UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ALEXANDER RICHARD VINSON

PathSim: um algoritmo para calcular a similaridade entre caminhos XML

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Carlos Alberto Heuser
Orientador

Porto Alegre, maio de 2007

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Vinson, Alexander Richard

PathSim: um algoritmo para calcular a similaridade entre caminhos XML / Alexander Richard Vinson. – Porto Alegre: PPGC da UFRGS, 2007.

71 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Orientador: Carlos Alberto Heuser.

1. XML. 2. Similaridade. 3. Comparação. 4. Caminhos. I. Heuser, Carlos Alberto. II. Título.

“The simplest answer is usually the correct answer“
— WILLIAM OF OCKHAM

AGRADECIMENTOS

A seção de agradecimentos de um trabalho é uma parte muito importante da monografia, pois é onde muitas pessoas começam a leitura do texto. Portanto, é melhor eu fazer esta parte com bastante calma, para não esquecer de agradecer a ninguém que me auxiliou no decorrer dos anos do mestrado ou que é importante em minha vida.

Inicialmente, eu agradeço aos meus pais, David e Photina, e a minha irmã Christina pela educação que me deram e pelo apoio e incentivo nas minhas jornadas. Vocês são exemplos a serem seguidos.

À minha namorada Sueleny, pela ajuda e incentivo para concretizar este trabalho, além do amor, companheirismo e paciência no decorrer destes sete anos de muita felicidade.

Aos meus amigos Giseli, Claudio, Marcos, Sérgio, Mariusa, Eduardo, Thiago, Gabriel, André Vargas, Lincoln, Raquel e Adrovane, que conheci na Ufrgs e foram excelentes companheiros de grupo, com discussões muitas vezes produtivas, grandes parceiros de churrasco e mesa de bar e bons adversários e colegas de times em jogos de diversos esportes. Em especial, à Giseli e ao Sérgio pela companhia nas disciplinas, pelas discussões sobre o trabalho e por terem ajudado na correção do texto.

Aos meus amigos Ronaldo, Márcio e Diego, pelas discussões interessantes sobre diversos assuntos, companhia para atividades diversas e paciência para me aturar como companheiro de casa.

Aos meus amigos André Panato, Rosário, Frank, Gustavo, Francisco, Letícia, Alexandre, Jñane e Hélio, pela amizade que, mesmo distante, mantém-se firme.

Ao professor Heuser, pela orientação e paciência disponibilizadas no andamento do trabalho e pela confiança depositada em mim, mesmo eu sendo um aluno de tão longe.

Aos professores Altigran e Edleno, pelas discussões sobre o trabalho e idéias para aperfeiçoá-lo.

À professora Viviane, pelo auxílio na definição dos experimentos e métodos para avaliá-los.

À UFRGS, e ao instituto de informática pela infraestrutura disponibilizada.

Ao Cnpq, pelo apoio financeiro nos dois anos de curso, sem o qual o desenvolvimento deste trabalho seria muito mais difícil.

Muito obrigado a todos,
Alexander Richard Vinson

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

EditDistance	Função de distância de edição entre cadeias de caracteres
ER	Entidade-Relacionamento
IDF	Inverse Document Frequency
PathSim	Algoritmo proposto que calcula a similaridade entre caminhos
PathSim_C	PathSim com comparação entre combinações de nomes de elementos
PathSim_A	PathSim com técnicas de alinhamento
TF	Term Frequency
UFRGS	Universidade Federal do Rio Grande do Sul
XML	eXtensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Algoritmos de similaridade que comparam dados expressos em XML são importantes em diversas aplicações que manipulam informações armazenadas nesse padrão. Sistemas de integração de dados XML e de consulta a instâncias XML são exemplos dessas aplicações. A utilização de funções de similaridade para efetuar as comparações nessas aplicações melhora seus resultados finais. A melhora ocorre porque as funções de similaridade possibilitam encontrar estruturas não idênticas às apresentadas nos parâmetros das consultas mas que armazenam informações relevantes.

Uma característica importante que pode ser utilizada para definir se dois elementos XML representam o mesmo objeto real é os caminhos que chegam a estes elementos nas suas respectivas árvores. No entanto, os nodos que representam um determinado objeto real em duas instâncias XML diferentes podem se acessados por caminhos distintos, devido a opções de modelagem dos documentos. Portanto um algoritmo para calcular a similaridade entre caminhos XML é importante para as aplicações descritas acima. Neste contexto, esta dissertação objetiva desenvolver um algoritmo de similaridade entre caminhos XML.

O resultado principal do trabalho é um algoritmo de similaridade entre caminhos XML, nomeado PathSim, que efetua o cálculo de similaridade entre dois caminhos baseado no número mínimo de operações de edição (inserção, remoção e substituição de nomes de elementos) necessárias para transformar um caminho no outro. Além deste algoritmo, foram desenvolvidas três funções de pré-processamento para simplificar os caminhos XML e melhoram os resultados do algoritmo. Adicionalmente, duas variações do algoritmo PathSim são apresentadas, uma incrementada com comparações entre combinações de nomes de elementos, nomeada PathSim$_C$, e a outra auxiliada por técnicas de alinhamento, nomeada PathSim$_A$. Experimentos utilizando documentos XML criados por terceiros, validam empiricamente os algoritmos PathSim e PathSim$_C$. Nos experimentos, os algoritmos foram comparados a uma abordagem para mensurar a similaridade entre caminhos encontrada na literatura. Os algoritmos apresentam melhores resultados que o baseline. Os ganhos variam de acordo com o ambiente onde os caminhos foram extraídos e com as funções de pré-processamento que foram aplicadas aos caminhos.

Palavras-chave: XML, similaridade, comparação, caminhos.

PathSim: A XML path similarity algorithm

ABSTRACT

Similarity algorithms for comparing XML data are important in various applications that manipulate information stored according to this standard. XML data integration systems and XML instance querying systems are examples of such applications. The use of similarity functions to evaluate comparisons in these applications improves their final results. The improvement occurs because similarity functions allow finding structures that are not identical to the query parameter but store relevant information.

One important feature that may be used to define if two XML elements represent the same real world object is the paths that lead to those objects in their corresponding trees. However, the nodes that represent a specific real world object in two different XML instances may be accessed by distinct paths, due to XML design decisions. Thus a method for assessing the similarity of XML paths is important in the applications described above. In this context, the goal of this dissertation is to develop a XML path similarity algorithm.

The main contribution of this work is a XML path similarity algorithm, named PathSim, that calculates the similarity between two paths by computing the minimum number of edit operations (element name insertions, deletions and substitutions) required to transform one path into another. Besides the algorithm, three preprocessing functions were developed to simplify XML paths and improve the results of the algorithm. Additionally, two variations of PathSim algorithm are presented, one enhanced with comparisons among combinations of element names, named PathSim$_C$, and the other one assisted by alignment techniques, named PathSim$_A$. Experiments using XML documents created by third parties validate the algorithms PathSim and PathSim$_C$ empirically. On the experiments, the algorithms are compared to a path similarity algorithm found in the literature. The proposed algorithms presents better results than the baseline. The gains vary according to the environment from which the paths were extracted and to the preprocessing functions applied.

Keywords: XML, similarity, comparison, paths.

1 INTRODUÇÃO

A Linguagem de Marcação Extensível (XML) tornou-se uma forma muito utilizada para armazenar dados de sistemas e trocar informações entre aplicações, principalmente entre as aplicações que têm interfaces disponíveis na Internet (BRADLEY, 1998; XML, 2006). A estrutura de um documento XML depende de que informações o sistema que o manipula necessita e da maneira como o analista, responsável pela modelagem, visualiza a melhor forma de descrever essas informações. Desta forma, os nomes dos elementos e suas estruturas internas são diretamente ligados ao entendimento do analista sobre o domínio da aplicação e à sua experiência em modelar documentos XML. Isto, associado à flexibilidade de modelagem que o padrão XML disponibiliza para a representação de informações, ocasiona o aparecimento de estruturas distintas que armazenam informações equivalentes.

Além de diferenças entre as estruturas, ocorrem, entre documentos XML, diferenças entre conteúdos textuais de elementos que representam o mesmo objeto. Essas diferenças ocorrem, freqüentemente, devido à manipulação das informações expressas em XML ser efetuada por pessoas diferentes. Em (LUJÁN-MORA; PALOMAR, 2001) são identificadas onze causas, cujas combinações justificam diferentes representações textuais da mesma informação em bases de dados nos idiomas Inglês e Espanhol. Entre essas causas estão a omissão de caracteres de acentuação, o uso de abreviações, o uso de termos em idiomas diferentes e a utilização de diferentes representações numéricas como representação arábica, romana ou por extenso. Cabe salientar que o escopo do estudo efetuado nesta dissertação está restrito às diferenças entre estruturas.

As diferenças de estrutura e de conteúdo textual são as principais motivações para o desenvolvimento de funções de similaridade que comparam informações armazenadas em estruturas XML. Os algoritmos de similaridade propõem avaliar a semelhança entre informações contidas nas estruturas e expressá-las por meio de valores reais. Cada algoritmo é baseado em uma ou mais características de documentos XML e de suas estruturas, porém alguns são restritos a certos domínios de aplicação.

A utilização de algoritmos de similaridade para comparar informações expressas em documentos XML é muito importante em processos que manipulam este tipo de documento, pois incrementa os resultados destes processos ao possibilitar que as comparações retornem valores intermediários entre 1 (que denota igualdade) e 0 (que denota total desigualdade). Dependendo do ambiente e do objetivo das aplicações, são várias as motivações para utilizar algoritmos de similaridade para comparar informações expressas em XML.

Em sistemas de integração de documentos, como AutoMed (BOYD et al., 2004; FAN; POULOVASSILIS, 2003) e Mix (BARU et al., June 1999), existe a necessidade de identificar-se representações diferentes da mesma informação. Essa identificação é

importante para evitar que um mesmo objeto seja representado mais de uma vez nas informações disponibilizadas pelo ambiente integrado. A utilização de funções de similaridade associadas a valores de limiar de similaridade auxilia na integração de esquemas (RAHM; BERNSTEIN, 2001) e de instâncias (WANG; MADNICK, 1989) nesses sistemas, unificando elementos equivalentes que apresentem representações diferentes.

Na figura 1.1 são apresentados dois elementos XML equivalentes com estruturas diferentes. Na visão disponibilizada por um ambiente de integração, essa informação deve aparecer somente uma vez. O exemplo apresenta diferenças de estrutura e de conteúdo textual entre instâncias. A primeira é exemplificada pelos elementos "autor" e "escritor", que têm descrições diferentes, mas representam o mesmo tipo de objeto do mundo real, e pelas informações de "instituição" e "titulação", que são representadas por atributos em uma estrutura e por elementos-filho na outra. A segunda é exemplificada pela diferença no conteúdo textual dos elementos "nome" das estruturas.

```
<livro>
    <autor instituicao="UFRGS" titulacao="Doutor">
        <nome>José O. Rech</nome>
    </autor>
</livro>
```

(a)

```
<livro>
    <escritor>
        <instituicao>UFRGS</instituicao>
        <titulacao>Doutor</titulacao>
        <nome>José Oliveira Rech</nome>
    </escritor>
</livro>
```

(b)

Figura 1.1: Exemplo de elementos XML com estruturas diferentes, que armazenam a mesma informação

Em sistemas de consulta a documentos XML, como o disponibilizado pelo Tamino (SCHÖNING, 2001), algoritmos de similaridade podem simplificar a interação do usuário por possibilitar que as comparações sejam efetuadas por funções diferentes da igualdade exata. Portanto, o uso de funções de similaridade permite ao usuário recuperar dados, mesmo sem ter conhecimento da estrutura na qual estes dados se encontram. Por exemplo, uma função de similaridade pode identificar que uma consulta cujo parâmetro é "livros/livro/autor/nome", executada no documento representado na figura 1.1(b) tem maior similaridade com o caminho "livro/escritor/nome" do que com os demais caminhos deste documento, o que indica que os elementos contidos neste caminho têm maiores possibilidades de corresponder à informação solicitada pelo usuário.

Em redes ponto-a-ponto que disponibilizam pesquisas a documentos XML, como Piazza (HALEVY et al., 2004; TATARINOV et al., 2003), um usuário efetua uma consulta em um ponto da rede, e esta é executada nos arquivos disponibilizados pelo próprio ponto e pelos pontos aos quais este tiver acesso. Por exemplo, na rede representada na figura 1.2, se um usuário efetuar uma consulta no ponto E, esta é executada inicialmente nos arquivos dos pontos D, E, F e G. Posteriormente, a consulta pode ser executada em arquivos de outros pontos. Na figura 1.2 apenas são exemplificadas conexões em uma rede ponto-a-ponto, abstendo-se de classificar a topologia e o mecanismo de propagação de consulta adotado. Em redes ponto-a-ponto, funções de similaridade são importantes, pois a consulta pode ser respondida por arquivos localizados em diferentes pontos, e estes arquivos podem ter estrururas diferentes. Se as comparações entre o parâmetro da consulta e as informações nos documentos XML da rede forem efetuadas sem o auxílio de funções de similaridade, a consulta precisa ser adaptada para cada estrutura diferente dos documentos da rede para retornar todos os resultados. No entanto, a principal contribuição

das funções de similaridade, em ambiente ponto-a-ponto, é não exigir que o usuário que realiza a consulta conheça as estruturas de todos os documentos da rede. Essa exigência penalizaria a rede, pois, para ser satisfeita, cada ponto precisaria conter as estruturas de todos os documentos que pode consultar, ocasionando muito tráfego de informações, e tornando a abordagem inviável para grandes redes.

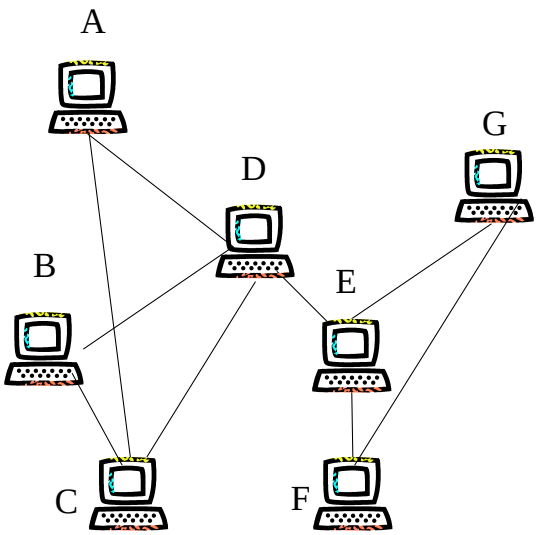


Figura 1.2: Exemplo de rede ponto-a-ponto

Na estrutura de um documento XML, os elementos-filho de um elemento-pai representam objetos reais que compõem o objeto real que o elemento-pai representa. Portanto, o nome do elemento-pai de um elemento-filho é importante para especificar a informação que o elemento-filho contém. Assim, toda a seqüência hierárquica (ou caminho) sobre a qual uma informação está contida, é importante para especificá-la. Por exemplo, na figura 1.1(b), o caminho "livro/escritor/nome" especifica a informação textual "José Oliveira Rech", indicando que é o nome do escritor de um livro.

A especificação do conteúdo textual de um elemento, pelo caminho XML, e a necessidade de mecanismos para comparar, por similaridade, informações representadas em XML são a motivação do estudo e desenvolvimento de uma função de similaridade entre caminhos XML.

O objetivo desta dissertação é desenvolver uma função de similaridade entre caminhos XML. Como resultado da pesquisa realizada, foi definido um algoritmo que calcula a similaridade entre dois caminhos, baseado no número mínimo de operações sobre nomes de elementos necessárias para transformar um caminho no outro. As comparações entre nomes de elementos são efetuadas por funções de similaridade, e os valores são utilizados na computação do valor final de similaridade entre os caminhos. O algoritmo baseia-se na suposição de que a equivalência sintática de caminhos implica em uma equivalência semântica dos mesmos. O algoritmo é nomeado PathSim, e auxilia sistemas que utilizam informações expressas em XML, efetuando comparações por similaridade para mensurar o impacto de diferenças entre as estruturas das estruturas que geraram os caminhos. Três variantes do algoritmo são descritas nesta dissertação:

- **PathSim** é o algoritmo principal de comparação de caminhos;

- **PathSim**$_C$ é a variante do algoritmo PathSim que apresenta suporte à comparação entre combinações de nomes de elementos. Este tipo de comparação é utilizado entre caminhos oriundos de documentos XML do mesmo domínio mas com granularidades diferentes. Quando dois documentos de granularidades diferentes modelam o mesmo domínio, o nome de um elemento de um documento pode ser a combinação de nomes de vários elementos do outro documento;

- **PathSim**$_A$ é a variante do algoritmo PathSim com técnicas de alinhamento de cadeias de caracteres aplicadas as seqüências de nomes de elementos dos caminhos. Essas técnicas são úteis em sistemas que objetivam efetuar pesquisas mais abrangentes, ou seja, consultas que objetivam selecionar um determinado padrão na resposta, independente do contexto em que o padrão apareça. Por exemplo, em uma aplicação deste tipo, o usuário pode informar o caminho "livro/autor" como parâmetro da consulta, com o objetivo de requisitar qualquer informação disponível sobre autores de livros. A resposta a essa consulta deve incluir todas as informações disponíveis sobre qualquer autor de livro, portanto, caminhos como "biblioteca/livros/livro/autor" e "livro/autor/nome" devem fazer parte da resposta.

Em sistemas que utilizam um caminho XML como parâmetro de comparação, por exemplo, sistemas de consulta a documentos e a redes ponto-a-ponto, o resultado do algoritmo PathSim pode ser comparado a um limiar de similaridade para determinar se uma informação é relevante, ou o resultado pode ser mostrado ao usuário na resposta final como um score de relevância da informação. De maneira diferente, sistemas de integração XML, que comparam estruturas complexas, devem extrair os parâmetros de consulta, de um dos conjuntos de caminhos, para comparar aos caminhos do outro conjunto de caminhos. Portanto, a utilização dos valores de similaridade, nesses sistemas, pode ser efetuada através de um cálculo sobre os valores, para determinar o valor final de similaridade entre as estruturas.

Outro resultado deste trabalho é a definição de funções de pré-processamento, que simplificam os caminhos comparados pelo algoritmo PathSim. Essas funções, além de incrementar os resultados obtidos nas comparações entre caminhos utilizando o algoritmo PathSim, também podem ser utilizadas para melhorar os resultados de outros algoritmos que manipulam caminhos XML. As simplificações são efetuadas através da remoção de caracteres dos caminhos que possam atrapalhar o cálculo do valor final de similaridade. Dessa forma, as funções de pré-processamento tornam os resultados das comparações do algoritmo PathSim mais precisos, além de diminuir o custo computacional da comparação.

A similaridade de caminhos é utilizada em (DO; RAHM, 2002) como um casador híbrido, que propõe concatenar os nomes dos elementos da seqüência de uma modelagem e utilizar uma função de similaridade aplicada às concatenações, para determinar a semelhança entre os caminhos. A utilização do valor de similaridade entre caminhos de estruturas apresentam bons resultados nessa abordagem. Os algoritmos PathSim e PathSim$_C$ são comparados a essa abordagem na seção de experimentos. Os experimentos realizados validam empiricamente os algoritmos PathSim e PathSim$_C$ e as funções de pré-processamento.

Esta dissertação está organizada em cinco capítulos e dois apêndices.

O Capítulo 2 apresenta o estado da arte de algoritmos que propõem-se a mensurar a similaridade entre informações expressas em XML. Também são descritos algoritmos de similaridade utilizados como base para a proposta do algoritmo PathSim, métodos de

avaliação da eficácia de funções de similaridade e outros trabalhos importantes para o entendimento das decisões na definição do algoritmo.

