da Informacao</depto>
    <filial>Juiz de Fora</filial>
  </funcionario>
  ...
</empresa>
```

(a) versão 1

Documento XML versão 2

```
<?xml version="1.0" encoding="UTF-8"?>
<empresa>
  <funcionario>
    <cpf>12365444410</cpf>
    <nome>Joao</nome>
    <salario>2500</salario>
    <cargo>Analista de Sistemas Pleno</cargo>
    <depto>Tecnologia da Informacao</depto>
    <filial>Juiz de Fora</filial>
  </funcionario>
  <funcionario>
    <cpf>56798012314</cpf>
    <nome>Pedro</nome>
    <salario>1900</salario>
    <cargo>Suporte Help Desk</cargo>
    <depto>Tecnologia da Informacao</depto>
    <filial>Sao Paulo</filial>
  </funcionario>
...
</empresa>
```

(b) versão 2

Figura 1: Fragmento de duas versões de um documento XML

Em suma, em documentos XML grandes, que passaram por muitas modificações entre suas versões, pode haver uma redução significativa no número de modificações a serem apresentadas ao usuário ao migrar de modificações sintáticas para modificações semânticas. No exemplo desta seção, duas modificações sintáticas (aumento de salário e mudança de cargo) puderam ser sumarizadas em uma modificação semântica (promoção). Contudo, essa relação pode ser ainda maior, permitindo que centenas de modificações sintáticas sejam transformadas em dezenas de modificações semânticas. Essa característica tem potencial para contribuir com a facilidade de análise e compreensão sobre a evolução de documentos XML.

3. ABORDAGEM PROPOSTA

A abordagem proposta neste trabalho processa os dados contidos em duas versões de um documento XML com o objetivo de compreender a razão das modificações, possibilitando a obtenção de conhecimento semântico a partir de informações explícitas nas versões do documento ou deduzidas automaticamente através de regras (informações implícitas).

A Figura 2 apresenta a abordagem proposta, onde a entrada é composta por duas versões de um documento XML, que são pré-processadas (convertidas para fatos Prolog). Estes fatos são usados como entrada na máquina de inferência que, a partir das regras manuais (definidas a critério de um usuário especialista do domínio), resultam na razão das modificações do documento, retratando a sua evolução. Desta forma, a identificação da razão ou semântica de uma modificação é a principal contribuição deste trabalho.

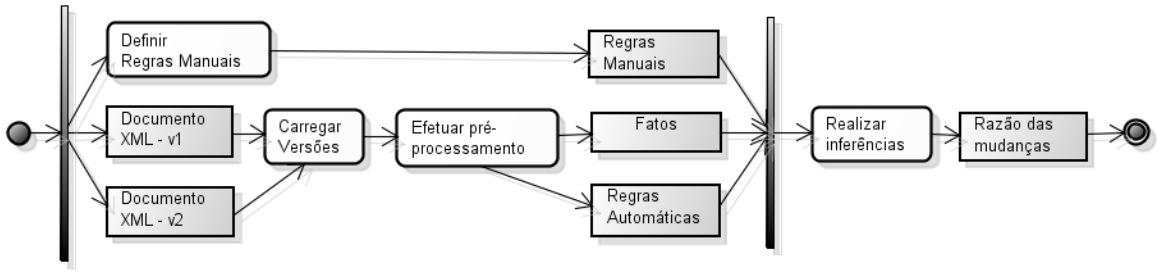

Figura 2: Abordagem proposta

Para viabilizar a realização das inferências é necessário traduzir os dados contidos no documento XML para uma linguagem que forneça esta capacidade, tal como Datalog [Huang *et al.* 2011], RuleML [Boley 2003] ou Prolog [Bratko 2001]. Datalog é uma linguagem de consulta não procedural baseada em Prolog que surgiu da combinação de programação em lógica com bancos de dados. O fato de Datalog não permitir termos complexos como argumento do predicado e possuir restrições no uso de negação e recursividade, tornou-a incompatível com a abordagem proposta. A linguagem RuleML é o resultado de um esforço para fornecer um padrão de definição de regras na Web e descreve tanto as informações como os seus relacionamentos, tornando possível a realização de inferência. O fato de a RuleML ser uma linguagem de marcação voltada para representar regras de inferência com foco na Web e suas implementações atuais utilizarem uma máquina Datalog como mecanismo de inferência, foram motivos para que seu uso fosse descartado nesta proposta. Consequentemente optou-se por Prolog para estabelecer relações a partir de documentos XML. De fato, a linguagem Prolog já foi utilizada com sucesso para realizar consultas com inferência a documentos XML [Lima *et al.* 2012].

