Um *Framework* para Suporte à Preparação e Comparação de Similaridade de Documentos XML*

Fábio dos Santos, Carlos Alberto Souza Junior, Rodrigo Gonçalves, Ronaldo dos Santos Mello

[1]Departamento de Informática e Estatística – Centro Tecnológico
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brasil

{haqim,carlosm,rodrigog,ronaldo}@inf.ufsc.br

Abstract. *One way to publish information in the Web is to create XML data sources. In these data sources, information is contained in one or more XML documents with a particular structure and content format. In this paper, we introduce a framework to prepare and to support the comparison of XML documents from different data sources, aiming at a further integration of similar XML instances. It is composed by some processes with one or more stages. The main contribution of this framework is to facilitate the similarity score definition between heterogeneous XML instances, allowing an uniformization of XML data defined in different contexts by different authors.*

Resumo. *Uma das formas utilizadas para publicação de informações na Web é a disponibilização de fontes de dados XML. Nestas fontes, as informações estão contidas em um ou mais documentos XML que possuem uma estrutura e padronização de conteúdo particular. Este artigo apresenta um framework para preparar e apoiar o processo de comparação de documentos XML advindos de diferentes fontes, visando uma posterior integração de instâncias XML similares. Ele é composto por alguns processos que possuem um ou mais estágios. A principal contribuição deste framework é facilitar a determinação de similaridade entre instâncias XML heterogêneas, possibilitando uma melhor uniformização dos dados XML definidos em diferentes contextos por diferentes autores.*

1. Introdução

Junto a métodos tradicionais de publicação como revistas e livros, a Web tornou-se uma das formas mais populares para a publicação de informações, em parte decorrente do baixo custo e facilidade de acesso. Entretanto, não existe um controle central sobre a forma como estas informações são publicadas. Desta forma, é comum a existência de informações duplicadas entre duas ou mais fontes de dados da Web. A recuperação e a integração destas informações muitas vezes é trabalhosa, devido à necessidade de eliminar esta duplicidade indesejável de informação [Carvalho and da Silva 2003].

*O desenvolvimento deste trabalho conta com o suporte financeiro do Projeto DIGITEX do CNPq. Número do Processo Institucional: CTInfo 550.845/2005-4.

No caso de fontes de dados na Web baseadas em documentos XML (eXtensible Markup Language)[1], têm-se outros fatores que tornam ainda mais complicado este processo de integração. Dada a natureza dinâmica dos documentos XML e a liberdade na forma de estruturá-los, é muito comum que uma mesma informação seja representada de várias formas distintas entre si na estrutura e no tipo de dado [Flesca et al. 2002].

Motivado por esta problemática, um *framework*[2] de suporte à preparação e comparação de similaridade entre documentos XML é apresentado, visando uma posterior integração das suas instâncias de dados. O foco deste artigo são as atividades de preparação de documentos XML, denominadas atividades de *pré-processamento*. Tais atividades são bastante importantes, uma vez que documentos XML são documentos semi-estruturados e cada documento pode definir sua própria representação de uma informação. Desta forma, faz-se necessária uma uniformização na estrutura de seus dados, buscando melhorar o desempenho da determinação de similaridade entre as instâncias XML nestes documentos.

O desenvolvimento deste *framework* leva em conta a necessidade de suprir deficiências encontradas em trabalhos relacionados à determinação de similaridade entre instâncias XML [Wang et al. 1994, Jiang et al. 1995, Nierman and Jagadish 2002, pong Leung et al. 2003, Buttler 2004, Flesca et al. 2002, Carvalho and da Silva 2003, Dorneles et al. 2004]. Estas deficiências estão relacionadas ao fato de que cada trabalho foca na definição de uma única forma de comparação entre documentos XML, sem considerar que, sob diferentes contextos, a natureza dinâmica dos documentos XML pode influenciar negativamente a comparação. Esta problemática é considerada no contexto do *framework* proposto, que busca ser dinâmico e adaptável a diferentes contextos.