O Capítulo 3 apresenta o algoritmo PathSim e as duas variantes acima mencionadas. Além das variantes, são explicados: (i) o conceito de caminho XML utilizado neste trabalho e (ii) como os caminhos são extraídos dos documentos XML. Também são apresentadas as funções de pré-processamento que otimizam os resultados obtidos pelos algoritmos de similaridade.

O Capítulo 4 apresenta os experimentos realizados, que mostram os ganhos obtidos com os algoritmos PathSim e PathSim$_C$. Dois experimentos são detalhados neste capítulo.

O Capítulo 5 apresenta a conclusão do trabalho e enumera alguns trabalhos futuros que podem ser realizados.

O Apêndice A apresenta os caminhos extraídos dos documentos XML, criados por alunos da UFRGS. Estes caminhos foram utilizados como parâmetros das consultas e como fonte de caminhos nos experimentos.

O Apêndice B apresenta tabelas com os resultados detalhados dos experimentos para cada medida de avaliação.

2 ESTADO DA ARTE

Comparações por similaridade entre informações contidas em documentos XML vêm da necessidade de encontrar-se representações diferentes de uma mesma informação em diversos documentos. Essas comparações são efetuadas por funções de similaridade, sendo que cada função pode ter um limiar associado que indica o valor limite para que as duas entradas da função sejam considerados equivalentes. Uma função de similaridade utiliza, em seu algoritmo, uma ou mais características do padrão XML (nome de elemento, conteúdo de elemento, estrutura de elemento) nas comparações que definem o valor de similaridade entre as entradas.

Na literatura, vários métodos foram propostos para efetuar a comparação entre elementos ou documentos XML. Alguns propõem utilizar a representação dos documentos como árvores, e computar a similaridade entre eles baseada na distância de edição entre as árvores (COBENA; ABITEBOUL; MARIAN, 2002; NIERMAN; JAGADISH, 2002; WANG; DEWITT; CAI, 2003). Outros trabalhos propõem calcular similaridade entre elementos XML através de um cálculo sobre valores que refletem semelhanças existentes entre os elementos-filho destes (DORNELES et al., 2004; MA; CHBEIR, 2005). Quando as aplicações limitam-se a efetuar comparações entre estruturas, existem propostas que utilizam TF-IDF e relaxamento de consultas (estruturas) para otimizar a procura de uma determinada informação estruturada em um documento XML (AMER-YAHIA et al., 2005). Estes trabalhos e outros conceitos, utilizados como base desta dissertação, são apresentados nas seções deste capítulo.

2.1 Funções de Similaridade

Uma função de similaridade é um algoritmo que compara dois ou mais itens, e realiza comparações e cálculos para medir numericamente a semelhança existente entre eles. A maioria das funções de similaridade recebe como entrada dois itens e retorna como resultado um valor real normalizado no intervalo $[0, 1]$, onde 0 representa a completa desigualdade entre as entradas e 1 significa que os itens são equivalentes para a função.

Para determinar o valor final de similaridade, uma função analisa certas características dos itens e computa um valor baseado nessa comparação. Dessa forma, duas propriedades podem ser associadas às funções de similaridade:

- Reflexão: O resultado da função de similaridade comparando um item com ele mesmo é sempre o valor máximo definido pela função (1 na normalização freqüentemente utilizada);
- Simetria: O valor computado de similaridade entre um item A e um item B é igual ao valor computado de similaridade entre o item B e o item A.

Duas funções de similaridade são as bases para o desenvolvimento dos algoritmos PathSim e PathSim$_A$. São elas, respectivamente, o algoritmo editDistance e o algoritmo de Sellers. Ambos são explicados nas próximas seções.

2.1.1 EditDistance

A função de distância de edição editDistance (LEVENSHTEIN, 1966) compara duas cadeias de caracteres, e determina o número mínimo de operações atômicas necessárias para transformar uma cadeia na outra. As operações computáveis nesse cálculo são: inserção, remoção e substituição de um caractere. A partir do resultado da função é possível calcular-se um valor de similaridade entre as cadeias de caracteres.

O algoritmo para calcular a distância de edição entre uma cadeia de caracteres A e uma cadeia de caracteres B, respectivamente com i e j caracteres, utilizando apenas as operações atômicas, é apresentado na figura 2.1.

Inicialmente, a matriz M é criada no passo 2 e tem a sua primeira linha e primeira coluna inicializada no passo 3. Os valores associados às demais células são calculados pelo mínimo entre três valores, que representam o custo das três operações atômicas. Os valores calculados pelos passos 4(a) e 4(b) são responsáveis por computar os custos das operações de inclusão e remoção de caracteres, enquanto que o valor calculado pelo passo 4(c) computa o custo da operação de substituição de caracteres. O cálculo dessa última operação é auxiliado pela função $sub(x, y)$ que retorna 0, se $x = y$, e 1, caso contrário.

1. Sejam A e B duas cadeias de caracteres tais que |A|= i e |B|= j
2. Criar uma matriz M com $i + 1$ linhas e $j + 1$ colunas
3. Inicializar a primeira linha e a primeira coluna
   (a) Primeira linha $M[0, q] = q$, para $0 \leq q \leq j$
   (b) Primeira coluna. $M[p, 0] = p$, para $1 \leq p \leq i$
4. Calcular o valor associado às demais células $M[p, q]$ da matriz através do mínimo entre:
   (a) A posição acima mais um. $(M[p - 1, q] + 1)$
   (b) A posição à esquerda mais um. $(M[p, q - 1] + 1)$
   (c) A posição na diagonal esquerda superior mais o custo de substituição do caractere $A[p]$ por $B[q]$
   $(M[p - 1, q - 1] + sub(A[p], B[q]))$

Figura 2.1: Algoritmo editDistance

Ao final da execução do passo 4, a matriz está completamente preenchida. Cada célula da matriz $M[x, y]$ armazena o número mínimo de operações atômicas necessárias para a seqüência dos x primeiros caracteres de A transformar-se na seqüência dos y primeiros caracteres de B, ou vice-versa. Portanto, o valor contido na célula $M[i, j]$, reflete o número mínimo de operações necessárias para a cadeia A transformar-se na cadeia B.

$$editDistance(A, B) = \frac{Max(i, j) - M[i, j]}{Max(i, j)} \ ou \ editDistance(A, B) = 1 - \frac{M[i, j]}{Max(i, j)} \qquad (2.1)$$

Para transformar o número de operações em uma medida de similaridade pode ser utilizada qualquer uma das fórmulas apresentadas 2.1. A fórmula a esquerda calcula a

similaridade final pela normalização do número de operações que não são necessárias para efetuar a transformação (máximo possível menos o mínimo necessário) no intervalo $[0, 1]$. A normalização é feita com divisão do número de operações que não são necessárias pelo máximo número de operações possíveis. A fórmula à direita é uma simplificação da fórmula a esquerda.

Na figura 2.2 é apresentdo um exemplo de preenchimento da matriz de custo do editDistance. No exemplo, está sendo calculada a distância de edição entre as cadeias "autor" e "autentico". A distância de edição calculada pelo algoritmo editDistance entre as cadeias de caracteres apresentada na célula $M[5, 9]$ são seis operações . Logo a similaridade final entre elas é de 0.333, obtida pelo cálculo $Sim(autor, autentico) = \frac{(9-6)}{9}$.

		a	u	t	e	n	t	i	c	o
	0	1	2	3	4	5	6	7	8	9
a	1	0	1	2	3	4	5	6	7	8
u	2	1	0	1	2	3	4	5	6	7
t	3	2	1	0	1	2	3	4	5	6
o	4	3	2	1	1	2	3	4	5	5
r	5	4	3	2	2	2	3	4	5	6

Figura 2.2: Exemplo de preenchimento da matriz de custo do algoritmo editDistance

A função de distância editDistance e o algoritmo de similaridade associado a ela são amplamente utilizados e apresentam excelentes resultados em aplicações que comparam cadeias de caracteres.

2.1.2 Algoritmo de Sellers

O algoritmo proposto em (SELLERS, 1980) tem como objetivo efetuar uma busca por uma seqüência de caracteres em outra seqüência de caracteres com, no máximo, k operações de diferença. Ou seja, sua proposta é determinar se uma cadeia de caracteres A é sub-cadeia de outra cadeia de caracteres B com, no máximo, k operações de diferença. O algoritmo de Sellers tem por base o algoritmo editDistance. No entanto, apresenta adaptações na matriz de custo M e na forma de sua interpretação. Este algoritmo é eficiente para efetuar buscas por padrões em seqüências genéticas.

A adaptação efetuada na matriz do algoritmo é relativa aos valores associados à primeira linha. Diferentemente do algoritmo editDistance, que inicializa a linha com valores seqüenciais iniciados em 0, o algoritmo de Sellers inicializa a matriz com toda a primeira linha com o valor 0. Essa alteração simples faz com que o cálculo do número mínimo de operações para efetuar a transformação, da cadeia A na cadeia B, ignore qualquer prefixo da cadeia B, ou seja, a alteração realiza um processo de *stemming* (ver Seção 2.5) de prefixo na cadeia B.

Como o objetivo do algoritmo é identificar subseqüências de B que sejam até k operações diferentes de A, além de ignorar prefixos de B, sufixos que aumentem a diferença também devem ser ignorados. Para tornar a influência dos sufixos de B nula e identificar quais subseqüências são até k operações diferentes de A, toda a última linha da matriz deve ser analisada. Se o valor na célula analisada for menor ou igual a k, a subseqüência que encerra-se naquela posição faz parte da lista de resposta.

Na figura 2.3 é exemplificado o preenchimento da matriz de custo do algoritmo de Sellers. No exemplo, a consulta da seqüência "abt" na seqüência "autaatbta" com, no

máximo uma operação de diferença, retorna três subseqüências como resposta, ou seja, três subconjuntos da cadeia de caracteres “autaatbta” podem ser extraídas de modo a corresponder a, no máximo, uma operação de diferença ao padrão imposto pela consulta “abt”. De fato as subseqüências “aut”, “aat” e “tbt” são os itens da lista de resposta a consulta. No entanto, se a consulta procurasse por subseqüências que não apresentassem erros, ou seja, zero operações necessárias para a subseqüências transformar-se no padrão consultado, nenhuma subseqüência seria retornada como resposta.

		a	u	t	a	a	t	b	t	a
	0	0	0	0	0	0	0	0	0	0
a	1	0	1	1	0	0	1	1	1	0
b	2	1	1	2	1	1	1	1	2	1
t	3	2	2	1	2	2	1	2	1	2

Figura 2.3: Exemplo de preenchimento da matriz de custo do algoritmo de Sellers

Para seqüências genéticas, o algoritmo de Sellers permite uma avaliação eficiente de possíveis alinhamentos entre duas seqüências. No entanto, seu comportamento não é adequado para comparar cadeias que representem palavras, pois a avaliação do alinhamento entre itens de uma cadeia é interessante quando cada elemento da cadeia tem um significado próprio, o que não acontece na formação das palavras. O algoritmo de Sellers é interessante para caminhos pois estes são formados por nomes dos elementos, que têm significado próprio. O algoritmo PathSim$_A$ tem como base o algoritmo de Sellers.

2.2 Similaridade em XML

Nesta seção são explicadas algumas abordagens para mensurar a similaridade de informações expressas em XML. As funções estão divididas em dois grupos. No primeiro, são apresentados algoritmos que, apesar de não objetivarem especificamente comparar informações em XML, podem efetuar comparações entre as estruturas dos documentos. O segundo grupo apresenta algoritmos voltados diretamente para a comparação de informações expressas em XML, ou seja, comparam estrutura e conteúdo.

2.2.1 Similaridade de Estrutura

Algoritmos de similaridade estrutural mensuram a semelhança entre as estruturas de documentos semi-estruturados ignorando o conteúdo textual neste cálculo. Os elementos desses documentos podem conter, além de informação textual, outros elementos e atributos. Os atributos podem ser transformados em elementos simples. Essa característica recursiva possibilita a extração de uma árvore da estrutura do documento XML. Nessa árvore, o nodo-raiz representa o elemento raiz, e os nodos-folha representam os elementos simples ou vazios. Todos os filhos de um nodo na árvore representam elementos contidos no elemento que o nodo representa. Portanto, o método de cálculo de similaridade entre elementos XML mais intuitivo é utilizar valores de similaridades calculados entre elementos-filho para aferir a similaridade entre os elementos-pai.

Utilizando este princípio, em (DORNELES et al., 2004) são apresentadas fórmulas para calcular a similaridade entre duas coleções de elementos, baseadas em valores de similaridade calculados entre elementos componentes dessas coleções. Três tipos de cole-

ções (tuplas, listas e conjuntos) têm suas fórmulas definidas pela média das similaridades entre os componentes. As fórmulas da similaridade entre tuplas, listas e conjuntos são apresentadas respectivamente nas fórmulas 2.2, 2.3 e 2.4. Nas fórmulas ε_x é um coleção de elementos sendo comparada, ε_x^y é o y-ésimo elemento da coleção, $\varepsilon_x^y.\eta$ é o nome do y-ésimo elemento da coleção, n é o número de elementos de ε_p e m é o número de elementos de ε_d. A comparação entre duas tuplas leva em conta que os nomes dos elementos comparados deve ser o mesmo e que não pode existir mais de um elemento com o mesmo nome em uma coleção. A comparação entre listas, além da igualdade de nomes, exige que os índices dos elementos sejam os mesmos nas coleções. Já a comparação entre conjuntos apenas exige que os nomes dos elementos sejam iguais.

$$tupleSim(\varepsilon_p, \varepsilon_d) = \frac{\Sigma_{\varepsilon_p^i.\eta=\varepsilon_d^j.\eta}(sim(\varepsilon_p^i, \varepsilon_d^j))}{max(m,n)} \tag{2.2}$$

$$listSim(\varepsilon_p, \varepsilon_d) = \frac{\Sigma_{\varepsilon_p^i.\eta=\varepsilon_d^j.\eta \ and \ i=j}(sim(\varepsilon_p^i, \varepsilon_d^j))}{max(m,n)} \tag{2.3}$$

$$setSim(\varepsilon_p, \varepsilon_d) = \frac{\Sigma_{\varepsilon_p^i.\eta=\varepsilon_d^j.\eta}(max(sim(\varepsilon_p^i, [\varepsilon_d^1, ..., \varepsilon_d^m])))}{max(m,n)} \tag{2.4}$$

Apesar da abordagem não ser unicamente direcionada à comparação de documentos XML, suas fórmulas podem ser utilizadas para calcular a similaridade entre elementos complexos de documentos XML e, conseqüentemente, entre estruturas ou documentos XML. Para medir a semelhança entre elementos onde a ordem dos elementos-filho é importante, pode-se utilizar o cálculo de similaridade entre listas (fórmula 2.3). Já para calcular a semelhança entre elementos quando a ordem dos elementos-filho não é relevante, pode-se utilizar o cálculo de similaridade entre conjuntos (fórmula 2.4).

Em um trabalho semelhante, (MA; CHBEIR, 2005) apresenta uma abordagem específica para elementos XML, na qual o cálculo da similaridade entre os elementos utiliza somente combinações dos valores de similaridade dos elementos-filho. Essa função recursiva utiliza algoritmos específicos para cada tipo de dado que os documentos XML podem armazenar para calcular a similaridade entre elementos-folha. A fórmula 2.5 apresenta a forma genérica do algoritmo que calcula similaridade entre elementos não-folha na abordagem, onde N_p é o número de tipos de elementos-filho de ε_p, e $Sim_{ele}^{C_i}(\varepsilon_p, \varepsilon_d)$ é a similaridade entre os elementos de ε_p e ε_d do tipo C_i.

$$Sim_{inst}^{\varepsilon}(\varepsilon_p, \varepsilon_d) = \frac{1}{N_p}\Sigma_{i=1}^{N} Sim_{ele}^{C_i}(\varepsilon_p, \varepsilon_d) \tag{2.5}$$

Em uma abordagem que se baseia na estrutura sendo comparada por mais níveis da hierarquia, (JOSHI et al., 2003) propôs medir a similaridade entre dois documentos da Internet que tenham estrutura definida por rótulos, baseada no conjunto de caminhos extraídos destes. Cada caminho dos documentos é composto pela lista de elementos que contêm o elemento raiz, um elemento folha e a lista de elementos percorrida entre os dois elementos. A similaridade final entre dois documentos é calculada pelo somatório das freqüências dos caminhos comuns a ambos os documentos, dividida pelo somatório das freqüências dos caminhos da união dos conjuntos, como representado na fórmula 2.6.

$$Sim(d_j, d_l) = \frac{\Sigma_{k=1,N}\ min(d_{jk}, d_{lk})}{\Sigma_{k=1,N}\ max(d_{jk}, d_{lk})} \tag{2.6}$$

Na fórmula 2.6, a função d_{jk} é calculada pela divisão da freqüência do caminho k no documento j pela máxima freqüência obtida no documento j. Portanto $min(d_{jk}, d_{lk})$ calcula a freqüência do caminho k de um documento que representa a intersecção dos documentos, enquanto que, $max(d_{jk}, d_{lk})$ calcula a freqüência do caminho k de um documento que representa a união dos documentos. A variável N é o número de elementos do conjunto resultante da união dos caminhos dos documentos d_j e d_l.

As abordagens de comparação entre coleções de elementos, comparação entre conjuntos dos elementos-filho e comparação entre conjuntos de caminhos apresentados nesta seção são restritas pois exigem que, para serem comparados, os elementos obedeçam a uma mesma definição, ou seja, os nomes dos elementos componentes devem ser os mesmos para serem comparados. As únicas diferenças suportadas por essas abordagens são quanto à cardinalidade das relações.

Um algoritmo de comparação por similaridade de caminhos extraídos de estruturas foi proposto em (DO; RAHM, 2002). Nessa abordagem, cada caminho associado a um objeto representa a seqüência dos objetos modelados para alcançar este. Portanto, para estruturas XML, cada elemento simples tem associado a ele um caminho contendo o nome dos elementos-pais e o próprio nome do elemento. A similaridade entre os caminhos é calculada por uma função que efetua a computação diretamente sobre as cadeias de caracteres que representam o caminho. Funções para comparar cadeias de caracteres, como editDistance (apresentado na Seção 2.1.1), Jaro (JARO, ????; COHEN; RAVIKUMAR; FIENBERG, 2003) e Soundex (COHEN; RAVIKUMAR; FIENBERG, 2003), ou específicas para a comparação entre grandes nomes como a *Name* proposta em (DO; RAHM, 2002) podem ser utilizadas para comparar os caminhos. Na abordagem proposta, é utilizada uma função de comparação de nomes. A proposta é genérica e possibilita efetuar comparações entre estruturas diferentes, por exemplo, a comparação de um documento XML com uma esquema ER. Portanto, não possui mecanismos específicos para características dos caminhos XML, apesar de funcionar para estes.

2.2.2 Diferenças entre Árvores

Um método bem discutido na literatura para computar similaridade entre informações contidas em documentos XML é mapear uma árvore para cada estrutura que se quer comparar, e definir a similaridade entre as estruturas através da utilização de métodos de comparação de árvores (ZHANG; SHASHA, 1989; KLEIN, 1998; WANG; DEWITT; CAI, 2003; COBENA; ABITEBOUL; MARIAN, 2002; WANG; ZHANG, 2001; ZHANG, 1996; VALIENTE, 2001; AL-EKRAM; ADMA; BAYSAL, 2005).

Alguns algoritmos determinam a diferença entre duas árvores calculando o número de operações sobre os nodos necessárias para transformar uma árvore na outra. Estes métodos são classificados como algoritmos de distância de edição de árvores (BILLE, ????). As principais operações computáveis nesses algoritmos são: inserção, remoção e substituição de um nodo (ZHANG; SHASHA, 1989; KLEIN, 1998). Outras abordagens propõem, além das operações anteriores, computar operações de inserção e remoção de sub-árvores (WANG; DEWITT; CAI, 2003). Em cada abordagem, são definidos os custos de todas as operações, assim podem ser aplicados diferentes pesos, de acordo com a influência da modificação causada pela operação.