Assim, o pré-processamento dos documentos XML de entrada (tradução) é efetuado pelo método proposto por Lima *et al* [2012], onde o processo de tradução gera vários fatos Prolog a partir de um documento XML, transformando os elementos em predicados e seus conteúdos em constantes. Uma adaptação deste método, utilizado apenas na fase de pré-processamento, foi proposta para gerar um único conjunto de fatos baseado em informações de duas versões de um documento XML, com uma identificação da raiz do documento XML associada à versão (v1 ou v2, por exemplo). Desta forma, é possível comparar as informações dos arquivos XML utilizando para tanto este único conjunto de fatos.

Por exemplo, as duas versões do documento XML apresentadas na Seção 2, após serem convertidas para Prolog, geram os fatos apresentados na Figura 3. Para relacionar predicados distintos, um identificador (uma constante Prolog) é acrescentado como parâmetro, caracterizando que há correspondência entre esses dados no documento XML. Como exemplo, pode-se mencionar a relação de pai/filho existente entre a empresa (empresa (id1)) e um de seus funcionários (funcionário (id1, id2)) destacada na Figura 3.a.

Analisando a definição dos predicados, um especialista (com conhecimento no domínio e em Prolog) pode criar regras manuais que permitam inferir informações relevantes. Um exemplo de regra manual é apresentado na Figura 4.a, onde pode ser verificado, no mesmo contexto do exemplo motivacional da Seção 2, se um determinado funcionário recebeu aumento. Isso é considerado verdade quando o conteúdo do elemento salário da versão 1 do documento XML é menor do que na versão 2 e a pessoa se manteve no mesmo cargo. Se o funcionário, além de ter recebido um aumento de salário, também mudou de cargo, isto implica em uma promoção (Figura 4.b).

Para facilitar a escrita das regras manuais foram criadas algumas regras adicionais que garantem, por exemplo, que ao verificar se um funcionário recebeu aumento, a comparação seja feita realmente entre o mesmo funcionário nas duas versões do documento XML, usando para isso o atributo chave (cpf). A Figura 5 mostra a regra *mesmo_func*, que identifica os funcionários unicamente em v1 e v2.

Documento XML versão 1 convertido em fatos

```
empresa(id1).
funcionario(id1, id2).
cpf(id2, '12365444410').
nome(id2, 'joao').
salario(id2, 1600).
cargo(id2, 'analista de sistemas').
depto(id2, 'tecnologia da informacao').
filial(id2, 'juiz de fora').
funcionario(id1, id3).
cpf(id3, '56798012314').
nome(id3, 'pedro').
salario(id3, 1900).
cargo(id3, 'suporte help desk').
depto(id3, 'tecnologia da informacao').
filial(id3, 'juiz de fora').
...
```

(a) v1 – fatos

Documento XML versão 2 convertido em fatos

```
empresa(id8).
funcionario(id8, id9).
cpf(id9, '12365444410').
nome(id9, 'joao').
salario(id9, 2500).
cargo(id9, 'analista de sistemas pleno').
depto(id9, 'tecnologia da informacao').
filial(id9, 'juiz de fora').
funcionario(id8, id10).
cpf(id10, '56798012314').
nome(id10, 'pedro').
salario(id10, 1900).
cargo(id10, 'suporte help desk').
depto(id10, 'tecnologia da informacao').
filial(id10, 'sao paulo').
...
```

(b) v2 – fatos

Figura 3: Fragmentos convertidos em fatos

```
recebeu_aumento(Nome) :-
   mesmo_func(Fb,Fm),
   salario(Fb,Sb),salario(Fm,Sm),Sb<Sm,
   cargo(Fb,C),cargo(Fm,C),
   nome(Fm,Nome).
```

(a) Regra recebeu_aumento

```
foi_promovido(Nome) :-
   mesmo_func(Fb,Fm),
   salario(Fb,Sb),salario(Fm,Sm),Sb<Sm,
   cargo(Fb,Cb),cargo(Fm,Cm),Cb\==Cm,
   nome(Fb,Nome).