A aplicação imediata deste *framework* é a validação de pesquisas desenvolvidas no Grupo de Banco de Dados da UFSC[3] cujo objetivo é a determinação de similaridade e o casamento de instâncias de dados XML. Estas pesquisas estão ocorrendo no ambito do projeto *Digitex*. Este projeto visa a autoria, busca e integração de instâncias XML referentes a dados de bibliotecas digitais. A intenção do *framework* neste contexto é servir como uma base para a implementação e teste de algoritmos de comparação e integração de dados XML, permitindo futuras melhorias nos processos que executam estas atividades. Desta forma, diversas vantagens são obtidas: análise da execução combinada de diferentes algoritmos, em diferentes seqüências de execução; comparação de resultados da aplicação de diferentes abordagens de determinação de similaridade de instâncias XML; construção de uma plataforma de fácil expansão, que permita a inclusão de novos processos ou a implementação de processos alternativos de comparação e integração de dados.

Este artigo está organizado da seguinte forma. A seção dois apresenta uma revisão dos trabalhos relacionados. A seção três descreve o projeto das atividades de *pré-processamento* realizadas pelo *framework*, que é o foco deste artigo, bem como do processo que executa estas atividades. A seção quatro descreve a modelagem do *framework* e como está sendo conduzida a sua implementação, incluindo tarefas de teste,

[1] http://www.w3.org/XML/

[2] Termos como *tag* e *framework*, apesar de apresentarem tradução para o português, serão utilizados na sua língua original neste artigo. Isto deve-se à maior popularidade destas formas em relação às formas traduzidas.

[3] http://www.grupobd.inf.ufsc.br/

documentação e tecnologias adotadas. A última seção é dedicada às considerações finais.

2. Trabalhos Relacionados

Diversos trabalhos relacionados às atividades suportadas pelo *framework* podem ser encontrados na literatura. Os trabalhos pioneiros de Wang et al. [Wang et al. 1994, Jiang et al. 1995] analisam as estruturas em árvore dos documentos XML para determinar a similaridade dos mesmos. Jagadish et al. [Nierman and Jagadish 2002] utiliza uma abordagem similar, baseada no cálculo da distância de edição entre as árvores como medida de similaridade entre documentos XML.

O trabalho de Leung et al. [pong Leung et al. 2003] propõe um algoritmo baseado em mineração de dados para calcular a similaridade estrutural entre documentos XML. Já a utilização de séries temporais para determinar a similaridade entre os documentos sendo comparados é a idéia proposta pelo trabalho de Flesca et al. [Flesca et al. 2002]. O trabalho de Buttler et al. [Buttler 2004] propõe-se a utilizar os diversos caminhos hierarquicos (*paths*) extraídos dos documentos XML como forma de estabelecer a comparação de similaridade entre eles.

Outros trabalhos como o de Weis et al. [Weis and Naumann 2004] e de Carvalho et al. [Carvalho and da Silva 2003] consideram a estrutura e o conteúdo dos documentos XML na determinação da similaridade entre os documentos XML.

Existem ainda trabalhos que focam em aspectos mais pontuais da comparação entre documentos XML. O trabalho de Dorneles et al. [Dorneles et al. 2004], por exemplo, trata a comparação entre tuplas e listas em documentos semi-estruturados, com ênfase em documentos XML. Ele estabelece métricas que consideram os aspectos semânticos das estruturas na análise da similaridade.

Todos estes trabalhos, apesar de apresentarem diferentes abordagens para tratar o problema, podem ser integrados ao *framework*, aproveitando-se dos recursos relacionados à preparação e compatibilização de documentos XML que ele disponibiliza. Desta forma, a eficiência de seus algoritmos e sistemas de comparação pode ser testada e melhorada. Além disso, o *framework* possibilita que estas e novas abordagens possam ser comparadas entre si através dos resultados gerados por elas.

Assim, o *framework* proposto apresenta um enfoque diferente dos trabalhos relacionados. Ele não se propõe a apresentar uma nova abordagem para comparação de similaridade, mas sim a melhorar o desempenho de abordagens já existentes através de suas atividades de *pré-processamento*. Devido a sua natureza aberta e genérica, tais abordagens podem ser incorporadas a ele e produzir resultados de comparação com melhor desempenho. Nenhuma proposta similar na literatura foi encontrada.