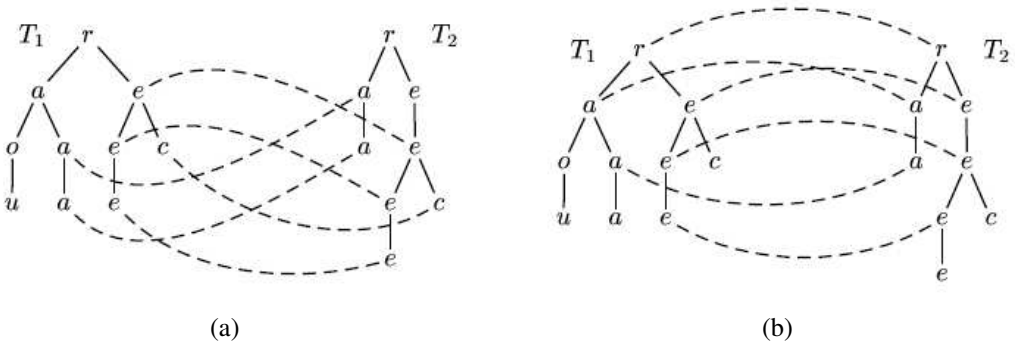
Um caso especial dos algoritmos de distância de edição de árvores é o estudo da inclusão de uma árvore em outra (KILPELäINEN; MANNILA, 1991). A única operação possível neste método é a remoção de nodos. Uma árvore está inclusa em outra se, com apenas operações de exclusão de nodos na segunda árvore, é possível se chegar na pri-

meira.

Alguns dos métodos de distância de edição entre árvores foram especificamente desenvolvidos para comparar árvores XML (WANG; DEWITT; CAI, 2003; COBENA; ABITEBOUL; MARIAN, 2002; MARIAN et al., 2001; AL-EKRAM; ADMA; BAYSAL, 2005). Essas abordagens são baseadas no número de operações necessárias para se transformar uma árvore XML na outra. O objetivo principal desses algoritmos é identificar a evolução de um documento com o passar do tempo. Uma operação específica para árvores XML foi definida em (WANG; DEWITT; CAI, 2003), nela é possível mover um nodo-filho de um nodo-pai para outro nodo-pai.

Apesar de ser muito eficaz teoricamente, a contagem do número de operações para se transformar uma árvore A em uma árvore B tem um custo computacional alto. Então, os métodos dividem-se entre os que encontram um resultado ótimo com alto custo computacional, e os que encontram um resultado de menor qualidade com custo computacional menor. Além disso, para documentos XML, os algoritmos de distância de edição de árvores apresentam outra grande desvantagem, pois penalizam muito a similaridade entre duas estruturas se apenas uma delas apresentar um alto nível de detalhamento de informações. Este caso ocorre quando o conteúdo de um elemento de uma representação está disposto em vários elementos da outra representação. Desta forma, a distância de edição entre árvores resulta num valor elevado, levando a um valor de similaridade baixo.

Outra abordagem para definir a semelhança entre duas árvores envolve efetuar a computação baseada no número de correspondências entre os nodos das árvores. A comparação entre os nodos é feita pelos rótulos (nomes dos elementos XML) dos mesmos. As comparações podem ser iniciadas nos nodos-folha, e subir nas árvores enquanto os nodos forem equivalentes, este método é nomeado *bottom-up* (VALIENTE, 2001). Na figura 2.4(a) são mostrados os seis mapeamentos obtidos pela abordagem *bottom-up* para a comparação entre as árvores $T1$ e $T2$. Outra possível seqüência de comparações é efetuar a primeira comparação entre os elementos raiz das árvores e, recursivamente, comparar os nodos-filho enquanto os nodos-pai forem iguais, este método é nomeado *top-down* (TANAKA; TANAKA, 1988). Na figura 2.4(b) são mostrados os seis mapeamentos obtidos com a abordagem *top-down* de mapeamento de elementos entre duas árvores $T1$ e $T2$. Apesar do número de mapeamentos identificados entre as árvores serem iguais nas abordagens, os mapeamentos são diferentes.



(a) (b)

Figura 2.4: Exemplo de mapeamentos feitos pelas medidas de distância entre árvores: (a) bottom-up e (b) top-down

Abordagens similares tentam definir a similaridade entre documentos baseada no alinhamento das árvores extraídas dos documentos XML (ZHAI; LIU, 2006; WANG; GUSFIELD, 1996). O alinhamento de árvores é feito com a inserção de nodos com nomes em

branco, em ambas as árvores, em posições onde os nodos não diferentes. Desta forma as árvores tornam-se isomórficas quando os rótulos dos nodos são ignorados. As árvores isomórficas são sobrepostas e o valor de transformação entre elas é calculado baseado nas diferenças entre os nomes dos nodos nas mesmas posições (BILLE, ????). Desta forma, após a sobreposição das árvores modificadas, os valores de desigualdade entre os elementos em posições equivalentes são manipulados para calcular a similaridade final, o cálculo mais simples soma todas as dissimilaridades entre os elementos. No entanto, essas funções de alinhamento não apresentam resultados precisos para documentos XML, pois não ocorre uma repetição de padrões significativa nos dados desses documentos, o que acarreta em valores de similaridade baixos para comparações de documentos com a mesma informação.

2.3 Utilizando Várias Funções para Melhorar os Resultados

Uma função de similaridade é desenvolvida para analisar determinadas características e retornar máximos resultados quando estas são iguais nas representações sendo comparadas. Entretanto, a variedade de maneiras disponíveis para se modelar uma informação torna difícil que uma função apresente bons resultados para todos os casos. Em um caminho inverso, alguns pares de representação apresentam bons resultados com a comparação de mais de uma característica e, portanto, mais de uma função de similaridade pode apresentar bons resultados na comparação. A combinação de resultados de várias funções de similaridade pode minimizar as deficiências de algumas funções em determinados ambientes, bem como otimizar os resultados das funções em ambientes que podem ser comparados por várias funções de similaridade.

Para sistemas de identificação de instâncias, (ZHAO; RAM, 2005) propôs alguns métodos para combinar vários classificadores para a definição de um resultado final entre duas entradas. Neste trabalho, um classificador é responsável por comparar duas instâncias e inferir se elas são equivalentes. Na figura 2.5, adaptada de (ZHAO; RAM, 2005), são mostrados os métodos propostos. Todos utilizam reconhecimento de padrões, aprendizado de máquina e redes de inteligência artificial.

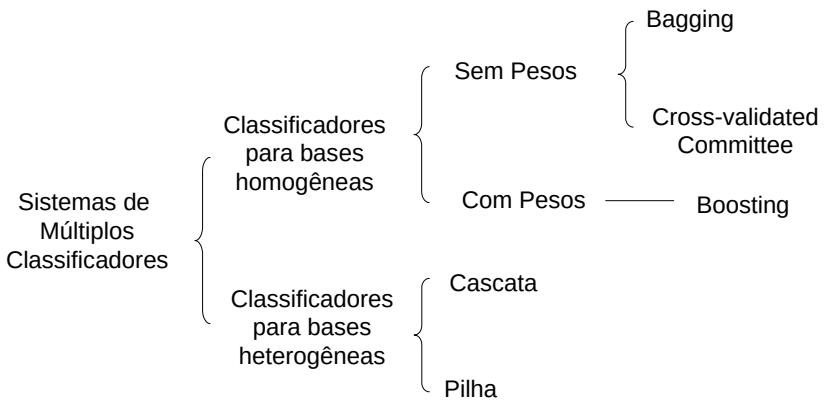


Figura 2.5: Classificação de métodos para combinar vários classificadores de entidades, adaptada de (ZHAO; RAM, 2005)

Nas abordagens para bases homogêneas, cada método chega a uma decisão, e as decisões de todos os métodos são agrupadas para definir a decisão final. Nas abordagens sem pesos, todos os métodos têm a mesma importância na tomada da decisão final, enquanto

que na abordagem que utiliza pesos, esses são definidos por uma medida de precisão calculada anteriorormente em cada método. “Bagging” e “Cross-validated Committee” diferenciam-se no método de escolha dos elementos de treinamento. Para o classificador, a primeira abordagem seleciona *n* elementos com reposição para cada método, já a segunda divide o conjunto de elementos do treinamento em grupos disjuntos.

Nas abordagens para bases heterogêneas ocorre uma interação entre os métodos, onde cada método subseqüente a ser treinado recebe os resultados do método anterior. A diferença entre a abordagem Cascata e Pilha é que, na primeira, os resultados são utilizados juntamente com os elementos a serem comparados, enquanto que, na segunda, apenas os resultados são utilizados no treinamento do próximo método.

Apesar de não serem direcionados à combinação de funções de similaridade, os algoritmos propostos mostram a importância do uso de múltiplos algoritmos para a tomada de decisão sobre a equivalência entre instâncias. Em algoritmos de similaridade, o conceito de combinação de funções pode ser utilizado de duas maneiras. A primeira é relativa a calcular a similaridade entre dois elementos utilizando mais de uma função. Já a segunda é relativa à utilização de funções de similaridade internamente nos algoritmos de outras funções de similaridade, por exemplo, a utilização de funções de similaridade que utilizam *thesaurus* (OGAWA; INUZUKA, 2004; KUKULENZ; HERGET; PAULI, 2005), ontologias (LIANG et al., 2006; WEN; JIANG; SHADBOLT, 2006) e dicionários de sinônimos (PLAS; TIEDEMANN, 2006), no algoritmo de uma função de similaridade que efetua a comparação estrutural de elementos.

2.4 Medidas de Avaliação de Funções de Similaridade

Várias medidas de avaliação de funções podem ser aplicadas para medir a eficácia de funções de similaridade. Cada medida tem um objetivo e foi projetada para avaliar as funções conforme determinada característica desejada (BAEZA-YATES; RIBEIRO-NETO, 1999). As medidas utilizadas nesta dissertação para avaliar as funções propostas são: Revocação, Precisão, Média Harmônica e a Medida F. Todas são explicadas nesta seção.

Para explicar as medidas nas próximas seções, na figura 2.6 é apresentado um esquema gráfico das possíveis classificações dos itens da fonte onde uma consulta está sendo efetuada. O retângulo representa todos os itens da fonte; a elipse à direita representa os itens relevantes para a consulta, ou seja, os itens que são resposta correta para a consulta; enquanto que, a elipse à esquerda representa os itens que a consulta retornou como resposta. A representação esquemática de uma função de similaridade perfeita tem o conjunto de retornados idêntico ao conjunto de relevantes, para todas as consultas.

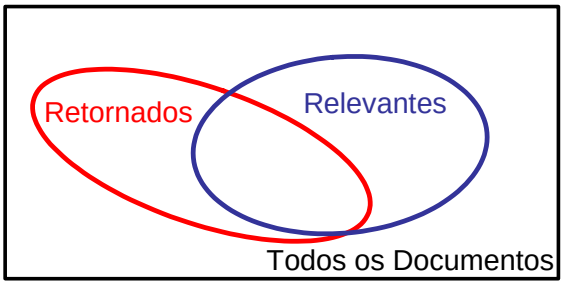


Figura 2.6: Classificação dos documentos de uma fonte em relação a uma consulta

2.4.1 Revocação

A revocação mensura a fração das respostas relevantes retornadas pela função. Portanto, o cálculo de seu valor é obtido pela divisão do número de respostas relevantes obtidas, pelo número total de respostas relevantes existentes, como mostra a fórmula 2.7.

$$Revocacão = \frac{Num(Retornados \bigcap Relevantes)}{Num(Relevantes)} \tag{2.7}$$

2.4.2 Precisão

A precisão mensura a taxa de acerto da função nos resultados retornados. Portanto, seu cálculo é efetuado pela divisão do número de respostas relevantes obtidas, pelo número total de respostas retornadas, como mostra a fórmula 2.8.

$$Precisão = \frac{Num(Retornados \bigcap Relevantes)}{Num(Retornados)} \tag{2.8}$$

A medida de precisão de uma consulta pode ser calculada em pontos específicos do ranking. Por exemplo, a precisão de uma consulta pode ser calculada somente considerando os cinco primeiros resultados da consulta. Na seção de experimentos, além da precisão das consultas, foram calculadas as precisões nos cinco, dez e quinze primeiros resultados.

2.4.3 Precisão Média

A precisão média calcula a média das precisões medidas ao se retornar cada resultado correto na lista de resposta.

$$PrecisãoMédia = \frac{\sum Precisões}{Num(Precisões)} \tag{2.9}$$

2.4.4 Média Harmônica

Analisar somente os valores de precisão ou revocação das consultas de uma função pode classificá-la de forma inadequada. Por exemplo, considerando uma consulta que tem cinco resultados relevantes em uma fonte, se uma função apresentar como resposta apenas dois resultados, ambos relevantes, a sua precisão será máxima. No entanto, três resultados não foram retornados. Essa ineficácia é computada pelo cálculo de revocação. De maneira análoga, se uma função retornar dez resultados, nos quais se incluem os cinco corretos, sua revocação será máxima, apesar da função ser imprecisa. Portanto, uma medida mais abrangente é necessária para mensurar a eficácia de uma função de similaridade.

A média harmônica de um conjunto de números reais positivos é calculada dividindo-se o número de membros do conjunto pela soma do inverso de cada membro (WIKIPEDIA, 2007). Por exemplo, a média harmônica entre os valores de precisão e revocação é calculada pela fórmula 2.10, e é uma alternativa para a avaliação de uma função de similaridade utilizando, simultaneamente, as medidas de precisão e revocação. Vale ressaltar que o uso da média harmônica no lugar da média aritmética faz o resultado final ter uma tendência para o menor valor, ou seja, a média harmônica é sempre menor que a média aritmética.

$$H(j) = \frac{2}{\frac{1}{r(j)} + \frac{1}{p(j)}} \tag{2.10}$$

O valor da média harmônica entre a precisão e a revocação de uma função também pode ser calculado em função dos conjuntos de respostas retornadas e relevantes. A fórmula 2.11 apresenta como este cálculo é efetuado. Essa fórmula é utilizada quando o valor da Medida F for calculado nos experimentos dessa dissertação, pois sua computação é mais simples e os resultados são idênticos.

$$H(j) = \frac{2 * Num(Retornados \bigcap Relevantes)}{Num(Retornados) + Num(Relevantes)} \tag{2.11}$$

2.4.5 Medida F

A medida F é uma variação da média harmônica, onde podem ser definidos os pesos dos valores de precisão e revocação no resultado final (VAN RIJSBERGEN, 1979). O peso x aplicado à fórmula 2.12, pode ser qualquer valor real positivo. Se o valor estiver no intervalo $[0, 1]$, o valor de revocação será mais valorizado, enquanto que se o valor de x for acima de 1, o valor mais valorizado será o da precisão.

$$F(j) = \frac{(1 + x) * p(j) * r(j)}{(x * p(j) + r(j))} \tag{2.12}$$

Alternativamente, a fórmula 2.13 calcula a média harmônica parametrizada entre os valores de precisão e revocação baseada nos conjuntos de valores retornados e relevantes. Entretanto, neste cálculo o valor de x está compreendido somente no intervalo $[0, 1]$ e reflete a importância dada ao valor da precisão da função. O complemento em 1 do valor de x é o peso atribuído ao valor da revocação. Essa fórmula será utilizada quando o valor da Medida F for calculada nos experimentos dessa dissertação, pois sua computação é mais simples e os resultados são idênticos.

$$F(j) = \frac{Num(Retornados \bigcap Relevantes)}{x * Num(Retornados) + (1 - x) * Num(Relevantes)} \tag{2.13}$$

2.5 Algoritmos de *Stemming*

Stemming é o processo de redução do tamanho de palavras, para torná-las mais simples e conseqüentemente facilitar o processo de comparação (JONES; WILLETT, 1997; ORENGO; HUYCK, 2001). Um *stem* é a parte que resta de uma palavra quando são retirados os afixos. Portanto, os algoritmos de *stemming*, também conhecidos por "*stemmers*" ou "*lemmatizers*", visam remover qualquer caractere que seja oriundo da formação da palavra, para deixá-la somente com sua raiz. Um algoritmo de *stemming* faz com que todas as variações morfológicas e sintáticas de uma palavra sejam representadas igualmente (CHAVES, 2003).

Entre os principais algoritmos de *stemming* encontram-se os algoritmos de força bruta, os de remoção de afixos e os estocásticos. Os algoritmos de força bruta mantêm uma tabela com a relação entre os termos e seus respectivos *stems*, onde uma consulta à tabela é efetuada para se identificar o *stem* de um termo. Os algoritmos de remoção de afixos têm regras para remoção de caracteres dos termos para formar o *stem*. Dependendo da posição de onde os caracteres são removidos, a ação pode ser classificada como remoção de prefixo ou remoção de sufixo. Já os algoritmos estocásticos são treinados com relações entre termos e *stems* e desenvolvem um modelo probabilístico com regras lingüísticas semelhantes à dos algoritmos de remoção de afixos.

Os algoritmos de *stemming* são utilizados principalmente em sistemas de Recuperação de Informação para melhorar a abrangência dos resultados de uma busca, pois tornam variantes morfológicas de uma palavra equivalentes. Um outro benefício, oriundo do uso de algoritmos de *stemming*, é a redução no tamanho dos arquivos de índice. As principais aplicações que utilizam *stemmers* são máquinas de busca, como por exemplo *Lycos* (LYCOS, 2007) e *Google* (GOOGLE, 2007), *thesaurus* e aplicações que utilizam processamento de linguagem natural para recuperação de informação.

Dependendo da aplicação onde a função de similaridade é utilizada, o auxílio por algoritmos de *stemming* pode melhorar seus resultados finais. O ganho é relativo à maior abrangência que uma consulta pode alcançar com a redução dos termos da consulta e da fonte a *stems*.

2.6 Conclusão

Neste capítulo foram apresentados algoritmos de comparação de informações expressas em XML. Os algoritmos foram divididos em dois grupos. O primeiro apresenta as abordagens que comparam apenas a estrutura dos documentos XML. Já o segundo grupo apresenta as abordagens que utilizam dados relativos à estrutura e ao conteúdo textual para definir o valor de similaridade entre dois documentos XML. Por serem baseadas em comparações por igualdade de nomes dos elementos, as abordagens apresentadas são restritas.

Também foram apresentados neste capítulo algoritmos de similaridade que são a base para o desenvolvimento do algoritmo PathSim, medidas para a avaliação de funções de similaridade, que são utilizados para definir o ganho da abordagem, e uma explanação sobre o *stemming* de palavras que é base para uma das funções de pré-processamento.

3 ALGORITMO PROPOSTO

O algoritmo para calcular o valor de similaridade entre dois caminhos XML proposto por esse trabalho é denominado PathSim. PathSim recebe dois caminhos como entrada e computa como resultado um valor de similaridade entre os caminhos. O resultado é normalizado no intervalo [0,1], onde 0 indica desigualdade completa entre os caminhos, enquanto que 1 representa que as entradas são equivalentes para a função.

O algoritmo PathSim tem por base o algoritmo editDistance (ver Seção 2.1.1), apresentando como diferencial o suporte a comparações por similaridade entre nomes de elementos, ao contrário de comparações por igualdade entre caracteres como no original. O algoritmo editDistance foi utilizado como base por ser uma função de distância de edição eficaz para computar o número de operações sobre caracteres que diferenciam duas palavras. Este número de operações possibilita calcular um valor de similaridade entre palavras. Visto que existe uma analogia entre cadeias de caracteres das palavras e seqüências de nomes de elementos dos caminhos, e o algoritmo editDistance calcula, com eficácia, a similaridade entre duas cadeias de caracteres; um algoritmo eficaz para mensurar a distância de edição entre dois caminhos e, posteriormente, definir a similaridade entre eles, pode ser definido baseado no algoritmo editDistance.