```

(b) Regra foi_promovido

Figura 4: Definição de regras manuais

```
mesmo_func(Fb,Fm):-
   funcionario(v1,Fb),funcionario(v2,Fm),
   cpf(Fb,CPF),cpf(Fm,CPF).
```

Figura 5: Definição da regra mesmo_func

Após o pré-processamento, a máquina de inferência atua sobre os fatos, considerando as regras manuais relativas ao domínio da aplicação. Com a aplicação do conjunto de regras, pode-se obter a razão da evolução do documento XML de v1 para v2. O ambiente proposto neste trabalho é genérico e pode ser utilizado em outros contextos, especificando-se apenas as regras manuais de acordo com o domínio.

4. EXEMPLO DE UTILIZAÇÃO

Para avaliar a viabilidade da proposta, implementou-se um protótipo em Java para criar um ambiente conciso, onde o usuário pode acompanhar o processo participando das etapas descritas anteriormente. A biblioteca tuProlog [Denti *et al.* 2001], que possibilita o uso da máquina de inferência Prolog em uma aplicação desenvolvida em Java, também foi usada. Um subconjunto de instruções da máquina Prolog contendo as suas propriedades essenciais é fornecido pela biblioteca e proporciona um ambiente capaz de carregar bases de conhecimento, processar consultas Prolog e fornecer as respostas desejadas.

Dando continuidade ao passo a passo descrito na Seção 3 e analisando novamente as duas versões do documento XML, cujos fragmentos são apresentados na Figura 1, é possível visualizar as modificações que foram efetuadas[1]. As características alteradas sobre os funcionários da empresa são mostradas na Tabela 1. Um delta[2] pode ser gerado para identificar essas modificações, mas isso não significa que será possível saber a razão delas.

[1] As versões completas do documento XML podem ser encontradas em http://sel.ic.uff.br/xchange

[2] O termo delta (diferença) representa o que mudou entre duas versões consecutivas (LEON, 2000).

Tabela 1: Modificações no documento XML

Funcionário	**Modificações (v1 para v2)**
João	Salario e cargo
Pedro	Filial
José	Cargo
Maria	Departamento
Ana	Salario
Tiago	Salario, cargo e filial
Lucas	Admitido
Joaquim, Jonas	Demitido

Com a máquina de inferência trabalhando com as regras manuais é possível verificar que, por exemplo, Tiago foi promovido e transferido, pois teve seu salário, cargo e filial alterados, e que João foi promovido, pois somente seu cargo e salário foram alterados. Ou seja, a versão 2 apresenta a evolução da empresa a partir da versão 1.

A razão das modificações ocorridas em um documento XML pode ser obtida a partir da composição das regras manuais individuais. Desta forma, todas as modificações semânticas entre duas versões de um documento XML podem ser identificadas. Um exemplo disso é apresentado na Figura 6. Nesse caso foram consideradas informações inferidas para todos os funcionários que: receberam aumento, foram promovidos, mudaram de função, foram transferidos, mudaram de departamento e foram promovidos e transferidos. O resultado da consulta à regra descrita na Figura 6 pode ser visualizado na Figura 7.

```
razao_mudancas :-
findall(N1,recebeu_aumento(N1),N1lista),write('Receberam aumento: '),write(N1lista), nl,
findall(N2,foi_promovido(N2),N2lista),write('Foram promovidos: '),write(N2lista), nl,
findall(N3,nova_funcao(N3),N3lista),write('Mudaram de funcao: '),write(N3lista), nl,
findall(N4,foi_transferido(N4),N4lista),write('Foram transferidos: '), write(N4lista),nl,
findall(N5,novo_depto(N5),N5lista),write('Mudaram de departamento: '),write(N5lista), nl,
findall(N6,foi_promovido_e_transferido(N6),N6lista),write('Foram Promovidos e
Transferidos:'), write(N6lista), nl,
findall(N7,foi_demitido(N7),N7lista),write('Foram demitidos: '),write(N7lista), nl,
findall(N7,foi_admitido(N8),N8lista),write('Foram admitidos: '),write(N8lista), nl.