3. Projeto do *Framework*

O *framework* está dividido em três atividades (ou processos) de pré-processamento de documentos XML, compostos cada um por vários estágios, e dois processos de comparação de similaridade. Cada um destes processos pode ou não ser invocado em um processo de determinação de similaridade, conforme a necessidade. Desta forma, o *framework* trabalha de forma flexível, permitindo a otimização de processamento (em termos de tempo e de qualidade) e a fácil execução de testes para analisar o funcionamento de algortimos de similaridade.

O primeiro processo faz uma análise e compatibilização das *tags* dos elementos dos documentos XML, buscando identificar *tags* com semântica equivalente, porém grafia distinta. O resultado do mesmo é a uniformização da grafia destas *tags*.

O segundo processo realiza operações simples de caráter estrutural nos elementos, buscando uniformizar as estruturas XML e otimizar a execução do processo seguinte. O terceiro processo realiza mudanças maiores e mais complexas, buscando uniformizar da melhor forma possível as estruturas dos documentos XML. Assim, os processamentos posteriores à preparação, que realizam a comparação dos documentos, não precisam se preocupar com diferenças originárias das suas estruturas. Aqui pode-se também decidir se é valido qualquer comparação subseqüente das instâncias, caso uma baixa similaridade estrutural entre os documentos for encontrada.

Os dois últimos processos são os responsáveis por calcular a similaridade entre os documentos XML e não estão implementados no momento. Portanto, eles não são apresentados, sendo, como comentado anteriormente, o foco deste artigo os processos de pré-processamento.

3.1. Estrutura dos Processos de Pré-Processamento do *Framework*

As atividades de pré-processamento do *framework* estão organizadas em três processos: *TagsProcess*, *BasicStructureProcess* e *AdvancedStructureProcess*. Estes estágios em geral são independentes uns dos outros, necessitando apenas que os dados de entrada estejam disponíveis para que possam gerar uma saída adequada. Vale ressaltar que um aspecto importante da estrutura geral do *framework* é a utilização de uma classe chamada *StageData*. Ela é responsável por realizar a comunicação e a passagem de dados entre os estágios de um determinado processo, permitindo que eles troquem informações entre si.

A estruturação proposta visa a independência dos processos entre si, ou seja, pode-se incluir ou remover processos ou estágios dos procesos a fim de obter uma melhor qualidade de comparação e/ou aumentar o desempenho de um determinado algoritmo. Ela auxilia também na execução de testes, como descrito anteriormente, pois é possível desabilitar qualquer processo/estágio e testá-los individualmente ou em conjunto.

Apesar da independência de processos e estágios proposta, uma ordem de execução deva ser respeitada para o correto funcionamento dos processos. Por exemplo, não faz sentido analisar as estruturas de dois documentos XML sem antes tratar as *tags* dos documentos para corrigir eventuais diferenças léxicas entre as mesmas. Por isso os cinco processos, quando presentes, ocorrem sempre na ordem descrita na seção anterior.

Os processos que compõem o *framework* são detalhados nas próximas seções. Seu fluxo de execução está ilustrado na Figura 1. A modelagem do *framework* segue uma abordagem orientada a objetos, sendo composta por um conjunto de classes organizado em quatro pacotes básicos, conforme mostra a Figura 2.

3.2. *Tags Process*

Este processo prepara os nomes dos elementos, buscando homogeneizar a forma como as *tags* que representam o mesmo conceito estão escritas. O objetivo aqui é ocultar diferenças nos nomes das *tags* com mesma semântica, a fim de não afetar a qualidade de um processo de comparação. Os seguintes estágios compõem este processo:

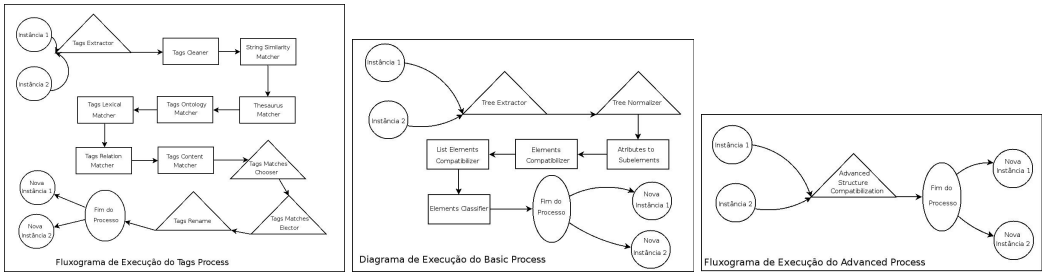

Figura 1. Fluxograma de execução dos estágios

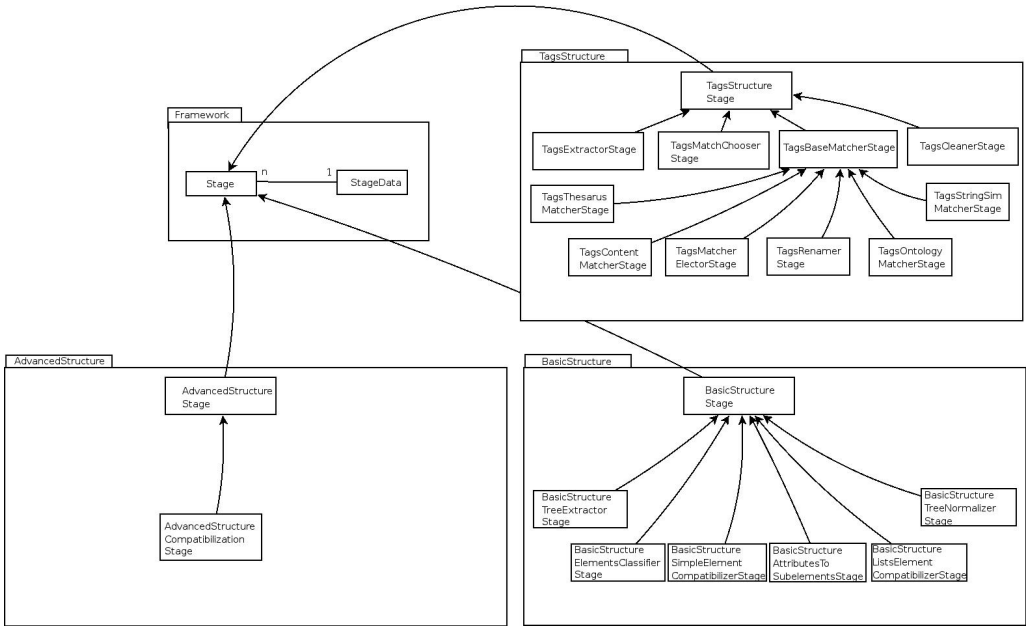

Figura 2. Modelagem das classes do *framework*

- **Tags Extractor**: extrai os nomes das *tags* e dos atributos presentes nos documentos. Não há nomes iguais no resultado deste estágio, mesmo se houver *tags* e atributos com mesmo nome. Portanto, todos os nomes extraídos são diferentes;
- **Tags Cleaner**: remove caracteres especiais, padroniza palavras compostas e remove numerais ao final de palavras. Por exemplo, a *tag* `DataDeNascimento` pode ficar `Data_De_Nascimento` ou vice-versa. A *tag* `article21` torna-se apenas `article`;
- **String Similarity Matcher**: utiliza uma métrica de similaridade para realizar a comparação dos nomes dos elementos obtidos nos estágios anteriores[4]. A partir de um conjunto de *tags* identificadas como equivalentes, elege-se uma única *tag* para representar este conjunto na versão final dos documentos XML;
- **Thesaurus Matcher**: realiza o casamento de elementos com base em um *thesaurus*. O casamento é determinado por uma métrica de similaridade (pode ser a mesma aplicada no estágio anterior), sendo que o casamento ocorre se o grau de similaridade for igual ou maior a um *threshold* previamente definido;
- **Tags Ontology Matcher**: realiza o casamento de elementos com base em uma

[4] A métrica utilizada atualmente para comparação é a *Jaro Winkler*. Futuramente, pretende-se flexibilizar no *framework* a escolha da métrica a ser utilizada.

ontologia. Uma análise semântica dos dados é realizada para determinar a equivalência entre os mesmos. O casamento ocorre da mesma forma que o estágio anterior;