O algoritmo PathSim apresenta as seguintes diferenças em relação ao algoritmo proposto por Levenshtein:

- O algoritmo PathSim compara caminhos XML, enquanto que o algoritmo editDistance compara cadeias de caracteres;
- O algoritmo PathSim utiliza comparações entre nomes de elementos dos caminhos para determinar o resultado final, enquanto que o algoritmo editDistance utiliza comparações entre caracteres das cadeias;
- No algoritmo PathSim, as comparações entre nomes de elementos podem ser efetuadas por qualquer função de similaridade, enquanto que no algoritmo editDistance, as comparações são efetuadas unicamente por igualdade;
- No algoritmo PathSim, valores reais no intervalo [0,1], calculados pela função de similaridade entre nomes de elementos, são utilizados para mensurar o custo da operação de substituição de um nome de elemento, enquanto que no algoritmo editDistance, apenas os valores inteiros 0 e 1 podem ser utilizados como custo da operação de substituição.

Na figura 3.1 são apresentados os fluxogramas de obtenção de caminhos comparados pelo algoritmo PathSim, a partir de documentos XML apresentado na figura 3.1(a) e,

a partir de caminhos XML apresentado na figura 3.1(b). Quando se deseja extrair um caminho a partir de um documentos XML, este, se necessário, deve ser alterado para um estado simplificado. Após a simplificação do documento, um algoritmo de extração de caminhos gera os caminhos que serão entradas da fase de pré-processamento. Nessa etapa, o caminho é simplificado pelas funções de pré-processamento. Depois de realizado o pré-processamento, os caminhos podem ser comparados pelo algoritmo PathSim. Um arquivo XML gera um caminho para cada atributo, elemento vazio ou elemento com valor textual. Somente um caminho extraído do documento XML é utilizado como entrada do algoritmo PathSim.

Quando se deseja utilizar caminhos XML como entrada, o processo limita-se à aplicação das funções de pré-processamento, onde o caminho é simplificado. Após a simplificação, o caminho pode ser comparado a outro caminho pelo algoritmo PathSim. Todas essas etapas são explicadas neste capítulo.

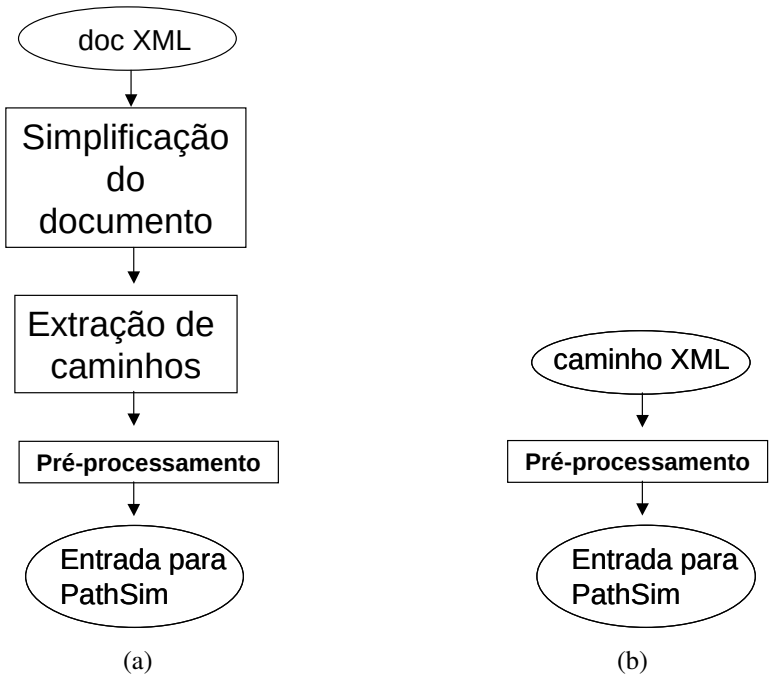


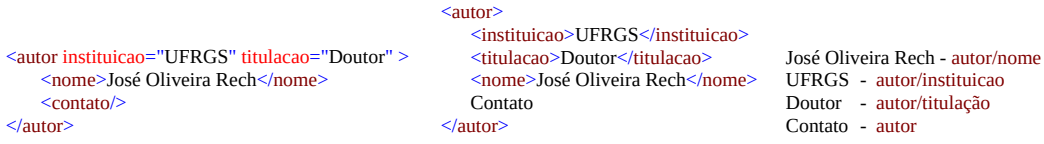
Figura 3.1: Fluxograma de obtenção de caminhos para o algoritmo PathSim: (a) a partir de um documento XML e (b) a partir de um caminho XML

Este capítulo está dividido em cinco seções. A seção 3.1 explica a definição de caminho XML utilizada por este trabalho, e como é efetuada a extração dos caminhos de um documento XML. A seção 3.2 apresenta o algoritmo PathSim detalhado e um exemplo do cálculo de similaridade. A seção 3.3 apresenta a variante PathSim$_C$, que possibilita comparações entre combinações de nomes de vários elementos dos caminhos. A seção 3.4 mostra a variante PathSim$_A$, que possibilita a utilização de técnicas de alinhamento de cadeias de caracteres aplicadas às seqüências de nomes de elementos. A seção 3.5 apresenta três funções de pré-processamento, que podem ser utilizadas para incrementar os resultados do algoritmo PathSim. A seção 3.6 apresenta um resumo desse capítulo e algumas considerações finais sobre o algoritmo.

3.1 Extração de caminhos XML

Informalmente, um caminho corresponde ao percurso, ou à seqüência de pontos, percorrido para se locomover de um lugar a outro. De maneira semelhante, em documentos XML, o caminho de um determinado elemento corresponde à seqüência de elementos percorridos do elemento raiz do documento até o próprio elemento, incluindo o elementos raiz e o próprio elemento. O caminho de um elemento XML representa a estrutura hierárquica que caracteriza a informação que este contém. Deste modo, para um elemento com conteúdo textual, o caminho XML é uma importante propriedade para especificá-lo e auxiliar um processo que objetiva integrar essa informação.

Nem todas as informações de um documento XML estão representadas diretamente em conteúdos textuais de elementos. Alguns documentos XML apresentam informações dispostas em elementos vazios ou atributos. O documento XML apresentado na figura 3.2(a) exemplifica ambas as situações, pois contém um elemento vazio, o elemento "contato", e dois conteúdos textuais contido em atributos, os conteúdos "UFRGS" e "Doutor". Para estas informações, a extração de caminhos não se aplica diretamente, pois o nome do atributo não faz parte do caminho, e o caminho XML de um elemento vazio não caracteriza nenhuma informação textual. Portanto, para obter-se a extração completa dos caminhos de um documento, este deve estar em um formato simplificado. Entende-se por documento em formato simplificado, aquele em que todas as informações estão contidas em conteúdos de elementos. Na figura 3.2(b) é mostrado um exemplo de um documento XML simplificado. Para possibilitar a extração completa de caminhos, documentos que contém atributos ou elementos vazios devem passar por um processo de transformação. Neste processo, as informações contidas nessas estruturas passam a ser representadas por elementos e conteúdos textuais. Na figura 3.2(b) é apresentado a versão simplificada do XML do documento mostrado na figura 3.2(a).



(a) (b) (c)

Figura 3.2: Exemplo de extração de caminhos de um documento XML: (a) Documento XML original, (b) Documento XML no formato simplificado e (c) caminhos extraídos do documento XML

A transformação de um atributo é efetuada com a criação de um elemento com o nome do atributo. Este elemento será filho do elemento ao qual o atributo pertencia, e seu conteúdo textual será o valor do atributo. A transformação dos atributos "instituicao" e "titulacao" do documento apresentado na figura 3.2(a) cria, respectivamente, os elementos "instituicao" e "titulacao" do documento apresentado na figura 3.2(b).

A transformação de um elemento vazio é feita atribuindo o nome do elemento vazio ao conteúdo (direto ou indireto) do seu elemento-pai. Chamamos de conteúdo direto, se o nome do elemento vazio tornar-se conteúdo textual do elemento-pai, e indireto, se ele tornar-se conteúdo de outro elemento-filho do elemento-pai. A decisão entre conteúdo direto ou indireto depende da existência de conteúdo textual do elemento-pai. O nome do elemento vazio será colocado como conteúdo textual do elemento pai, se este for originalmente nulo, ou armazenado em outro elemento filho do elemento-pai, caso contrário. Na transformação do documento XML apresentado na figura 3.2(a) para o documento

mostrado na figura 3.2(b), o elemento vazio “contato” tornou-se conteúdo do elemento “autor”, exemplificando a transformação para conteúdo direto do elemento-pai.

A extração do caminho de um elemento corresponde a fazer o caminhamento do elemento raiz a este elemento, e armazenar, seqüencialmente, os nomes dos elementos transpassados separados por uma barra “/”. Na figura 3.2(c) são mostrados os caminhos extraídos para todos os elementos textuais do documento 3.2(b). Toda informação textual de um documento tem um caminho associado a ele, e, por este caminho ser uma propriedade importante na caracterização da informação, comparações entre caminhos são importantes para algoritmos de comparação e integração de documentos XML.

3.2 Algoritmo Principal - PathSim

O algoritmo de comparação de caminhos PathSim tem como base o algoritmo editDistance que compara cadeias de caracteres. A função que efetua o cálculo de similaridade é baseada no número de operações de inclusão, remoção e substituição de nomes de elementos necessárias para transformar-se um caminho no outro. Uma matriz de propagação de custos é utilizada como estrutura de armazenamento dos valores parciais calculados. Esta matriz é a base da programação dinâmica utilizada na computação do custo de transformação. A programação dinâmica utilizada no algoritmo ocorre pelo uso de valores de células calculadas em passos anteriores no cálculo dos valores de outras células.

O algoritmo apresentado na figura 3.3 detalha a computação efetuada pelo algoritmo PathSim para a computação da distância de edição entre dois caminhos fictícios A e B compostos, respectivamente, por i e j nomes de elementos.

1. Criar uma matriz M com $i + 1$ linhas e $j + 1$ colunas
2. Inicializar a primeira linha e a primeira coluna
   (a) Primeira linha $M[0, q] = q$, para $0 \leq q \leq j$
   (b) Primeira coluna. $M[p, 0] = p$, para $1 \leq p \leq i$
3. Calcular o valor para as demais células $M[p, q]$ da matriz através do mínimo entre:
   (a) A posição acima mais um. $(M[p - 1, q] + 1)$
   (b) A posição à esquerda mais um. $(M[p, q - 1] + 1)$
   (c) A posição na diagonal esquerda superior mais a dissimilaridade entre os nomes dos elementos correspondentes àquela posição.
   $(M[p - 1, q - 1] + (1 - Sim(A[p], B[q])))$

Figura 3.3: Algoritmo PathSim

Inicialmente, o algoritmo PathSim cria, no passo 1, e inicializa, no passo 2, a matriz que será usada para armazenar e propagar os custos das operações de transformar um caminho no outro. De maneira análoga ao algoritmo editDistance, cada célula da matriz armazena o custo de transformar um sub-caminho de uma entrada em um sub-caminho da outra entrada. Por exemplo, a célula $M[2, 3]$ armazena o custo de transformar o sub-caminho formado pelos dois primeiros nomes de elementos do caminho A no sub-caminho composto pelos três primeiros nomes de elementos do caminho B. Portanto, o

custo final, ou seja, o custo de transformar o caminho A no caminho B, ou vice-versa, é armazenado na última coluna da última linha da matriz ($M[i, j]$).

Os valores associados às células que não pertencem à primeira linha ou à primeira coluna da matriz, ou seja, as que não foram inicializadas no passo 2, são calculados a partir de valores associados a algumas células vizinhas (superior, esquerda e diagonal superior esquerda), acrescidas do custo da operação correspondente (inserção, remoção ou substituição de um nome de elemento).

Os custos das operações de inserção e remoção de nome de elementos são iguais aos utilizados pelo algoritmo editDistance, ou seja, o valor associado à utilização de uma dessas operações é calculado pela adição do valor 1 ao valor da célula correspondente na matriz de custos (passos 3(a) e 3(b)). As operações de inserção são calculadas pelo passo 3(b), se for uma inserção de um nome de elemento no caminho A em relação ao caminho B, e 3(a), se for uma inserção de um nome de elemento no caminho B em relação ao caminho A. Já as operações de remoção são calculadas pelo passo 3(a), se for uma remoção de um nome de elemento do caminho A em relação ao caminho B, e 3(b), se for uma remoção de um nome de elemento do caminho B em relação ao caminho A. Essa distinção é importante para o entendimento do algoritmo, mas não influencia no cálculo do custo final; pois ambas as operações têm o mesmo custo, e o objetivo da matriz é computar quantas operações são necessárias para se transformar um caminho no outro. Portanto, é suficiente para um bom entendimento do algoritmo saber que os passos 3(a) e 3(b) são responsáveis por mensurar os custos das operações de inserção e remoção de nomes de elementos.

Diferentemente do cálculo dos custos das operações de inserção e remoção de um nome de elemento, o cálculo do custo da operação de substituição de um nome de elemento não é igual ao apresentado pelo algoritmo editDistance. No algoritmo editDistance, o custo de uma operação de substituição é computado como 0, se os itens comparados forem iguais, ou 1, se forem diferentes. No algoritmo PathSim, este valor é definido em função de uma métrica de similaridade entre os nomes de elementos comparados. O passo 3(c) apresenta o cálculo do custo da operação de substituição efetuado pela adição do valor de dissimilaridade entre os nomes de elementos correspondentes à célula da matriz de custos. Desta forma, o algoritmo PathSim possibilita utilizar todos os valores reais no intervalo $[0, 1]$, e não apenas suas extremidades, para definir o custo de uma operação de substituição. O objetivo de utilizar esse intervalo é possibilitar a representação de uma substituição parcial. Uma substituição parcial significa que os nomes dos elementos são diferentes, como sugere a operação de substituição no algoritmo editDistance, mas não totalmente. A utilização de um valor que reflete essa semelhança entre os nomes de elementos auxilia a expressar com mais exatidão a similaridade entre os caminhos. Qualquer função de similaridade entre nomes de elementos pode ser utilizada para computar o custo da operação de substituição, incluindo funções que utilizem *thesaurus*, ontologias e dicionários de sinônimos; bem como, a combinação de várias funções, essas se mostram particularmente interessantes por maximizar os resultados das funções como mostrado na seção 2.3.

Após o total preenchimento da matriz de custos, a similaridade final entre os caminhos A e B é calculada pela fórmula 3.1. Esta fórmula é análoga à utilizada no cálculo da similaridade final do algoritmo editDistance. A fórmula que calcula a similaridade entre os caminhos A e B normaliza, no intervalo $[0, 1]$, a medida do máximo número de operações possíveis para transformar um caminho no outro (o número máximo de elementos nos caminhos), diminuído do número mínimo de operações necessárias para efetuar a mesma

transformação (calculado pela matriz, representado na última coluna da última linha, ou seja, $M[i, j]$). A normalização é feita pela divisão do valor calculado na subtração, pelo máximo número de operações possíveis para transformar um caminho no outro.

$$Sim(A, B) = \frac{Max(i, j) - M[i, j]}{Max(i, j)} \tag{3.1}$$

Semelhanças existentes entre nomes de elementos de dois caminhos diferentes auxiliam no cálculo do valor de similaridade entre os caminhos. Isto acontece pela melhora na precisão do cálculo do custo da operação de substituição de nomes de elementos. No entanto, valores reduzidos de similaridade, aferidos por pares de nomes de elementos que são bastante diferentes, influenciam indevidamente na propagação de custos na matriz, prejudicando a precisão do resultado final. Por exemplo, utilizando a métrica editDistance para comparar os nomes dos elementos, o valor de similaridade entre os nomes de elementos "carro" e " livro" é de 0.40, no entanto, essa semelhança não deve influenciar no cálculo da similaridade entre dois caminhos que contenham esses nomes de elementos. Apesar da similaridade entre os nomes não ser nula (no exemplo a similaridade é de 0.40), a semelhança entre os nomes é irrelevante para determinar a similaridade entre os caminhos. Logo, essa similaridade deve ser ignorada no cálculo do valor final, ou seja, ao comparar dois nomes de elementos pouco similares o algoritmo não deve mensurar o valor da operação de substituição pelo valor da similaridade, computando uma operação completa de custo 1. Para tanto, um limiar de similaridade entre os nomes de elementos deve ser definido. Desta forma, uma operação de substituição pode assumir custos no intervalo $[0, 1 - limiar]$, caso a similaridade entre os nomes de elementos ultrapasse o valor do limiar, ou custo 1, caso contrário.

A complexidade do algoritmo PathSim é diretamente ligada a complexidade da função de similaridade utilizada para comparar os nomes de elementos. A complexidade pessimista do algoritmo PathSim é $\Theta(n^2.m)$, onde n é o número máximo de nomes de elementos entre A e B, e M é a complexidade da função de similaridade que compara os nomes de elementos.

3.2.1 Um Exemplo do Cálculo

Para exemplificar o cálculo de similaridade entre caminhos efetuado pelo algoritmo PathSim, esta seção apresenta, passo a passo, a computação do valor de similaridade entre os caminhos "Autor/Nomes" e "Autor/Nome/Filiacao". A função de similaridade utilizada para comparar nomes de elementos é a função editDistance, o limiar de similaridade é 0.70, e não são aplicadas funções de pré-processamento, pois, no exemplo, foram escolhidos caminhos simples que não precisam dessas funções para facilitar o entendimento do algoritmo PathSim. O limiar de similaridade utilizado foi definido pela observação de dados empíricos.

Visto que os caminhos têm, respectivamente, 2 e 3 nomes de elementos, a execução do passo 1 do algoritmo cria uma matriz de custo M com 3 linhas e 4 colunas, representada pela área mais escura da figura 3.4(a). Para implementar o passo 2, a primeira linha e a primeira coluna são inicializadas, fazendo com que a matriz fique como a da figura 3.4(b). Estas etapas são idênticas às etapas iniciais do algoritmo editDistance.

Para calcular o valor associado às demais células, como apresentado no passo 3 do algoritmo, é necessário calcular o mínimo entre três valores. São eles: o valor da célula à direita $+1$, o valor da célula à esquerda $+1$ e o valor da célula na diagonal direita superior + a dissimilaridade entre os nomes dos elementos daquela célula. Portanto, para

a célula $M[1,1]$, é o mínimo entre $(1+1,\ 1+1,\ 0+0)$, visto que os nomes de elementos "Autor" do primeiro caminho e "Autor" do segundo caminho são idênticos para a função editDistance. A análise da computação da célula $M[2,2]$ ressalta a diferença entre o algoritmo PathSim e o algoritmo editDistance, pois é efetuada pelo mínimo entre $(1+1,\ 1+1,\ 0+0.2)$, onde 0.2 é o custo de substituição do nome de elemento "Nome" por "Nomes", visto que a similaridade entre os nomes dos elementos é de 0.8. Se um algoritmo de comparação por igualdade fosse utilizado, como o proposto no editDistance, estes nomes de elementos seriam considerados diferentes, e o custo da operação de substituição seria de 1. No entanto, este valor da operação de substituição só pode ser utilizado porque a similaridade entre os nomes de elementos (0.8) é maior que o limiar de similaridade definido (0.7). Na figura 3.4(c) é mostrada a matriz M depois do cálculo do valor da célula $M[2,2]$.