```

Figura 6: Regra de evolução do documento XML

```
Receberam aumento: [Ana]
Foram promovidos: [Joao, Tiago, Paula]
Mudaram de funcao: [Jose]
Foram transferidos: [Pedro, Tiago]
Mudaram de departamento: [Maria]
Foram Promovidos e Transferidos: [Tiago]
Foram demitidos: [Joaquim, Jonas]
Foram admitidos: [Lucas]
```

Figura 7: Resultado obtido a partir da regra de evolução

5. TRABALHOS RELACIONADOS

Existem algumas iniciativas relacionadas à proposta deste artigo, que foram identificadas a partir de um mapeamento sistemático da literatura. O mapeamento sistemático da literatura é um meio de descobrir, avaliar e interpretar as pesquisas disponíveis e relevantes sobre uma questão de pesquisa, um tópico ou um fenômeno de interesse [Kitchenham 2004]. O procedimento de seleção dos artigos foi realizado em três etapas. Na primeira etapa do mapeamento sistemático foi realizada uma seleção e catalogação preliminar de artigos a partir da busca nas bibliotecas Compendex, IEEE Xplore e Scopus seguindo os critérios definidos no protocolo da revisão sistemática. Todas as 403 publicações retornadas foram analisadas com base em alguns critérios de exclusão e 67 publicações foram aceitas. Em uma terceira etapa, após nova análise, 9 publicações foram identificadas como relevantes no contexto da pesquisa e são descritas a seguir.

Existem algumas ferramentas (ou trabalhos publicados) para detecção de modificações ou atualizações em uma página Web que é frequentemente acessada. WebVigiL [Chamakura *et al.* 2005] é um sistema

de monitoramento de modificações, notificação e de apresentação de páginas Web de propósito geral. Lim e Ng [2001] propõem um algoritmo heurístico para detectar automaticamente modificações entre dois documentos HTML. Uma extensão do trabalho, apresentada em Lim e Ng [2004], visa a descoberta de mudanças entre dois arquivos HTML ou XML, hierarquicamente estruturados e representados como uma árvore ordenada. Já os trabalhos apresentados por Zhao et al. [2004], [2006] atuam na identificação de estruturas que mudam frequentemente em documentos XML, usando uma abordagem denominada mineração de delta estrutural. O objetivo é extrair conhecimento a partir de sequências de modificações estruturais em documentos XML. O foco destas abordagens está relacionado às mudanças sintáticas enquanto a abordagem proposta neste artigo está interessada em alterações semânticas.

Em Wang et al. [2003], Song et al. [2007], Cobena et al. [2002], o foco está relacionado às operações de transformação, ou seja, dada uma versão 1 de um documento XML, o objetivo é descobrir qual é a ordem correta das operações, para se obter a versão 2. Wang et al. [2003] propõem o algoritmo X-Diff que utiliza o conceito de árvores não ordenadas para verificar modificações entre versões de documentos XML e se preocupa em garantir a minimalidade das sequências de operações de transformação. Uma extensão deste algoritmo é o BIODIFF [Song *et al.* 2007], usado na Biologia Molecular Computacional para detectar diferenças exatas em dados que representam genomas, para fins de estudos das relações existentes entre diferentes espécies de seres vivos. Por fim, Cobena et al. [2002] propõem o algoritmo XyDiff, para verificar modificações entre versões de documentos XML para o projeto Xyleme [Abiteboul S. *et al.* 2002], que investigava *data warehouses* dinâmicos para armazenamento de grandes volumes de dados. Apesar da relevância dessas abordagens, o foco difere do trabalho proposto neste artigo, que vai além da ordem correta das operações de transformação, tentando identificar mudanças semânticas entre as versões do documento XML.

6. CONSIDERAÇÕES FINAIS

Gerenciar as modificações em documentos XML é uma necessidade cada vez mais crescente. Diante disso, este artigo apresenta uma abordagem de evolução de documentos XML baseada em inferência, utilizando Prolog. Esta proposta optou por elaborar formas de compreender a evolução de documentos XML em alto nível. As versões de um documento XML foram transformadas em fatos Prolog e, a partir de um conjunto de regras, foi possível identificar a intenção do usuário ao criar a segunda versão.