- **Tags Content Matcher**: realiza o casamento dos elementos pelo conteúdo das suas *tags*. Todos os conteúdos das tags que fazem parte de dois elementos XML são buscados, definindo-se grupos de conteúdos dos mesmos. A similaridade é então verificada pela comparação desses grupos e o casamento também considera um *threshold* mínimo;
- **Tags Chooser**: a partir de uma lista com os casamentos realizados em estágios anteriores e seus respectivos graus de similaridades, define conjuntos de similaridade. Cada conjunto é formado por um grupo de nomes de elementos similares;
- **Tags Elector**: elege os nomes das *tags* que irão substituir os demais semelhantes na versão final dos documentos XML, para cada conjunto de similaridade gerado no estágio anterior. A ontologia e a *thesaurus* podem ajudar na escolha do melhor representante. Caso isso não ocorra, adota-se o exemplar do conjunto que, em média, mais se assemelha aos demais;
- **Tags Renaming**: substitui os nomes das *tags* nos documentos XML, uma vez concluída a compatibilização das *tags* e a eleição dos representantes pelos estágios anteriores.

3.3. Basic Structure Process

Este processo é responsável pela adequação e uniformização inicial das estruturas dos elementos presentes nos documentos XML. Os estágios deste processo são os seguintes:

- **Basic Structure Stage**: carrega, através do *StageData*, os documentos XML a serem analisados;
- **Tree Extractor**: define as estruturas dos documentos obtidos no estágio anterior. Estas estruturas são representadas por árvores que, no *framework*, são contruídas utilizando a biblioteca JDSL[5]. Estas árvores servem de base para a comparação estrutural dos documentos pelos próximos estágios;
  Tree Normalizer: gera a *árvore normalizada* de cada estrutura em árvore obtida no estágio anterior. Uma árvore normalizada é aquela que não tem elementos repetidos.A Figura 3 exemplifica a execução deste estágio.

Árvore original	Árvore normalizada

autores
  autor
  nome
  autor
  nome
	autores
	  autor
	  nome

Figura 3. Exemplo de funcionamento do Tree Normalizer

- **Element Classifier**: gera as listas com as possíveis estruturas dos elementos para poder comparar as mesmas posteriormente. Cada lista guarda as estruturas de cada elemento encontrado no documento. Por exemplo, se os documentos contêm vários elementos `autores` e vários elementos `nome`, cada estrutura presente em `autores` é armazenada numa lista e as estruturas presentes em `nome` em outra. O resultado deste processo é armazenado no `StageData`.

[5]http://www.cs.brown.edu/cgc/jdsl/

- **Simple Element Compatibilizer**: busca, a partir dos resultados obtidos do estágio anterior, elementos de mesmo nome que estão definidos como simples e complexo. Uma vez encontrados, os elementos complexos são transformados em elementos simples. A Figura 4 demonstra tal transformação.

Entrada	<endereco> <cidade>Florianópolis</cidade> <rua>Álvaro de Carvalho</rua> </endereco>	<endereco>Florianópolis</endereco>
Saída	<endereco>Florianópolis Álvaro de Carvalho</endereco>	<endereco>Florianópolis</endereco>

Figura 4. Exemplo de funcionamento do Simple Elements Compatibilizer

- **List Element Compatibilizer**: busca elementos que são simples e listas ao mesmo tempo. Caso sejam encontrados tais elementos, os elementos listas são transformados em elementos simples, conforme exemplifica a Figura 5.

Entrada	<autores> <nome>Heuser</nome> <nome>Ronaldo</nome> </autores>	<autores>Heuser Ronaldo Júlio</autores>
Saída	<autores>Heuser Ronaldo</autores>	<autores>Heuser Ronaldo Júlio</autores>

Figura 5. Exemplo de funcionamento do List Element Compatibilizer

- **Attribute Compatibilizer**: transforma os atributos em sub-elementos de seus respectivos elementos. Desta forma, reduz-se as diferenças estruturais entre os elementos. A Figura 6 demonstra o seu funcionamento.

Entrada	<livro ano=``1997´´> </autores>
Saída	<livro> <ano>1997</ano> </livro>

Figura 6. Exemplo de funcionamento do Attribute Compatibilizer

3.4. Advanced Structure Process

Este processo dá continuidade às transformações estruturais nas instâncias XML. Técnicas mais complexas são utilizadas para tornar mais similares as estruturas das instâncias sendo comparadas.