		Autor	Nome	Filiacao
Autor				
Nomes				

(a)

		Autor	Nome	Filiacao
	0	1	2	3
Autor	1			
Nomes	2			

(b)

		Autor	Nome	Filiacao
	0	1	2	3
Autor	1	0	1	2
Nomes	2	1	0.2	

(c)

		Autor	Nome	Filiacao
	0	1	2	3
Autor	1	0	1	2
Nomes	2	1	0.2	1.2

(d)

Figura 3.4: Matriz de custo do algoritmo PathSim em várias etapas: (a)Vazia, (b) Inicializada, (c) Após o cálculo da posição $M[2,2]$ (d) totalmente preenchida

De acordo com o algoritmo PathSim, o custo de transformar um dos caminhos no outro é de 1.2 operações, como mostrado na célula $M[2,3]$, apresentado na figura 3.4(d). Visto que o máximo número de operações para efetuar a transformação é 3, o valor de similaridade entre os caminhos é de $PathSim = \frac{(3-1.2)}{3} = 0.6$, de acordo com a fórmula 3.1.

3.3 Variante com Comparação entre Combinações de Nomes de Elementos - PathSim$_C$

A fórmula apresentada para determinar o custo da operação de substituição do PathSim é baseada em comparações diretas de pares de elementos, um elemento de cada caminho. Entretanto, essa comparação pode não ser eficaz para determinar a similaridade entre dois caminhos oriundos de documentos com níveis de granularidade diferentes. Um elemento em um dos caminhos pode ter sua descrição dividida em vários elementos do outro caminho. Por exemplo, para comparar os caminhos "Livro/Autornome" e

"Livro/Autor/Nome", o algoritmo PathSim efetua cálculos de similaridade entre os elementos "autor" com "autornome" e "nome" com "autornome", os quais seriam ineficazes para representar a similaridade entre esses caminhos. Para auxiliar o algoritmo PathSim em casos deste tipo, comparações entre combinações de nomes de elementos são necessárias. A variante do algoritmo PathSim com comparações entre combinações de elementos é nomeada PathSim$_C$.

Efetuar uma comparação entre combinações de nomes de elementos corresponde a calcular a similaridade entre a combinação de m nomes de elementos de um caminho com n nomes de elementos do outro caminho. A função que efetua a comparação entre as combinações é definida na implementação desta funcionalidade. A fórmula para calcular o custo de uma operação de substituição com comparação entre combinações de elementos é apresentada na fórmula 3.2. Na fórmula, as variáveis m e n representam, respectivamente, o número de nomes de elementos dos caminhos A e B que são combinados antes da comparação pela função de similaridade. A instanciação desta fórmula para $m = 1$ e $n = 1$ representa a fórmula utilizada para o cálculo do custo da operação de substituição do PathSim (passo 3(c)).

$$M[p,q] = M[p-m, q-n] + max(m,n) * (1 - Sim(A[p-m+1] + .. + A[p]), (B[q-n+1] + .. + B[q]))) \tag{3.2}$$

Três partes da fórmula para o cálculo do custo de substituição com comparação entre combinações de nomes de elementos devem ser observadas em detalhes:

- $X[y-z+1] + .. + X[y]$, onde X, y e z podem ser A, p e m ou B, q e n; representa a combinação de todos $z-1$ nomes de elementos anteriores ao nome de elemento y do caminho X com o nome de elemento $X[y]$;

- $M[p-m, q-n]$ é utilizado para adicionar o custo de substituição da combinação de nome de elementos a um valor m células acima e n células a esquerda na matriz de custos;

- $max(m,n)$ é o multiplicador do valor da substituição dos elementos combinados. Este é necessário pois a utilização da comparação entre combinações de elementos dos caminhos pode substituir mais de uma operação necessária para efetuar a transformação entre os caminhos.

Figura 3.5: Exemplo de uma matriz de custos do algoritmo PathSim$_C$ que implementa a comparação com combinação de 2 elementos de um caminho com 1 elemento do outro caminho

Na figura 3.5 é mostrada a matriz M preenchida pela comparação entre os caminhos "Livro/Autornome" e "Livro/Autor/Nome" utilizando o algoritmo PathSim$_C$. Ao lado

da célula $M[2,3]$, aparece uma representação da instanciação da função que calcula o valor a ser associado a esta célula. Além dos cálculos de custo de inserção, remoção e substituição simples, o cálculo de uma comparação com combinação nomes de elementos para substituição usando $m = 1$ e $n = 2$ também é representado. A combinação de nomes de elementos instanciada no exemplo calcula o valor de uma substituição pela fórmula $M[2,3] = M[1,1] + max(1,2) * (1 - Sim(A[2], (B[2] + B[3])))$, onde a similaridade é calculada entre o nome de elemento "Autornome" e a combinação dos nomes de elementos "Autor" e "Nome" (marcados por elipses na figura 3.5). A utilização desta comparação diminui o número de operações necessárias para se efetuar a transformação de um caminho no outro, portanto, aumenta o valor de similaridade entre os caminhos, tornando o algoritmo PathSim$_C$ mais preciso que o algoritmo PathSim para este exemplo.

A complexidade do algoritmo $PathSim_C$ é diferente da complexidade do algoritmo PathSim e dependente dos valores de n e m adotados nas comparações múltiplas.

3.4 Variante com Técnicas de Alinhamento - PathSim$_A$

A comparação entre caminhos com o PathSim é eficaz quando a granularidade dos caminhos não apresenta diferenças expressivas. Para casos simples de diferença de granularidade com combinação de nomes de elementos, o algoritmo PathSim$_C$ é eficaz. No entanto, para aplicações onde são efetuadas consultas mais abrangentes, ou seja, onde se objetiva determinar a existência de uma informação independente da estrutura anterior na hierarquia, ou do nível de detalhamento utilizado para armazenar as informações, ambos os algoritmo apresentam restrições. O melhor exemplo deste tipo de ambiente é uma consulta a uma rede ponto-a-ponto, onde o usuário desconhece a estrutura dos documentos disponíveis e, portanto, também desconhece o nível de detalhe deles. Logo, existe a possibilidade do caminho informado pelo usuário e os caminhos extraídos dos documentos da rede apresentar níveis diferentes, de modo que o parâmetro esteja incluído no caminho extraído do documento da rede. O problema do algoritmo PathSim nesses sistemas ocorre devido à grande redução no valor das similaridades finais entre os caminhos. Isto acontece devido ao grande número de operações de inserção e remoção de nomes de elementos necessárias para fazer a transformação de um caminho no outro. Portanto, estas aplicações merecem um tratamento especial, através de uma variante do algoritmo para adequá-lo ao objetivo das aplicações e, assim, melhorar a precisão da função.

As diferenças nos níveis de detalhamento entre dois caminhos podem ter duas origens. São elas:

- Um melhor detalhamento das informações e, portanto, distribuição do conteúdo em mais elementos por um dos documentos que originam os caminhos;

- Uma diferença de escopo entre os arquivos que originaram os caminhos, um deles propondo-se a armazenar um escopo de informações que englobe o escopo do outro.

Visualmente, essas diferenças são identificadas por um número de nomes de elementos maior respectivamente no fim e no início dos caminhos. Por exemplo, o caminho B="Pais/estado/cidade/capital/habitante/masculino/nome" em relação ao caminho A="Cidades/habitantes/masculino" apresenta os dois casos de diferença. A diferença de nível de detalhamento é exemplificada pelo nome de elemento "nome" que o caminho B tem extra em relação ao caminho A, enquanto que a diferença no escopo é exemplificada pela seqüência de nomes de elementos "Pais/estado" que o caminho B apresenta a mais em relação ao caminho A.

As técnicas de alinhamento aplicam-se principalmente a sistemas que necessitam efetuar uma pesquisa em uma fonte de dados da qual não têm conhecimento da estrutura. O exemplo clássico dessas aplicações são as redes ponto-a-ponto que disponibilizam dados XML. Por exemplo, em uma rede ponto-a-ponto, um usuário efetua uma consulta informando o caminho "livro/autor", e esta retorna uma lista com os caminhos disponíveis que mais se assemelham à consulta. Entre os caminhos que compõem a resposta, alguns podem apresentar níveis de detalhamento diferentes, mas a mesma semântica da consulta, como os caminhos "catalogo/livro/autor" e "livro/autor/nome". Ambos os caminhos devem ser retornados com máximo valor de similaridade, pois eles contêm a informação requisitada pela consulta, apesar de terem nível de detalhamento ou escopos diferentes.

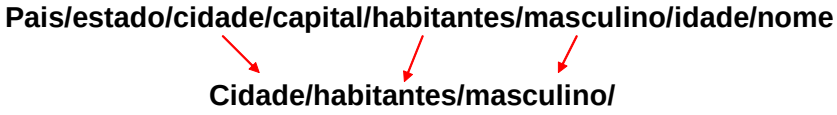


Figura 3.6: Exemplo de alinhamento de Caminhos

A variante PathSim$_A$ objetiva, com a utilização de técnicas de alinhamento de cadeias de caracteres, definir onde sub-seqüências de nomes de elementos do caminho mais extenso começam e terminam, de maneira a melhor assemelhar-se aos a seqüência de nomes de elementos do menor caminho. Dessa forma, o algoritmo PathSim$_A$ apresenta maiores valores finais de similaridade, quando comparado ao PathSim, por elimiar as diferenças nos níveis de detalhamento e de escopo dos caminhos. Na figura 3.6 são mostrados o caminho A alinhado ao caminho B juntamente com as correspondências entre os nomes de elementos de ambos os caminhos.

O algoritmo PathSim$_A$ tem por base o algoritmo de Sellers (ver Seção 2.1.2). Foram efetuadas as mesmas adaptações efetuadas no editDistance para a criação do PathSim. Desta forma, o PathSim$_A$ pode identificar sub-seqüências de nomes de elementos de um primeiro caminho e a quantidade de operações que são necessárias para transformar cada uma delas no segundo caminho. Portanto, o PathSim$_A$ pode procurar pela sub-seqüência de nomes de elementos que têm o menor número de operações diferente do segundo caminho. Desta forma, o algoritmo PathSim$_A$ alinha os caminhos e determina o melhor posicionamento entre os dois para que o resultado final seja o maior valor de similaridade possível. Visto que as comparações entre nomes de elementos são efetuadas por funções de similaridade, as operações de substituição são representadas por valores no intervalo $[0, 1]$, de maneira análoga ao PathSim.

A disposição dos caminhos na adaptação do algoritmo de Seller é determinante no cálculo do custo de transformação dos caminhos. As diferenças entre os algoritmos visam alinhar o caminho que é representado pelas linhas da matriz no caminho alocado nas colunas. No algoritmo $PathSim_A$, o caminho A será alinhado no caminho B, portanto o caminho B deve ser o de maior número de nomes de elementos para que o alinhamento seja efetivo.

$$Sim(A, B) = \frac{i - Min(M[i])}{i} \tag{3.3}$$

O cálculo de similaridade entre caminhos do algoritmo PathSim$_A$ tem uma diferença em relação ao algoritmo PathSim, visto que um dos caminhos é podado pelo algoritmo. O valor final de similaridade é calculado pela fórmula 3.3, onde i é o número de nomes de elementos do caminho A e $Min(M[i])$ é o menor custo de transformação da última linha.

		Pais	estado	cidade	capital	habitante	masculino	nome
	0	0	0	0	0	0	0	0
Cidades	1	1	1	0.15	1	1	1	1
Habitantes	2	2	2	1.15	1.15	1.1	2	2
Masculino	3	3	3	2.15	2.15	2.1	1.1	2.1

Figura 3.7: Exemplo de preenchimento da matriz de custos do algoritmo PathSim$_A$

Na figura 3.7 é apresentado um exemplo do preenchimento da matriz de custos do algoritmo PathSim$_A$. No exemplo, o algoritmo calcula o melhor alinhamento do caminho A="Cidades/habitantes/masculino" em relação ao caminho B="Pais/estado/cidade/capital/ habitante/masculino/nome". A primeira linha da matriz foi toda inicializada com o valor 0, como no algoritmo de Seller, fazendo com que o cálculo do número de operações ignorasse qualquer prefixo do caminho B que incrementasse o número de operações da transformação. Desta forma, o PathSim$_A$ faz um *stemming* de nomes de elementos prefixos. O cálculo da similaridade final entre os caminhos, efetuado com instanciação da fórmula 3.3, resulta em $PathSim_A = \frac{(3-1.1)}{3} = 0.63$.

A computação do menor valor da última linha ($Min(M[3])$) remove sufixos que podem aumentar o número de operações necessárias na transformação, diminuindo assim a similaridade final, desta forma, o PathSim$_A$ faz um *stemming* de nomes de elementos sufixos. No exemplo, a utilização do mínimo valor da última linha reduz em uma operação a diferença entre os caminhos, portanto, esta operação eliminou um nome de elemento sufixo do caminho maior aumentando a similaridade final.

A complexidade do algoritmo $PathSim_A$ é igual a complexidade do algoritmo PathSim. Portanto, a complexidade pessimista do algoritmo $PathSim_A$ é $\Theta(n^2.m)$, onde n é o número máximo de nomes de elementos entre A e B, e M é a complexidade da função de similaridade que compara os nomes de elementos.

3.5 Funções de Pré-Processamento

Esta seção detalha funções de pré-processamento que podem ser aplicadas aos caminhos antes do cálculo do valor de similaridade entre os mesmos. O objetivo dessas funções é simplificar os caminhos que serão comparados através da redução dos textos que os descrevem, e assim, facilitar o processo de comparação. As simplificações efetuadas pelas funções de pré-processamento não acarretam em perda de informações semânticas na descrição do caminho. As melhorias nos resultados das comparações são mostradas nos experimentos da seção 4 e podem ser observadas nas tabelas com os resultados detalhados dos experimentos apresentadas no apêndice B.

A principal motivação para o desenvolvimento das funções de pré-processamento é a grande quantidade de informações irrelevantes e duplicadas existentes em documentos XML. Essas informações são oriundas de práticas comuns no processo de modelagem de documentos XML. Entre essas práticas destacam-se:

- O uso de separadores entre termos, quando o nome do elemento é composto por mais de uma palavra (termo);
- A utilização, no nome de um elemento, de termos anteriormente associados a nomes de outros elementos;

- A utilização do plural ou coletivo de termos como nome de elementos que agrupam outros elementos.

Nas próximas seções, são descritas três funções de pré-processamento independentes que propõem minimizar os efeitos dessas práticas. São elas: Remoção de Caracteres Especiais, Remoção de Afixos e Remoção de Nomes de Elementos Duplicados.

3.5.1 Remoção de Caracteres Especiais

Uma prática comum na modelagem de documentos XML é, em um elemento cujo nome é composto por mais de um termo, os termos serem intercalados por separadores. Por exemplo, ao modelar um documento responsável por armazenar as informações sobre os livros de uma biblioteca, um analista pode nomear o elemento modelado para manter as informações referentes a títulos de livros como "titulo_livro" ou "titulolivro". Ambas as representações utilizam os mesmos termos, e estes apresentam-se na mesma ordem. No entanto, a utilização de um separador em uma das descrições pode diminuir desnecessariamente o valor calculado por uma função de similaridade.

Apesar de facilitar a leitura do documento XML por um ser humano, os separadores não têm significado semântico relevante para alguns algoritmos de similaridade. Assim, a utilização desses caracteres como parte da entrada para esses algoritmos de comparação pode prejudicá-los no cálculo do valor de similaridade. Portanto, é importante remover caracteres especiais dos nomes dos elementos componentes do caminho, para aprimorar as respostas das funções de similaridade.

Além do sublinhado "_", utilizado no exemplo, outros caracteres permitidos em nomes de elementos no padrão XML, como espaço em branco (" "), hífen ("-") e o sinal de dois pontos (":"); podem ser utilizados como separadores de termos.

O algoritmo de remoção de caracteres especiais recebe, como entrada, a lista de nomes de elementos que compõe o caminho, e retorna a mesma lista sem caracteres especiais nos nomes de elementos. Seu funcionamento consiste em comparar cada caracter dos nomes dos elementos com os itens de uma lista pré-definida de caracteres especiais e, se forem iguais, remover o caracter do nome do elemento.

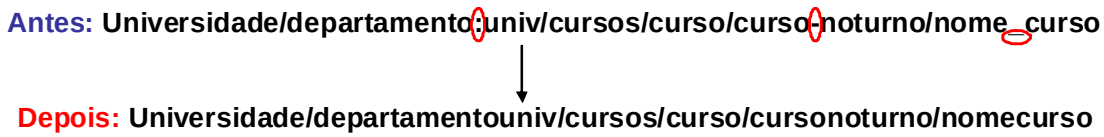


Figura 3.8: Exemplo de aplicação da função de remoção de caracteres especiais

Para exemplificar a utilização da função de remoção de caracteres especiais, na figura 3.8 é mostrado um caminho antes e depois da aplicação da função. A função remove os separadores marcados com elipses. São eles: o sinal de dois pontos do nome de elemento "departamento:univ", o hífen do nome de elemento "curso-noturno" e o sublinhado do nome de elemento "nome_curso". Portanto, três caracteres são removidos da representação original e, apesar dessa redução, a informação semântica é completamente mantida na nova representação do caminho.

3.5.2 Remoção de Afixos

Outra prática comum na modelagem de documentos XML é a utilização, no nome de um elemento, de parte ou do nome completo de um ou vários elementos que aparecem

anteriormente no caminho, ou seja, de elementos superiores hierarquicamente na estrutura do documento. Exemplificando no mesmo ambiente utilizado na função anterior, ao modelar um documento XML que mantém informações sobre os livros de uma biblioteca, se o analista utilizar a estrutura "biblioteca/acervo/livro/titulolivro", para armazenar o título do livro, o nome do último elemento ("titulolivro") contém o nome de um elemento anterior ("livro"). No exemplo, a informação de que o título se refere a um objeto livro está duplicada. Essa prática pode interferir em comparações de caminhos, quando uma função compara um caminho onde ela foi adotada com um caminho no qual ela não foi adotada.

Essa prática não se restringe a compor o final do nome de elementos. Utilizações similares podem compor o início ou a parte central do nome de elementos. Exemplos da utilização no início do nome de elementos são comuns em estruturas de documentos escritos no idioma Inglês. Por exemplo, "books" e "booktitle" são as traduções usuais dos nomes dos elementos do exemplo anterior em modelagens do mesmo domínio.

A função de pré-processamento proposta para minimizar esse problema restringe-se a remover afixos (prefixos e sufixos) dos nomes dos elementos, não atuando quando as partes oriundas dos nomes de outros elementos ocorrem no meio do nome de um elemento. Esta função atua como um algoritmo de *stemming* (ver Seção 2.5). Algoritmos de *stemming* de palavras removem afixos relativos à conjugação de verbos e formação de palavras. No entanto, a função de pré-processamento proposta somente elimina afixos referentes às informações já obtidas em outros nomes de elementos. Dessa forma, o objetivo dessa função é simplificar o caminho removendo informações duplicadas.

Apesar da remoção de afixos ser tratada como uma única função de pré-processamento, ela é composta por duas funções, uma para remover prefixos e outra para remover sufixos. As funções para remover prefixos e sufixos comparam os nomes de elementos de maneira semelhante. Ambas verificam, a partir do início do nome de um elemento anterior sendo comparado, qual a máxima cadeia de caracteres que coincide com a extremidade sendo reduzida. Desta forma, para um nome de elemento anterior $ElAnt$, cujos caracteres iniciais testados são representados pela variável $CaracIni$, e um nome de elemento El onde a seqüência de caracteres $CaracIni$ está sendo procurada, as expressões regulares 3.4 e 3.5 apresentam, respectivamente, as condições que devem ser satisfeitas por El para que um prefixo ou um sufixo sejam nele identificados. Se essas condições forem satisfeitas, a seqüência da variável $CaracIni$ pode ser removida do nome de elemento El.

$$(CaracIni)RestanteDeEl \tag{3.4}$$

$$InicioDeEl(CaracIni) \tag{3.5}$$

A função de remoção de afixos necessita da definição prévia de dois valores mínimos significativos: (i) O tamanho mínimo de uma cadeia de caracteres que forme um afixo significativo ($CaracIni$) e (ii) O tamanho mínimo de uma cadeia de caracteres que forme um nome de elemento representativo ($RestanteDeEl$ e $InicioDeEl$).