Embora as informações iniciais estejam representadas em documentos XML, os usuários especialistas elaboram as regras de inferência em Prolog, uma vez que os dados são traduzidos para esta linguagem. Uma sugestão de trabalho futuro é a adoção de uma interface gráfica que gere consultas Prolog a partir de uma seleção de opções em alto nível. Ainda que muitos dos usuários sejam de áreas tecnológicas, nem todos podem estar familiarizados com Prolog e esta interface facilitaria o processo de geração de consultas. Outros aspectos a serem explorados consistem no uso desta proposta não somente na análise de diferenças, como também no processo de combinação de versões, na evolução de esquemas [Moon *et al.* 2010] e no modo como a abordagem se comportará com uma base de dados reais.

Visando complementar a pesquisa de trabalhos relacionados, uma busca manual está sendo realizada na DBLP (com foco nas publicações da conferência VLDB), já que a busca automática não é suficiente para um mapeamento completo [Randolph 2009]. Outra estratégia, já em execução, diz respeito ao mapeamento das publicações relevantes, feito por meio da análise das publicações que são citadas nos trabalhos já identificados, bem como das publicações que citam os trabalhos já identificados. Com esta busca manual, espera-se alcançar novos trabalhos que também poderiam contribuir com esta abordagem. Além disso, o protocolo da revisão sistemática está sendo reexecutado com o objetivo de encontrar novas publicações relevantes, principalmente do ano corrente.

Uma limitação da abordagem atual que já está sendo resolvida em um trabalho em andamento, consiste em identificar automaticamente elementos correspondentes entre duas versões de um documento XML. Atualmente, o usuário especialista precisa cadastrar uma regra manual (que no

exemplo deste artigo é chamada de *mesmo_func* (Figura 5)), cuja função é identificar qual entidade na v1 corresponde a qual entidade na v2. Essa regra usa um atributo chave (no exemplo, o cpf). No entanto, dependendo de como os documentos XML estejam sendo gerenciados, não há garantias de que o valor permaneça o mesmo entre duas versões (por exemplo, se houver algum erro de digitação na v1 que foi corrigido na v2). A proposta é resolver esta limitação usando uma abordagem de cálculo de similaridade [Dorneles *et al.* 2009], evitando a necessidade de uso de atributos chave.

AGRADECIMENTOS

Os autores gostariam de agradecer ao CNPq (processos 305276/2010-7 e 305283/2011-1) e à FAPERJ (processos E-26/101.512/2010 e E-26/103.253/2011) pelo apoio financeiro.

REFERENCIAS

ABITEBOUL S., CLUET S., FERRAN G., ROUSSET M.-C. THE XYLEME PROJECT. COMPUTER NETWORKS 39(3): 225-238, 2002.

BOLEY H. THE RULE MARKUP LANGUAGE: RDF-XML DATA MODEL, XML SCHEMA HIERARCHY, AND XSL TRANSFORMATIONS. IN INTERNATIONAL CONFERENCE ON WEB KNOWLEDGE MANAGEMENT AND DECISION SUPPORT INAP'01., SPRINGER-VERLAG, BERLIN, HEIDELBERG, P. 5–22, 2003.

BRATKO I. PROLOG PROGRAMMING FOR ARTIFICIAL INTELLIGENCE. ADDISON WESLEY, HARLOW, ENGLAND; NEW YORK, 2001.

BRAY T., PAOLI J., SPERBERG-MCQUEEN C. M., MALER E., YERGEAU F. EXTENSIBLE MARKUP LANGUAGE (XML) 1.0 (FIFTH EDITION). HTTP://WWW.W3.ORG/TR/XML/, 2008.

CHAMAKURA S., SACHDE A., CHAKRAVARTHY S., ARORA A. WEBVIGIL: MONITORING MULTIPLE WEB PAGES AND PRESENTATION OF XML PAGES. IN DATA ENGINEERING WORKSHOPS, P. 1276, 2005.

CHIEN S.-Y., TSOTRAS V. J., ZANIOLO C. EFFICIENT MANAGEMENT OF MULTIVERSION DOCUMENTS BY OBJECT REFERENCING., 2001.

COBENA G., S. ABITEBOUL, MARIAN A. DETECTING CHANGES IN XML DOCUMENTS. IN INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE), SAN JOSE, CA, USA, P. 41-52, 2002.