Este processo possui um único estágio chamado *Complex Element Compatibilizer*. Neste estágio é feita a análise e compatibilização de categorias de elementos complexos. Para aqueles elementos onde existem sub-elementos diretos não compatíveis, ou seja, que existem em uma instância e não existem em outra (por exemplo, um elemento `autor` que

possui o sub-elemento direto `idade` e outro `autor` que não possui este sub-elemento), o processo realiza transformações nas estruturas das instâncias, visando torná-las mais compatíveis. Essa atividade tem o propósito de não prejudicar o cálculo final de similaridade, pois se um elemento é não comum do ponto de vista estrutural, mesmo assim, ele pode ter um conteúdo bem similar a outro elemento.

Uma vez detectados os elementos complexos com estruturas não compatíveis, três possíveis abordagens podem ser atualmente aplicadas para resolver as incompatibilidades encontradas:

- *Lista de dados*: define-se um sub-elemento `list` que possui como sub-elementos os conteúdos incompatíveis. A saída 1 da Figura 7 mostra o resultado desta transformação, considerando duas instâncias XML como entrada;

Entrada	`<endereco>`
	`  <cidade>Florianópolis</cidade>`
	`  <rua>Álvaro de Carvalho</rua>`
	`  <numero>222</numero>`
	`</endereco>`
		`<endereco>`
		`  <cidade>Florianópolis</cidade>`
		`</endereco>`

Saída 1	`<endereco>`
	`  <cidade>Florianópolis</cidade>`
	`  <list>`
	`  <item>Álvaro de Carvalho</item>`
	`  <item>222</item>`
	`  </list>`
	`</endereco>`
		`<endereco>`
		`  <cidade>Florianópolis</cidade>`
		`  <list/>`
		`</endereco>`

Saída 2	`<endereco>`
	`  <cidade>Florianópolis</cidade>`
	`  <list>Álvaro de Carvalho 222</list>`
	`</endereco>`
		`<endereco>`
		`  <cidade>Florianópolis</cidade>`
		`  <list/>`
		`</endereco>`

Saída 3	`<endereco>`
	`  <cidade>Florianópolis</cidade>`
	`</endereco>`
		`<endereco>`
		`  <cidade>Florianópolis</cidade>`
		`</endereco>`

Figura 7. Exemplo de funcionamento do Complex Element Compatibilizer

- *Bloco de texto*: define-se um sub-elemento `list` e seu conteúdo mantém os conteúdos de todos os elementos incompatíveis. Um exemplo é mostrado na saída 2 da Figura 7;
- *Remoção de Incompatibilidades*: remove-se os elementos incompatíveis. A saída 3 da Figura 7 mostra o resultado desta transformação.

4. Implementação e Testes

O *framework* encontra-se atualmente em desenvolvimento. Os processos de pré-processamento estão implementados e os processos de determinação de similaridade estão em fase de projeto e implementação.

A linguagem de programação utilizada é *Java*, que devido a sua portabilidade, possibilita uma maior difusão do *framework*. O programa está sendo desenvolvido com base em testes. Para isto, está-se utilizando unidades de testes automatizadas *JUnit*[6], que consistem em um *framework* que facilita a criação de código para a automação de testes

[6]http://www.junit.org/

com apresentação dos resultados. Com ele, pode ser verificado se cada método de uma classe funciona da forma esperada, exibindo possíveis erros ou falhas.

Além do *JUnit*, outros *frameworks* estão sendo utilizados no desenvolvimento, como *JDSL*, *SAX* e *DOM*. Algumas bibliotecas foram também adotadas para realizar a comparação de *strings* por métricas de similaridade: *SecondString*[7] e *Simmetrics*[8].

Alguns testes já foram realizados com o objetivo de analisar o impacto que as atividades de pré-processamento do *framework* produz em trabalhos relacionados à comparação de documentos XML. Os trabalhos eleitos para os testes foram o de Weis et al. [Weis and Naumann 2004], Jagadish et al. [Nierman and Jagadish 2002] e Carvalho et al. [Carvalho and da Silva 2003].

Uma amostra formada por 200 documentos de conhecidos repositórios XML de dados bibliográficos (*CiteSeer*[9] e *DBLP*[10]) foi utilizada. Dos 200 documentos, cada fonte contribuiu com 100 documentos, onde 50 deles eram duplicatas entre as duas fontes, ou seja, havia um total de 150 documentos distintos.