O primeiro mínimo significativo evita que sejam retiradas seqüências de caracteres que não são afixos significativos, as quais podem aparecer por coincidência. Por exemplo, dois nomes de elementos consecutivos podem ter os dois primeiros caracteres em comum, mas um não ser prefixo do outro, como a seqüência entre "barco" e "bandeira".

O segundo mínimo garante que o nome final do elemento seja significativo, ou seja, que não foram retirados caracteres demais de modo a descaracterizar o nome do elemento

(El). Visto que os afixos eliminados são oriundos dos nomes dos elementos anteriores, se esse limite não for definido para esta função, não existe como garantir que o resultado final seja a informação principal do nome do elemento. Por exemplo, sem a definição desse mínimo, na seqüência de elementos "barco" e "barcos" o segundo nome de elemento seria reduzido a "s" que não é significativo. Um problema semelhante é encontrado pelos algoritmos de *stemming* de palavras.

A definição desses limites é auxiliada pela definição dos limites mínimos de *stem* dos algoritmos de *stemming*. Esses consideram três caracteres como o tamanho mínimo para um *stem* ser relevante. Assim, a função de remoção de afixos é implementada com ambos limites em três caracteres.

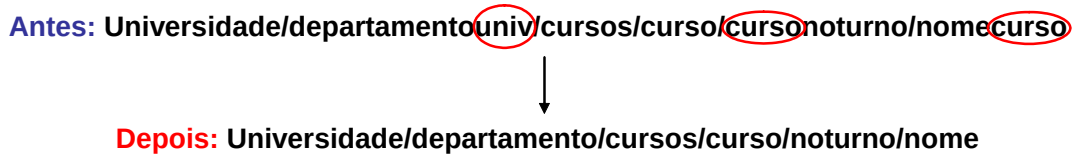


Figura 3.9: Exemplo de aplicação da função de remoção de afixos

Para exemplificar a aplicação da função de remoção de afixos, na figura 3.9 é mostrado um caminho antes e depois da aplicação da função. A função remove os afixos marcados com elipses. São eles: o sufixo "univ" do nome de elemento "departamentouniv", o prefixo curso do nome de elemento "cursonoturno" e o sufixo "curso" do nome de elemento "nomecurso". O mínimo significativo para o nome de um elemento previne que o nome de elemento "curso" seja excluído. De maneira análoga ao resultado da primeira função, ocorre uma diminuição no número de caracteres que representa o caminho. No exemplo, 14 caracteres foram subtraídos sem que a informação semântica fosse alterada.

3.5.3 Remoção de Nomes de Elementos Duplicados

Apesar de não ser exigido na modelagem de documentos XML, muitos analistas utilizam-se do artifício de criar um elemento de agrupamento para reunir estruturas repetidas. Por exemplo, com a utilização desse artifício na modelagem de documentos responsáveis por armazenar as informações sobre os livros de uma biblioteca, os autores, armazenados em elementos nomeados "autor", podem ser agrupados em um elemento nomeado "autores". Visto que elementos agrupadores são nomeados referindo-se ao conjunto de elementos que contêm, sua nomenclatura pode adicionar informação duplicada ao caminho, pois a informação pode estar expressa também no nome dos elementos contidos nos elementos agrupadores. No exemplo, essa prática faz com que o caminho do autor do livro tenha a informação de que este se trata de um autor duas vezes. Outras práticas também podem fazer com que alguns caminhos tenham um conceito representado por mais de um nome de elemento, portanto a remoção de alguns nomes de elementos não afeta o significado do caminho.

Mesmo permitido na modelagem de documentos XML, elementos agrupadores e os elementos que eles contêm não têm nomes idênticos. Portanto, uma função de similaridade deve ser usada para definir quando dois nomes de elementos representam o mesmo conceito, permitindo que um deles seja excluído. O valor do limiar para a exclusão dependerá do domínio do documento e da função de similaridade utilizada.

Outra definição necessária na implementação dessa função é qual nome de elemento deve ser excluído. Na implementação utilizada na seção de experimentos, esta função

elimina o nome de elemento com maior número de caracteres, visto que a formação do plural é feita freqüentemente pela adição de sufixos. Funções de similaridade que acessem dicionários, thesaurus ou ontologias específicos são necessários para identificar coletivos a serem removidos.

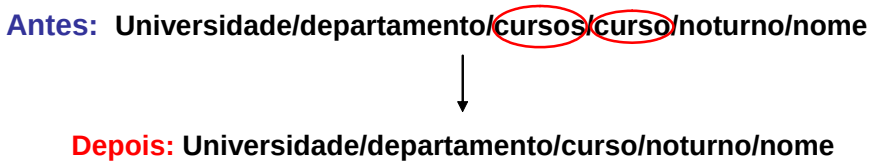


Figura 3.10: Exemplo de aplicação da função de remoção de nomes de elementos duplicados

O algoritmo de remoção de nomes de elementos duplicados resume-se a comparar, utilizando uma função de similaridade, os nomes de elementos do caminho entre si. Em cada comparação, se o resultado for superior ao limiar definido, a função remove um dos dois nomes de elementos, de acordo com a definição de qual nome de elemento deve ser removido.

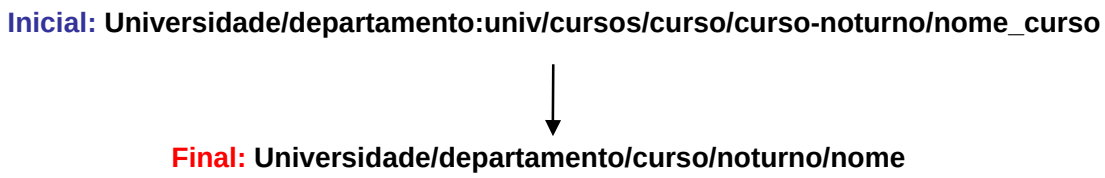


Figura 3.11: Exemplo de aplicação de todas funções de pré-processamento

Para exemplificar a aplicação da função de remoção de nomes de elementos duplicados, na figura 3.10 é mostrado um caminho antes e depois da aplicação da função. Os nomes de elementos marcados com elipses têm o mesmo significado semântico. A aplicação da função de pré-processamento remove o nome de elemento “cursos”. Novamente, ocorre uma diminuição no número de caracteres que representa o caminho. No exemplo, 6 caracteres são subtraídos, a informação sem que a informação semântica continua fosse alterada.

Na figura 3.11 é apresentado a representação inicial do caminho que utilizamos como exemplo e sua representação final após a aplicação das funções de pré-processamento. A aplicação das funções de pré-processamento subtraiu 23 caracteres da representação do caminho, o que corresponde a aproximadamente 36% da representação inicial. Essa diminuição do tamanho do caminho ocorreu sem haver perda de informação neste.

3.6 Considerações Finais

Neste capítulo, foram descritos em detalhes o algoritmo proposto e duas variações deste. O algoritmo nomeado PathSim é o algoritmo principal de similaridade entre dois caminhos, onde o cálculo de similaridade é efetuado baseado no número mínimo de operações necessárias para transformar um caminho no outro. A primeira variação explicada, nomeada PathSim$_C$, permite que sejam computadas diferenças entre combinações de nomes de elementos no cálculo do custo da operação de substituição, ou seja, se o nome de um elemento de um caminho for uma combinação dos nomes de alguns elementos

do outro caminho, o algoritmo PathSim$_C$ apresenta um valor de similaridade superior ao PathSim. A segunda variação apresentada, nomeada PathSim$_A$ permite o cálculo de similaridade entre caminhos com alinhamento dos mesmos.

Também foram descritas três funções de pré-processamento para caminhos e como os caminhos devem ser extraídos de documentos XML. As funções de pré-processamento simplificam os caminhos removendo caracteres especiais, afixos e nomes de elementos repetidos. O objetivo dessas funções é simplificar os caminhos para incrementar os resultados de algoritmos de comparação de caminhos.

O algoritmo PathSim e suas variantes podem efetuar as comparações entre nomes de elementos através de qualquer função de similaridade, inclusive funções que utilizem *thesaurus*, ontologias e dicionários. Também podem ser utilizadas combinações de funções. O valor de similaridade entre nomes de elementos calculado é utilizado para determinar a similaridade final entre os caminhos.

4 EXPERIMENTOS REALIZADOS

Esta seção apresenta os experimentos realizados para avaliar as três funções de pré-processamento e os algoritmos de similaridade entre caminhos PathSim e PathSim$_C$, no cálculo do valor de similaridade entre caminhos XML. O algoritmo PathSim$_A$ não vai ser avaliado nesses experimentos por dificuldade de montar a fonte de caminhos que simule consultas ponto-a-ponto e dificuldade em montar uma rede de usuário para avaliar o algoritmo.

Os experimentos simulam consultas a fontes de caminhos XML. Cada consulta tem um caminho como parâmetro, que representa a informação requerida pela consulta. As posições em que os caminhos corretos são retornados na lista de resposta são analisadas para avaliar a eficácia dos algoritmos.

No primeiro experimento, a fonte onde as consultas são executadas contém apenas caminhos XML do mesmo domínio do parâmetro usado na consulta, simulando consultas a fontes com domínios conhecidos, como as consultas efetuadas por sistemas de integração objetivando unificar as informações de documentos. Já no segundo experimento, além de caminhos do mesmo domínio do parâmetro consultado, também são incluídos caminhos de outros domínios, simulando consultas a fontes cujos domínios podem não ser conhecidos, como as consultas a redes ponto-a-ponto.

4.1 Parâmetros Utilizados

São testados nos experimentos o algoritmo PathSim, apresentado na seção 3.2, e a variante PathSim$_C$, apresentada na seção 3.3, que possibilita comparações entre combinações de nomes de elementos.

A base de comparação dos resultados obtidos pelos algoritmos nos experimentos é a função de similaridade entre caminhos de modelagens apresentada na seção 2.2.1. Esta abordagem propõe determinar a similaridade entre caminhos através da comparação entre as cadeias de caracteres diferentes de "/", extraídas dos caminhos. A função de similaridade utilizada para comparar as cadeias extraídas dos caminhos foi a função editDistance (ver Seção 2.1.1). No restante desse capítulo, a implementação desta abordagem será referida como "*baseline*".

Os parâmetros utilizados na implementação dos algoritmos PathSim e PathSim$_C$ são:

- A função de similaridade utilizada para efetuar a comparação entre nomes de elementos dos caminhos e determinar o custo de substituição de um nome de elemento pelo outro é a função editDistance;
- O limiar mínimo de similaridade entre nomes de elementos utilizado é 0.50. Este valor evita que similaridades baixas entre nomes de elementos interfiram indevi-

damente no valor final de similaridade entre os caminhos. Portanto, somente são computadas, como possíveis operações de substituição, as comparações entre nomes de elementos que superam 0.50. O limiar de similaridade utilizado foi definido pela observação de dados empíricos;

- As comparações entre combinações de nomes de elementos do PathSim$_C$ são efetuadas entre dois nomes de elementos de um caminho com um nome de elemento do outro caminho;

- A comparação entre combinações de nomes de elementos é efetuada pela função de similaridade Carla. Esta função foi escolhida por apresentar resultados mais precisos quando a seqüência dos nomes de elementos está invertida entre os caminhos (MERGEN; HEUSER, 2005).

A utilização da mesma função de similaridade para calcular a semelhança entre nomes de elementos nos algoritmos PathSim e PathSim$_C$, e para avaliar a similaridade final entre os caminhos no "*baseline*", faz com que os resultados obtidos expressem o ganho obtido pela abordagem de comparação de nomes de elementos do algoritmo PathSim.

As implementações dos algoritmos PathSim e PathSim$_C$ são comparadas com o algoritmo "*baseline*", utilizando as seguintes medidas:

- Média das precisões na análise dos 15, 10 e 5 primeiros resultados, e ao retornar o último resultado relevante;

- Relação Revocação *versus* Precisão;

- Média das revocações nas consultas;

- Média das médias harmônicas entre os valores de precisão e revocação das consultas;

- Média das medidas F, com pesos 0,2 e 0,8, respectivamente, para precisão e revocação;

- Média das precisão média por resultado correto retornado.

Quando necessário, a verificação da significância dos ganhos das abordagens em relação ao "*baseline*" é comprovada usando o teste estatístico *T-Student test* (PRESS et al., 1988).

4.2 Exemplo de Realização de uma Consulta nos Experimentos

Nos experimentos, realizar uma consulta é determinar a lista de caminhos da fonte de caminhos que têm similaridade com o caminho informado como parâmetro da consulta. Para efetuar uma consulta, é necessário calcular o valor de similaridade entre o parâmetro e cada caminho da fonte de caminhos. Todo caminho com valor de similaridade acima de 0, quando comparado ao caminho consultado, faz parte da resposta. Cada item da resposta é composto pelo caminho retornado e pelo valor de similaridade com o parâmetro. A lista de resposta é ordenada de forma decrescente pelo valor de similaridade. A tabela 4.1 exemplifica uma lista de resposta, onde a consulta que gerou esta lista tinha o caminho "empresa/funcionarios/funcionario/gerente/nome" como parâmetro.

Tabela 4.1: Resposta à uma consulta executada em uma fonte de caminhos XML

Valor de Similaridade	Caminho
0,85754985	empresa/funcionario/gerente/Pnome
0,75	loja/funcionarios/funcionario/gerente/nome
0,7037037	industria/funcionario/gerente/nome
0,5	emp/funcionarios/funcionario/nome
0,5	distribuidora/gerente/nome
0,476431	empresa/funcionario/presidente/sobrenome

Análises sobre a posição em que os caminhos corretos para a consulta aparecem na resposta avaliam a função de similaridade que gerou a mesma. Uma função de similaridade ideal retorna os caminhos relevantes para a consulta nos primeiras itens da resposta.

4.3 Experimento com Consultas a Fontes com Caminhos XML de um Domínio

Este experimento simula consultas a fontes que apenas contém caminhos do mesmo domínio do parâmetro consultado. Os principais exemplos de sistemas onde esta situação ocorre são sistemas de consultas a banco de dados XML e aplicações de integração de documentos XML. Em consultas a banco de dados XML, apesar de o usuário ter a possibilidade de utilizar funções de similaridade para consultar documentos sem conhecer sua estrutura, ele precisa indicar quais documentos deseja consultar e, portanto, conhece os arquivos existentes na base e o domínio a que eles pertencem. Já em sistema de integração de documentos XML, as informações de dois ou mais arquivos XML estão sendo integradas, logo estes arquivos devem pertencer ao mesmo domínio, e as comparações entre caminhos efetuadas neste sistema ocorrem entre caminhos do mesmo domínio.

4.3.1 Objetivo

O objetivo deste experimento é mostrar o ganho na utilização dos algoritmos PathSim e PathSim$_C$, em sistemas onde os itens da fonte de caminhos utilizada na consulta são todos do mesmo domínio do caminho consultado. Este ambiente é o mais genérico, pois engloba qualquer aplicação que se propõe a comparar duas estruturas XML do mesmo domínio.

4.3.2 Obtenção dos Caminhos Utilizados no Experimento

Para montar a fonte de caminhos, na qual as consultas serão efetuadas, foi solicitado aos alunos da disciplina de Fundamentos de Banco de Dados, do curso de Graduação em Ciência da Computação da UFRGS, que criassem um documento XML que contenha informações sobre artigos científicos. Dois exemplos textuais das informações a serem armazenadas nos arquivos foram apresentados aos alunos para auxiliá-los. Esses exemplos são apresentados na Figura 4.1.

Os alunos criaram treze documentos XML distintos. Esses documentos foram processados por um sistema de extração de caminhos, que simplifica os documentos e extrai os caminhos de acordo com a seção 3.1, gerando entre sete e quinze caminhos para cada documento. Na figura 4.2 são apresentados os caminhos extraídos de dois documentos. Todos os caminhos extraídos dos documentos criados pelos alunos são apresentados no

XX Simpósio Brasileiro de Banco de Dados - 2005
Uberlândia, MG, Brasil

Mapeamento de Definições XML Schema para SQL:1999.
Patrícia Martins, Alberto H. F. Laender.
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
31270-901 – Belo Horizonte – MG – Brasil
{patricia,laender}@dcc.ufmg.br

(a)

6th International Workshop on Web Information and Data Management,
Washington, DC, USA, 2004.

Measuring Similarity Between Collection of Values
Carina F. Dorneles, Carlos A. Heuser,
Andrei E. N. Lima
UFRGS, Porto Alegre, Brazil
dorneles, heuser, aenlima@inf.ufrgs.br
Altigran da Silva, Edleno de Moura
UFAM, Manaus, Brazil
alti, edleno@dcc.ufam.br

(b)

Figura 4.1: Exemplos de informações sobre artigos científicos a serem armazenados no documento XML

apêndice A.

Cada caminho extraído foi associado manualmente aos conceitos que a informação representa no documento XML. Por exemplo, o primeiro caminho apresentado na figura 4.2(a) foi associado ao mesmo conceito que o primeiro caminho apresentado na figura 4.2(b). Ao final da associação, obtiveram-se um modelo conceitual global e as associações entre os caminhos extraídos dos documentos XML e este modelo.

congressos/congresso/nome	
congressos/congresso/ano	
congressos/congresso/local/pais	
congressos/congresso/local/UF	
congressos/congresso/local/cidade	
congressos/congresso/artigos/artigo/autores/nome	conferencias/evento/nome
congressos/congresso/artigos/artigo/autores/email	conferencias/evento/ano
congressos/congresso/artigos/artigo/autores/inst	conferencias/evento/local
congressos/congresso/artigos/artigo/cep	conferencias/evento/artigo/nome_art
congressos/congresso/artigos/artigo/local/pais	conferencias/evento/artigo/autores/autor/nome_aut
congressos/congresso/artigos/artigo/local/UF	conferencias/evento/artigo/autores/autor/email
congressos/congresso/artigos/artigo/local/cidade	conferencias/evento/artigo/autores/autor/inst_aut/nome_inst
congressos/congresso/artigos/artigo/titulo	conferencias/evento/artigo/autores/autor/inst_aut/endereco_inst

(a) (b)

Figura 4.2: Exemplos de caminhos extraídos dos documentos XML criados pelos alunos

Todos os caminhos extraídos dos treze documentos foram agrupados em uma única fonte para formar a base de caminhos utilizada na realização das consultas deste experimento. A fonte gerada dispõe de 135 caminhos, cada um com seus conceitos associados. Vale ressaltar que cada conceito do modelo global pode ser representado por mais de um caminho.

4.3.3 Realização das Consultas

Cada caminho extraído dos documentos é utilizado como parâmetro de uma consulta a fonte de caminhos. Portanto, cada função de similaridade executa, neste experimento, 135 consultas. Os resultados das consultas são comparados para determinar qual apresenta melhores resultados.

As três funções de similaridade ("*baseline*", PathSim e PathSim$_C$) também são avaliadas quando associadas a funções de pré-processamento. Quatro associações diferentes foram avaliadas. São elas: i) Associação à função Remoção de Caracteres Especiais (SCE), ii) Associação à função Remoção de Afixos (S.Afixos), iii) Associação à função Remoção de Nomes de Elementos Duplicados (S. El. Dupl) e iv) Associação a todas as

funções de pré-processamento (Completo).