DENTI E., OMICINI A., RICCI A. TUPROLOG: A LIGHT-WEIGHT PROLOG FOR INTERNET APPLICATIONS AND INFRASTRUCTURES. PRACTICAL ASPECTS OF DECLARATIVE LANGUAGES: 184–198, 2001.

DORNELES C. F., NUNES M. F., HEUSER C. A., MOREIRA V. P., SILVA A. S. DA, MOURA E. S. DE. A STRATEGY FOR ALLOWING MEANINGFUL AND COMPARABLE SCORES IN APPROXIMATE MATCHING. INF. SYST. 34(8): 740–756, 2009.

HUANG S. S., GREEN T. J., LOO B. T. DATALOG AND EMERGING APPLICATIONS: AN INTERACTIVE TUTORIAL. IN INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA SIGMOD '11., ACM, NEW YORK, NY, USA, P. 1213–1216, 2011.

KITCHENHAM B. PROCEDURES FOR PERFORMING SYSTEMATIC REVIEWS. DEPARTMENT OF COMPUTER SCIENCE, KEELE UNIVERSITY AND NATIONAL ICT, AUSTRALIA, 2004.

LIM S. J., NG Y.-K. AN AUTOMATED CHANGE DETECTION ALGORITHM FOR HTML DOCUMENTS BASED ON SEMANTIC HIERARCHIES. IN PROCEEDINGS OF THE 17TH INTERNATIONAL CONFERENCE ON DATA ENGINEERING, IEEE COMPUTER SOCIETY, WASHINGTON, DC, USA, P. 303–312, 2001.

LIM S. J., NG Y.-K. CHANGE DISCOVERY OF HIERARCHICALLY STRUCTURED, ORDER-SENSITIVE DATA IN HTML/XML DOCUMENTS. IN APPLICATIONS AND THE INTERNET, IEEE COMPUTER SOCIETY, LOS ALAMITOS, CA, USA, P. 178, 2004.

LIMA D., DELGADO C., MURTA L., BRAGANHOLO V. TOWARDS QUERYING IMPLICIT KNOWLEDGE IN XML DOCUMENTS. JOURNAL OF INFORMATION AND DATA MANAGEMENT 3(1): 51, 2012.

MACKENZIE D., EGGERT P., STALLMAN R. COMPARING AND MERGING FILES - GNU DIFFUTILS MANUAL., 2011.

MOON H. J., CURINO C. A., ZANIOLO C. SCALABLE ARCHITECTURE AND QUERY OPTIMIZATION FORTRANSACTION-TIME DBS WITH EVOLVING SCHEMAS. IN INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA SIGMOD '10., ACM, NEW YORK, NY, USA, P. 207–218, 2010.

RANDOLPH J. A GUIDE TO WRITING THE DISSERTATION LITERATURE REVIEW. PRACTICAL ASSESSMENT, RESEARCH & EVALUATION 14(13), 2009.

RIZZOLO F., VAISMAN A. A. TEMPORAL XML: MODELING, INDEXING, AND QUERY PROCESSING. THE VLDB JOURNAL 17(5): 1179–1212, 2008.

SONG Y., BHOWMICK S. S., DEWEY,JR. C. F. BIODIFF: AN EFFECTIVE FAST CHANGE DETECTION ALGORITHM FOR BIOLOGICAL ANNOTATIONS. IN INTERNATIONAL CONFERENCE ON DATABASE SYSTEMS FOR ADVANCED APPLICATIONS, SPRINGER-VERLAG, BERLIN, HEIDELBERG, P. 275–287, 2007.

WANG Y., WITT D. J. DE, CAI J. X-DIFF: AN EFFECTIVE CHANGE DETECTION ALGORITHM FOR XML DOCUMENTS. IN INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE), BANGALORE, INDIA, P. 519-530, 2003.

ZHAO Q., BHOWMICK S. S., MOHANIA M., KAMBAYASHI Y. DISCOVERING FREQUENTLY CHANGING STRUCTURES FROM HISTORICAL STRUCTURAL DELTAS OF UNORDERED XML. IN ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT CIKM'04., ACM PRESS, WASHINGTON D.C., U.S.A., P. 188, 2004.

ZHAO Q., CHEN L., BHOWMICK S. S., MADRIA S. XML STRUCTURAL DELTA MINING: ISSUES AND CHALLENGES. DATA KNOWL. ENG. 59(3): 627–651, 2006.