Os documentos utilizados foram obtidos na Web, dos *sites* das próprias fontes de dados. Eles estão disponibilizados em arquivos no formato XML (DBLP) e Bibtex(Citeseer) - este último foi convertido para XML antes de ser utilizado.

Os testes ocorerram em três etapas: na primeira etapa nenhuma modificação foi aplicada aos documentos XML. Na segunda etapa, apenas o processo *TagsProcess* foi aplicado aos documentos. Na terceira etapa, todos os três processos de pré-processamento foram executados. Como resultado destes testes, verificou-se um aumento progressivo da taxa de *recall*, sem haver perda de precisão (*precision*). No trabalho de Carvalho et al., da primeira para a última etapa houve um aumento de 100% no *recall*, com *precision* final de 94%.

5. Conclusão

Este artigo apresenta um *framework* que se destina à preparação e comparação de documentos XML com vistas à integração de instâncias XML heterogêneas. Sua principal contribuição é atuar como um facilitador para a realização de testes para avaliação da qualidade dos algoritmos de determinação de similaridade de instâncias XML, como por exemplo, *recall* e *precision* dos casamentos obtidos entre as instâncias XML comparadas.

Trabalhos relacionados ao casamento de instâncias XML por similaridade preocupam-se em apresentar métricas que, em geral, não apresentam ótimo desempenho para qualquer amostra de dados. O *framework* proposto possui um objetivo principal diferente, que é melhorar a qualidade dos casamentos realizados por estas métricas através da aplicação de processos de pré-processamento. Os testes descritos na seçao anterior demonstram que este objetivo está sendo alcançado. Além disso, o *framework* serve como uma base para a incorporação de novas métricas que podem utilizar estes processos.

A etapa de integração de instâncias XML é o foco de outro trabalho de pesquisa em desenvolvimento no Grupo de Banco de Dados da UFSC. O objetivo deste trabalho é

[7] http://secondstring.sourceforge.net/

[8] http://sourceforge.net/projects/simmetrics/

[9] http://citeseer.ist.psu.edu

[10] www.informatik.uni-trier.de/ ley/db

a definição de operadores de integração para instâncias XML, bem como de um processo que executa estes operadores. Este trabalho futuramente será integrado ao framework.

Atividades futuras imediatas são a conclusão da implementação de dois processos de determinação de similaridade: *Similarity Weighting Process* e *Similarity Comparison Process*. Eles serão responsáveis por calcular a similaridade entre documentos XML através da atribuição de pesos aos elementos dos documentos (denotando sua importância na comparação) e o cálculo efetivo da similaridade entre eles. A implementação pretende ser desenvolvida de forma flexível, permitindo que métricas já implementadas ou algoritmos de determinação de pesos possam ser incluidos ou removidos dos processos.

Referências

Buttler, D. (2004). A short survey of document structure similarity algorithms. In *International Conference on Internet Computing*, pages 3–9.

Carvalho, J. C. P. and da Silva, A. S. (2003). Finding similar identities among objects from multiple web sources. In Chiang, R. H. L., Laender, A. H. F., and Lim, E.-P., editors, *WIDM*, pages 90–93. ACM.

Dorneles, C. F., Heuser, C. A., Lima, A. E. N., da Silva, A. S., and de Moura, E. S. (2004). Measuring similarity between collection of values. In *WIDM*, pages 56–63.

Flesca, S., Manco, G., Masciari, E., Pontieri, L., and Pugliese, A. (2002). Detecting structural similarities between XML documents. In *WebDB*, pages 55–60.

Jiang, Wang, and Zhang (1995). Alignment of trees – an alternative to tree edit. *TCS: Theoretical Computer Science*, 143.

Nierman, A. and Jagadish, H. V. (2002). Evaluating structural similarity in XML documents. In *WebDB*, pages 61–66.

pong Leung, H., Chung, K. F.-L., and fai Chan, S. C. (2003). A new sequential mining approach to XML document similarity computation. pages 356–362.

Wang, J. T.-L., Zhang, K., Jeong, K., and Shasha, D. (1994). A system for approximate tree matching. *IEEE Trans. Knowl. Data Eng.*, 6(4):559–571.

Weis, M. and Naumann, F. (2004). Detecting duplicate objects in XML documents. In Naumann, F. and Scannapieco, M., editors, *IQIS*, pages 10–19. ACM.