Portanto, este experimento avalia três algoritmos de similaridades com cinco associações de funções de pré-processamento diferentes. Como cada avaliação é composta por 135 consultas, foram realizados 2.025 consultas à fonte de caminhos. Visto que cada consulta efetua 135 comparações de caminhos, neste experimento foram efetuadas 273.375 comparações entre caminhos.

4.3.4 Análise dos Resultados

Os resultados obtidos pelas medidas de comparação entre as funções neste experimento foram divididos em três gráficos. O primeiro, apresentado na figura 4.3, é uma análise da relação Revocação *versus* Precisão. O segundo, apresentado na figura 4.4, analisa os valores de precisão dos primeiros caminhos dos resultados. Já o terceiro, apresentado na tabela 4.3, sintetiza os ganhos e perdas dos algoritmos PathSim e PathSim$_C$ em relação ao "*baseline*" nas médias das medidas de precisão, revocação, média harmônica, medida F e precisão por resultado.

A análise da variação da precisão em relação a revocação de diversas funções em um mesmo gráfico, permite visualmente identificar qual função apresenta melhor resultado em cada intervalo de revocação. Na figura 4.3 é apresentado o gráfico de Revocação *versus* Precisão dos algoritmos "*baseline*", PathSim e PathSim$_C$ nas consultas efetuadas no primeiro experimento. Visualmente, o primeiro apresenta resultados inferiores em níveis de revocação mais baixos mas seus valores de precisão são superiores nos níveis de revocação mais altos. Ou seja, os algoritmos PathSim e PathSim$_C$ são melhores para retornar os primeiros resultados corretos, no entanto, é superado pelo "*baseline*" quando são exigidos todos os caminhos corretos da consulta.

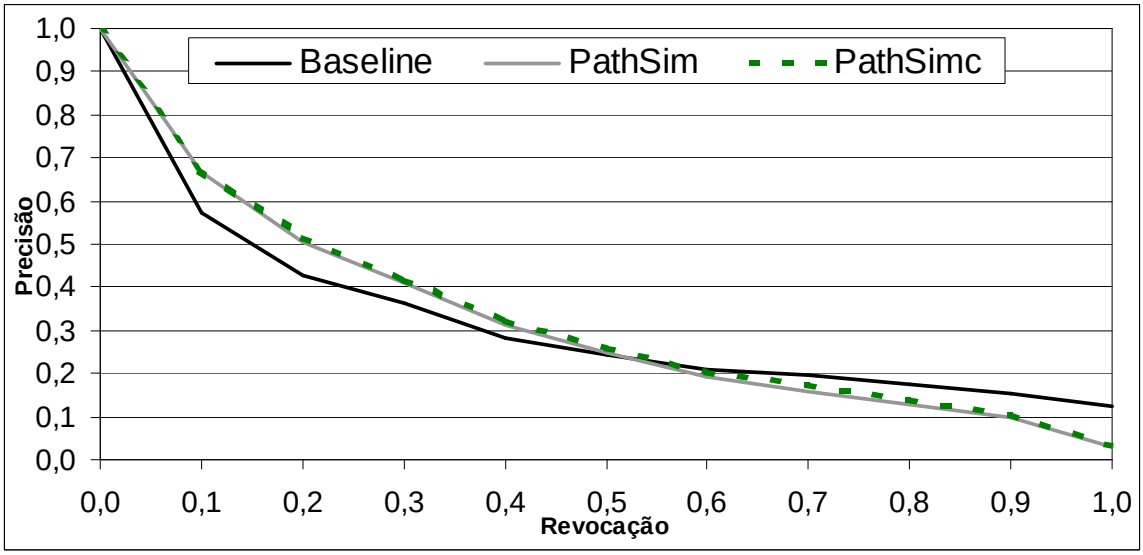


Figura 4.3: Revocação *versus* Precisão para consultas em uma fonte com caminhos XML do mesmo domínio do caminho consultado

Para analisar até que valor de revocação os algoritmos PathSim e PathSim$_C$ são superiores, todos os intervalos foram testados utilizando o método de análise *T-Student test*. Através da análise dos resultados deste, é possível concluir-se que o algoritmo PathSim apresenta uma melhora estatisticamente significativa de resultado para pesquisas que necessitem de revocação de até 0.6, enquanto que o algoritmo PathSim$_C$ apresenta uma

melhora estatisticamente significativa para valores de revocação de até $0,75$. Os valores das médias das precisões das funções para cada nível de revocação medidos neste experimento e utilizados para a criação do gráfico de Revocação *versus* Precisão e para os testes estatísticos são apresentados na tabela 4.2.

Tabela 4.2: Precisão por nível de revocação em uma fonte com caminhos XML de um domínio

Revocação	Baseline	PathSim	PathSimc
0	1	1	1
0,1	0,5728	0,666	0,6705
0,2	0,4268	0,5045	0,517
0,3	0,364	0,4102	0,4199
0,4	0,2832	0,3121	0,326
0,5	0,2433	0,2496	0,2611
0,6	0,2115	0,1921	0,205
0,7	0,1964	0,1597	0,1737
0,8	0,1753	0,1293	0,1414
0,9	0,1527	0,0985	0,1061
1	0,1219	0,031	0,0332

A segunda análise dos resultados deste experimento refere-se às precisões alcançadas pelas métricas quando analisados os resultados com maior valor de similaridade nas respostas, ou seja, os primeiros caminhos da lista da resposta às consultas. Na figura 4.4 são apresentados os gráficos com os valores da precisão dos três algoritmos com cinco associações diferentes a pré-processamento, são elas: i) Sem associação a funções de pré-processamento (Normal) ii)Associação à função Remoção de Caracteres Especiais (SCE), iii) Associação à função Remoção de Afixos (S.Afixos), iv) Associação à função Remoção de Nomes de Elementos Duplicados (S. El. Dupl) e v) Associação a todas as funções de pré-processamento (Completo).

Na figura 4.4 (a), são mostrados os valores de média da precisão das métricas para retornar os quinze primeiros caminhos por consulta. Na figura 4.4 (b) são apresentados os valores de média de precisão levando-se em conta apenas os dez primeiros caminhos retornados como resposta. Já na figura 4.4 (c) são apresentados os valores de média de precisão ao analisar os cinco primeiros caminhos das consultas.

A comparação das três situações apresentadas na figura 4.4 mostra que, além de sempre apresentar resultados superiores, os algoritmos PathSim e PathSim$_C$ apresentam ganhos mais significativos quanto mais ao topo da lista de resultados a avaliação estiver sendo efetuada, ou seja, os ganhos na análise dos cinco primeiros resultados (figura 4.4(c)), são maiores que os ganhos dos dez primeiros resultados (figura 4.4(b)), que são maiores que os ganhos nos quinze primeiros resultados (figura 4.4(a)). Esta característica é interessante para sistemas que objetivam realizar uma integração (semi)automática de esquemas, pois os melhores resultados encontram-se nas primeiras posições da lista de resposta.

Os resultados das comparações entre o "*baseline*"e os algoritmos PathSim e PathSim$_C$ utilizando as médias das medidas de precisão, revocação, média harmônica, medida F e precisão média por resultados; estão sintetizados na tabela 4.3. Todos os valores da tabela são referentes ao percentual de ganho, se maior que 0, ou perda, se menor que 0, do algoritmo em relação ao "*baseline*". Os resultados detalhados dos experimentos, que geraram esta tabela, são apresentados nas tabelas B.1, B.2, B.3, B.4 e B.5 do apêndice B. Todas as comparações foram feitas em condições iguais, ou seja, ambas os algoritmos

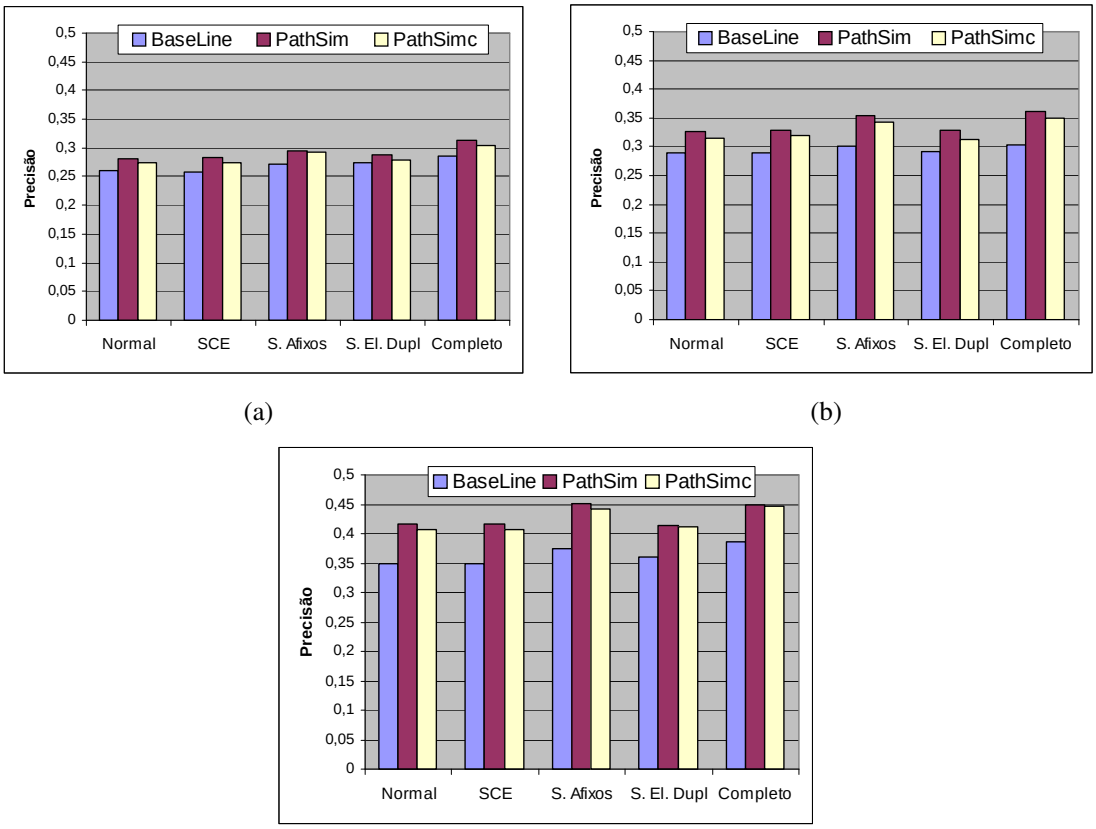


Figura 4.4: Valores de média da precisão das funções em consultas à fonte com caminhos XML de um domínio analisando: (a) 15 primeiros resultados (b) os primeiros 10 resultados e (c) os primeiros 5 resultados

comparados associados às mesmas funções de pré-processamento.

A coluna "Precisão" da tabela 4.3 apresenta os ganhos em porcentagem obtidos pelos algoritmos, quando comparadas as médias dos valores de precisão das consultas. O ganho da utilização do algoritmo PathSim varia entre 29% e 72%, enquanto que os ganhos do algoritmo PathSim$_C$ variam entre 31% e 61%. As variações apresentadas dependem de quais funções de pré-processamento estão sendo utilizadas. Tanto nesta análise, quanto nas próximas, essas variações mostram que a aplicação das funções de pré-processamento apresenta benefício superior no algoritmo PathSim do que no "*baseline*".

Quando medidas as média dos valores de revocação nas consultas neste experimento, o "*baseline*" tem resultados superiores, indicando que os algoritmos PathSim e PathSim$_C$ retornam menos resultados corretos. Os percentuais de perda são apresentados na coluna "Revocação" da tabela 4.3, variando entre 14% e 22%, para o algoritmo PathSim, e entre 13% e 21%, para o algoritmo PathSim$_C$.

Quando medidas as médias dos valores de média harmônica, entre os ganhos na precisão e as perdas na revocação das consultas, os algoritmos PathSim e PathSim$_C$ apresentam resultados superiores ao "*baseline*". Os ganhos são apresentados na coluna "Média Harmônica" da tabela 4.3, e indicam um ganho para o algoritmo PathSim que varia entre 18% e 33% e para o algoritmo PathSim$_C$, varia entre 21% e 31%.

A análise da média das medidas F, aplicando pesos de 20% para a precisão e 80% para a revocação, também mostra ganhos dos algoritmos PathSim e PathSim$_C$ em relação ao

Tabela 4.3: Ganho dos algoritmos PathSim e PathSim$_C$ para consultas efetuadas em fontes com caminhos do mesmo domínio do consultado

	Precisao		Revocação		Média Harmônica		Medida F (P=0.2,R=0.8)		Precisão Média Por Resultado	
	%PathSim	%PathSimc	%PathSim	%PathSimc	%PathSim	%PathSimc	%PathSim	%PathSimc	%PathSim	%PathSimc
Normal	29,0772	31,8777	-15,8005	-14,9494	18,5792	21,0241	5,7078	7,6030	12,5580	12,5950
SCE	29,1915	33,1430	-15,7182	-14,8671	18,7012	22,1632	5,8291	8,4539	12,5820	12,7876
S. Afixos	31,1559	34,6288	-14,2210	-13,3461	21,1722	24,2904	8,2050	10,6719	14,7438	13,6263
S. El. Dupl	69,5377	54,3393	-22,5682	-21,2607	30,2658	25,9123	7,6979	7,0356	21,0508	20,6093
Completo	72,1608	61,7013	-20,5996	-19,4999	33,6239	31,2469	10,7756	10,6568	21,1440	21,2048

"*baseline*". Os percentuais de ganho são apresentados na coluna "Medida F", e mostram ganhos entre 5% e 10% para o algoritmo PathSim e entre 7% e 10% para o algoritmo PathSim$_C$. Esta análise, hostil a funções que tenham baixa revocação, é bastante usada em sistemas de recuperação de informação. Apesar dos algoritmos PathSim e PathSim$_C$ apresentarem resultados inferiores na medida de revocação, quando comparados ao "*baseline*", os ganhos na medida de precisão são tão expressivos que os torna superiores na medida F.

A última coluna da tabela, nomeada "Precisão Média por Resultado", mostra o ganho obtido em relação ao "*baseline*", quando analisadas as médias das precisões médias por resultado. Nesta análise os ganhos dos algoritmos PathSim e PathSim$_C$ variam entre 12% e 21%. Esta análise é importante, pois o valor da média da precisão por resultado avalia a ordem em que os caminhos corretos foram retornados nas respostas às consultas e, portanto, é uma análise mais justa para comparar uma função que retorna todos os itens com uma que retorna apenas a maioria.

Visualizando todas essas análises, conclui-se que, apesar da diminuição observada na medida de revocação, o ganho na precisão torna os algoritmos PathSim e PathSim$_C$ superiores ao algoritmo "*baseline*", em um ambiente que contenha somente caminhos do domínio do caminho consultado.

4.4 Experimento com Consultas a Fontes com Caminhos XML de Vários Domínios

Este experimento simula consultas a caminhos XML em uma fonte onde existem, além de caminhos do domínio do caminho consultado, caminhos de outros domínios. O principal exemplo de sistema onde esta situação ocorre são as redes ponto-a-ponto.

Em redes ponto-a-ponto, o usuário apresenta o caminho XML da informação requerida, e cada ponto da rede o compara com os caminhos das informações disponíveis. Portanto, são feitas comparações com todas as estruturas disponíveis na rede e, provavelmente, alguns caminhos extraídos dessas estruturas são de domínios diferentes ao do caminho informado pelo usuário. Este experimento é importante para determinar se um algoritmo é eficiente em dizer que dois caminhos são realmente diferentes, ou seja, retornar similaridade 0 como resultados das comparações.

4.4.1 Objetivo

O objetivo deste experimento é comprovar que os algoritmos PathSim e PathSim$_C$ também apresenta melhorias significativas em sistemas onde existe a comparação entre

caminhos de domínios diferentes, ou seja, comprovar que mesmo com caminhos de outros domínios incluídos na fonte as comparações apresentam melhores resultados que a outra abordagem.

4.4.2 Obtenção da Fonte de Caminhos

A composição da fonte utilizada neste experimento é feita pela adição de caminhos de domínios diversos à fonte do primeiro experimento. Os caminhos adicionados foram extraídos de 19 documentos XML coletados na Internet. Os domínios desses arquivos são bem diversos, entre eles Catálogos de cd, currículo de professores, catálogo de plantas e informações sobre países. Os documentos extras geraram 1.202 caminhos, desta forma a fonte final desse experimentos contém 1.337. Os caminhos da fonte do primeiro experimentos são apresentados no apêndice A. Os caminhos adicionados não foram anexados à esta dissertação por questão de espaço.

4.4.3 Realização das Consultas

As mesmas 135 consultas realizadas no primeiro experimento foram processadas pelos três algoritmos nas cinco associações às funções de pré-processamento, consultando a nova fonte. O conjunto de respostas relevantes a cada consulta é o mesmo, pois entre os caminhos adicionados não existe nenhum que corresponda semanticamente a um dos caminhos consultados. Assim, a análise do conjunto de caminhos retornados pelas consultas possibilita a verificação da variação da precisão das funções em relação ao resultados obtidos no primeiro experimento.

Portanto, este experimento avalia três algoritmos de similaridades com cinco associações de funções de pré-processamento diferentes. Como cada avaliação é composta por 135 consultas, foram realizados 2.025 consultas à fonte de caminhos. Visto que cada consulta efetua 1.337 comparações de caminhos, neste experimento foram efetuadas 2.707.425 comparações entre caminhos.

4.4.4 Análise dos Resultados

Visto que apenas adicionamos caminhos de outros domínios à fonte de caminhos, e utilizamos os mesmos parâmetros de consulta, a diferença entre as respostas deste experimento, e do primeiro, se resume a inserção, na resposta, de caminhos que não são resposta correta ao caminho consultado. Assim, os valores de precisão e revocação do segundo experimento podem apenas ser iguais, ou inferiores, aos resultados mensurados no primeiro experimento.

A análise dos resultados deste experimento mostra que a precisão dos algoritmos PathSim e PathSim$_C$ sofrem uma ligeira redução de valores com a adição dos caminhos irrelevantes a fonte, enquanto que a precisão dos resultados do "*baseline*" sofre uma redução significativa.

O gráfico de revocação *versus* precisão deste experimento é apresentado na figura 4.5. Comparando-o, visualmente, com o gráfico correspondente do primeiro experimento (figura 4.3) nota-se que o "*baseline*" sofreu uma perda sensível na precisão dos resultados, principalmente quando se analisa o intervalo de revocação $[0.7, 1.0]$.

Esta impressão é comprovada ao se utilizar o método *T-Student test* para determinar o limite de revocação em que o ganho dos algoritmos PathSim e PathSim$_C$ são estatisticamente relevantes. As análises dos resultados do *T-Student test* mostram que o algoritmo PathSim tem um ganho significativo para consultas com revocação de até 0.7, enquanto

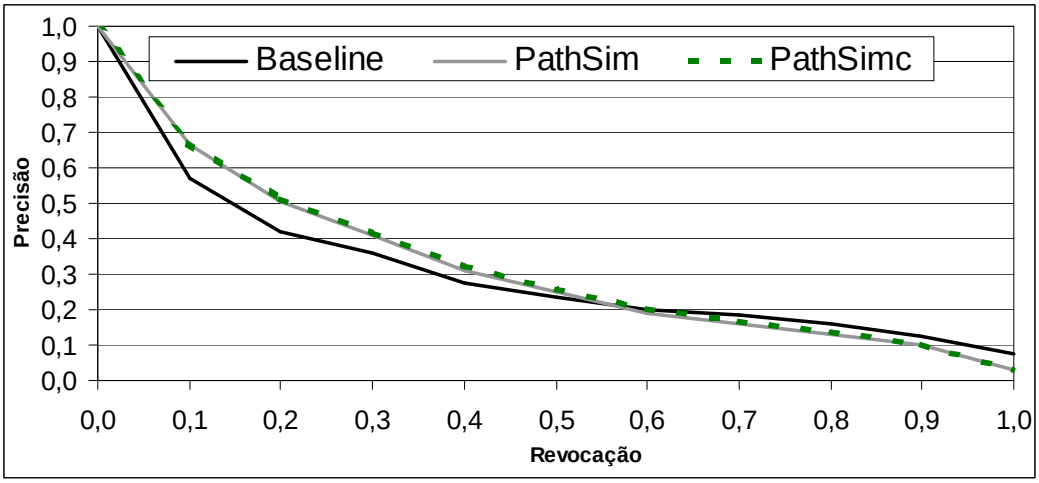


Figura 4.5: Revocação *versus* Precisão para o consultas em uma fonte com caminhos XML de diversos domínios

Tabela 4.4: Precisão por nível de revocação em uma fonte com caminhos XML de vários domínios

Revocação	Baseline	PathSim	PathSimc
0	1	1	1
0,1	0,5681	0,6654	0,67
0,2	0,422	0,5037	0,5164
0,3	0,3592	0,4084	0,4181
0,4	0,2764	0,3104	0,3242
0,5	0,2356	0,2483	0,2596
0,6	0,2018	0,1914	0,2038
0,7	0,1839	0,1591	0,1724
0,8	0,1599	0,1288	0,1392
0,9	0,1275	0,098	0,1026
1	0,0751	0,0302	0,0318

que o algoritmo PathSim$_C$ tem ganho significativo para consultas com revocação até 0.95. Ocorreu um sensível aumento nos limiares de revocação em que os algoritmos apresentam ganhos significativos. Os valores de precisão média dos algoritmos em diferentes níveis de revocação medidos neste experimento e utilizados para a montagem do gráfico de Revocação *versus* Precisão e para os testes estatísticos são apresentados na tabela 4.4.

Na figura 4.6 são apresentados gráficos análogos aos apresentados na figura 4.4, sendo aplicados aos resultados das consultas no segundo experimento. Os valores de precisão obtidos pelas três funções são idênticos aos obtidos no primeiro experimento nas três análises ((a)apenas os 15 primeiros caminhos, (b) apenas os 10 primeiros caminhos e (c) apenas os 5 primeiros caminhos). Isto indica que os caminhos adicionados a fonte de dados não interferem nos primeiros resultados da resposta de nenhum dos algoritmos testados nem do "*baseline*".

Para observar a interferência dos caminhos de outros domínios na resposta completa, são realizadas análises similares a da tabela 4.3, para os resultados do segundo experimento. Estas análises são mostradas na tabela 4.5. A comparação entre as tabelas mostra uma expressiva melhora nas porcentagens de ganho onde os algoritmos PathSim e PathSim$_C$ são superiores ao "*baseline*". Isto ocorre, primordialmente, devido a interferência dos caminhos de outros domínios na resposta do "*baseline*". Pois as interferências

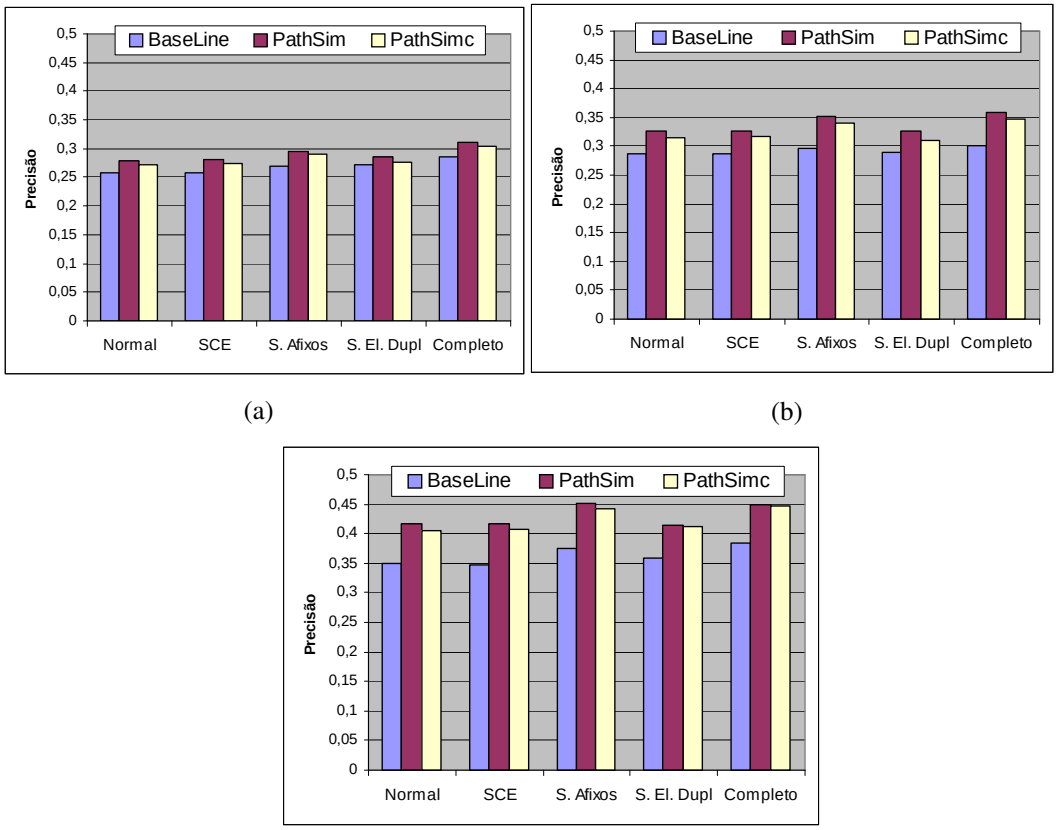


Figura 4.6: Valores da precisão média das funções em consultas à fonte com caminhos de diversos domínios analisando: (a) os primeiros 15 caminhos (b) os primeiros 10 caminhos e (c) os primeiros 5 caminhos

destes nos resultados obtidos pelos algoritmos PathSim e PathSim$_C$ são muito pequenas. Assim, as melhorias nos valores das porcentagens de ganho das funções são devido a diminuição dos valores das medidas que avaliam o "*baseline*". Os resultados detalhados dos experimentos que geraram esta tabela são apresentados nas tabelas B.6, B.7, B.8, B.9 e B.10 do apêndice B.

A tabela 4.5 apresenta, na coluna "Precisão", os ganhos na média de precisão do algoritmo PathSim variando entre 147% e 216% e do algoritmo PathSim$_C$ variando entre 148% e 193%. A coluna "Revocação" mostra que o algoritmo PathSim apresenta perdas na revocação que variam de 14% a 22% e o algoritmo PathSim$_C$ apresenta perdas que variam de 13% a 21%. As perdas na média dos valores de revocação mantiveram-se constantes no primeiro e no segundo experimento. A coluna "Média Harmônica" mostra que, quando comparados com a média das médias harmônicas das consultas, o algoritmo PathSim apresenta ganhos que variam de 119% a 138% e o algoritmo PathSim$_C$ apresenta ganhos que variam de 120% a 131%. A coluna "Medida F" apresenta a comparação utilizando 20% de peso para a precisão e 80% de peso para a revocação. Mesmo nessa comparação que prima por funções que apresentam melhore desempenho na medida de revocação, o algoritmo PathSim apresenta ganhos que variam de 79% a 84%, enquanto que o algoritmo PathSim$_C$ apresenta ganhos que variam entre 75% e 84%. A última coluna "Precisão Média por Resultado" mostra que os algoritmos PathSim e PathSim$_C$ têm ganhos que variam de 17% a 25%.

Tabela 4.5: Ganho das funções PathSim e PathSim$_C$ para pesquisas em uma base que contém caminhos de diversos domínios

	Precisao		Revocação		Média Harmônica		Medida F (P=0.2,R=0.8)		Precisão Média Por Resultado	
	%PathSim	%PathSimc	%PathSim	%PathSimc	%PathSim	%PathSimc	%PathSim	%PathSimc	%PathSim	%PathSimc
Normal	148,4915	148,9909	-15,8005	-14,9494	119,8446	120,1574	79,3647	79,6101	17,2491	17,0534
SCE	151,3536	152,6372	-15,7182	-14,8671	122,3022	123,2328	81,2000	81,7481	17,2085	17,0557
S. Afixos	147,4414	150,7108	-14,2210	-13,3461	121,3381	124,1035	82,1145	84,1638	18,5516	17,3319
S. El. Dupl	213,7327	181,2571	-22,5682	-21,2607	133,8768	121,9392	79,0359	75,0752	25,2757	25,6349
Completo	216,3316	193,2653	-20,5996	-19,4999	138,8428	131,6410	84,3284	81,9664	24,8230	24,9845

Analisando os valores da tabela 4.5 da tabela 4.3 nota-se que a única medida em que não existe alteração significativa nos valores é na medida de Revocação. Isto significa que a adição de caminhos de outros domínios na fonte de caminhos não altera a única métrica na qual os algoritmos PathSim e PathSim$_C$ são inferiores ao algoritmo "*baseline*". A melhora observada nas medidas Média Harmônica, Medida F e Precisão Média por Resultado indicam que a influência dos caminhos adicionados na fonte é superior no algoritmo "*baseline*" do que nos algoritmos PathSim e PathSim$_C$. De forma análoga à conclusão da análise do gráfico de Revocação *versus* Precisão, a diferença menor na última comparação indica que a influência ocorre ao retornar os últimos resultados.

A conclusão do segundo experimento é que a adição de caminhos de outros domínios na fonte de caminhos não causa um efeito danoso significativo nos resultados obtidos pelos algoritmos PathSim e PathSim$_C$. No entanto, a precisão do "*baseline*" é bastante prejudicada nesse ambiente, principalmente para retornar os últimos caminhos corretos das consultas. Portanto, nesse ambiente, o algoritmo PathSim é bem superior para valores baixos de revocação e apresenta comportamento equivalente ao "*baseline*" em níveis de revocação altos.

5 CONCLUSÃO

A importância dos algoritmos de comparação por similaridade de informações expressas em XML aumenta com o crescente uso desse padrão para armazenar dados e trocar informações entre sistemas. Em uma estrutura XML, uma informação textual pode ser caracterizada pelo caminho do elemento raiz até o elemento que contém esta informação. Essa especificação oferecida pelo caminho torna necessário o desenvolvimento de funções de similaridade que mensurem a semelhança entre caminhos XML.

O algoritmo editDistance é um excelente método para mensurar a distância de edição entre cadeias de caracteres e, posteriormente, definir a similaridade entre elas baseada nesta distância. Analisando um caminho como uma cadeia de nomes de elementos, esta dissertação define um algoritmo, com base no editDistance, que mensura a distância de edição entre dois caminhos. Este algoritmo pode efetuar a comparação entre nomes de elementos utilizando qualquer função de similaridade, ou combinação de funções, que retorne um valor real no intervalo [0,1], possibilitando que o número de operações da distância calculada seja um valor real. Utilizando o valor de distância de edição calculado, o algoritmo determina a similaridade entre os caminhos.

As seguintes contribuições são apresentadas por esta dissertação:

Funções de simplificação de caminhos Três funções de pré-processamento foram propostas para simplificar caminhos XML com uma perda semântica mínima. Apesar de serem projetadas para melhorar os resultados do algoritmo PathSim, estas funções podem ser utilizadas na fase de pré-processamento de qualquer função de similaridade entre caminhos.

PathSim Foi proposto um algoritmo para calcular a similaridade entre caminhos baseada no número de operações sobre nomes de elementos necessárias para transformar um caminho no outro. O cálculo do custo da operação de substituição utiliza valores de semelhança entre os nomes dos elementos componentes dos caminhos, tornando os resultados mais precisos. O algoritmo PathSim permite a utilização de qualquer função de similaridade entre nomes de elementos para calcular o custo de uma operação de substituição, portanto uma implementação do algoritmo pode ser incrementada com funções específicas de um domínio, além de poder beneficiar-se do uso de dicionários de sinônimos, thesaurus e ontologias de termos. Combinações de funções de similaridade também podem ser utilizadas para computar a semelhança entre dois nomes de elementos.

PathSim$_C$ Foi proposto uma variante do algoritmo PathSim, nomeada PathSim$_C$, que permite a comparação de combinações de n nomes de elementos de um caminho com a combinação de m nomes de elementos do outro. Esta variante apresenta

grande utilidade na comparação entre estruturas de níveis de detalhamento diferente, pois os nomes de alguns elementos de uma estrutura podem ser compactados para o nome de um elemento de outra estrutura.

PathSim$_A$ Foi proposto uma variante do algoritmo PathSim, nomeada PathSim$_A$, que utiliza técnicas de alinhamento entre cadeias de caracteres aplicadas às seqüências de nomes de elementos. As técnicas de alinhamento local utilizadas propiciam melhores resultados em comparações em que um caminho engloba o outro. Entre os sistemas que podem beneficiar-se com este algoritmo destacam-se as redes ponto-a-ponto que disponibilizam consultas a documentos XML.

Foram realizados experimentos com os algoritmos PathSim e PathSim$_C$. Ambos apresentaram resultados superiores, em análises de diversas medidas, à abordagem usada como base para comparação. Apenas quando mensurada a revocação dos algoritmos os resultados são ligeiramente inferiores. O menor valor da medida de revocação ocorre devido os algoritmos utilizarem limiares de similaridade entre os nomes dos elementos, logo, estruturas que utilizam sinônimos para representar o mesmo objeto não foram encontradas pelos algoritmos. A utilização de funções de similaridade entre nomes de elementos específicas para o domínio auxiliadas por dicionários de sinônimos podem diminuir essa diferença.

As análises mais importantes efetuadas são as médias das precisões médias por resultado retornado. Essas mostram que o algoritmo PathSim apresenta ganhos de 12%, quando as consultas são efetuadas em bases com caminhos do mesmo domínio da consulta e, ganhos de 17%, quando as consultas são efetuadas em bases com caminhos de diversos domínios. A maior precisão do algoritmo PathSim em baixos valores de revocação, mostrado na seção de experimentos, torna o algoritmo ideal para aplicativos onde o usuário precisa tomar uma decisão baseado nos primeiros itens de resposta, por exemplo sistemas de integração semi-automática.

Os experimentos foram efetuados sem a utilização de algoritmos de comparação entre nomes de elementos específicos para o domínio dos caminhos. A utilização de algoritmos de similaridade que utilizam *thesaurus*, dicionários de sinônimos ou ontologias de domínio como função de comparação de nomes de elementos poderia melhorar os resultados obtidos pelos algoritmos PathSim e PathSim$_C$.

A continuação da pesquisa realizada nesta dissertação pode ser efetuada em diversas frentes:

Funções de simplificação de caminhos Efetuar experimentos específicos para comprovar o benefício da utilização das funções de pré-processamento de caminhos associadas ao algoritmo PathSim, suas variantes e outros algoritmos que efetuem comparações entre caminhos XML.

PathSim Coletar documentos XML para fazer outros experimentos. Os caminhos da base utilizada foram extraídos de documentos que tiveram a mesma origem, visto que os alunos basearam-se nas mesmas informações para criar os documentos. Mesmo assim, os caminhos extraídos apresentaram estruturas bem diferente. Seria interessante mensurar as medidas de avaliação do algoritmo em bases com documentos do mesmo domínio mas origens diferentes. Também é importante fazer experimentos utilizando funções de similaridade entre nomes de elementos que utilizem dicionários de sinônimos, *thesaurus*, ontologias e combinação de funções de

similairdade, para definir os ganhos obtidos com funções de similaridade específicas a um domínio.

PathSim$_C$ Fazer mais experimentos em bases diferentes para comprovar melhor o benefício deste tipo de comparação, pois os caminhos da base de teste não apresentam muitos casos de composição de nomes de elementos. Realizar experimentos com variações diferentes de combinações para verificar o limite do número de nomes de elementos de uma composição que apresentam bons resultados.

PathSim$_A$ Montar uma base de dados para experimentos contendo documentos de um domínio que apresentem diferentes níveis de granularidades e de escopo nas estruturas. Também podem ser adicionados a esta base caminhos de outros domínios, para fazer experimentos que simulem consultas ponto-a-ponto e comprovar a eficácia do algoritmo PathSim$_A$ nesses sistemas.

Similaridade de elementos XML Pesquisar a melhor forma de utilizar valores de similaridade entre caminhos extraídos de dois elementos XML, para determinar a similaridade entre os elementos. Para tanto, é necessário combinar, além dos valores de similaridade entre caminhos calculados pelo algoritmo PathSim, as similaridades entre os conteúdos textuais que o último elemento dos caminhos armazenam. Funções de similaridade específicas para o domínio do documento devem ser utilizadas para se alcançar bons resultados.

REFERÊNCIAS

AL-EKRAM, R.; ADMA, A.; BAYSAL, O. diffX: an algorithm to detect changes in multi-version xml documents. In: ANNUAL INTERNATIONAL CONFERENCE HOSTED BY THE IBM CENTERS FOR ADVANCED STUDIES, CASCON, 15., 2005. **Proceedings...** [S.l.: s.n.], 2005. p.1–11.

AMER-YAHIA, S.; KOUDAS, N.; MARIAN, A.; SRIVASTAVA, D.; TOMAN, D. Structure and Content Scoring for XML. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 31., 2005. **Proceedings...** [S.l.: s.n.], 2005. p.361–372.

BAEZA-YATES, R. A.; RIBEIRO-NETO, B. A. **Modern Information Retrieval**. [S.l.]: ACM Press: Addison-Wesley, 1999.

BARU, C. et al. XML-based information mediation with MIX. **SIGMOD Record**, New York, v.28, n.2, p.597–599, June 1999. Trabalho apresentado na SIGMOD,1999.

BILLE, P. **Tree Edit Distance, Alignment Distance and Inclusion**. [S.l.:s.n.], 2003. (TR-2003-23).

BOYD, M. et al. AutoMed: a bav data integration system for heterogeneous data sources. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, CAISE, 16., 2004. **Proceedings...** Riga: Riga Technical Uiversity, 2004. p.82–97.

BRADLEY, N. **The XML Companion**. Boston, MA, USA: Addison-Wesley Longman, 1998.

CHAVES, M. S. Um estudo e apreciação sobre algoritmos de stemming para a língua portuguesa. In: JORNADAS IBEROAMERICANAS DE INFORMáTICA, 9., 2003, Cartagena de Indias, Colômbia. **Anais...** [S.l.: s.n.], 2003.

COBENA, G.; ABITEBOUL, S.; MARIAN, A. Detecting Changes in XML Documents. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 18., 2002. **Proceedings...** [S.l.: s.n.], 2002. p.41–52.

COHEN, W. W.; RAVIKUMAR, P.; FIENBERG, S. E. A Comparison of String Distance Metrics for Name-Matching Tasks. In: WORKSHOP ON INFORMATION INTEGRATION ON THE WEB, IIWEB, 2., 2003. **Proceedings...** [S.l.: s.n.], 2003. p.73–78.

DO, H. H.; RAHM, E. Coma - A System for Flexible Combination of Schema Matching Approaches. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 28., 2002, Hong Kong, China. **Proceedings...** [S.l.: s.n.], 2002. p.610–621.

DORNELES, C. F.; HEUSER, C. A.; LIMA, A. E. N.; SILVA, A. S. da; MOURA, E. S. de. Measuring similarity between collection of values. In: INTERNATIONAL WORKSHOP ON WEB INFORMATION AND DATA MANAGEMENT, WIDM, 6., 2004. **Proceedings...** New York: ACM Press, 2004. p.56–63.

FAN, H.; POULOVASSILIS, A. Using AutoMed metadata in data warehousing environments. In: INTERNATIONAL WORKSHOP ON DATA WAREHOUSING AND OLAP, DOLAP, 6., 2003, New Orleans, USA. **Proceedings...** [S.l.: s.n.], 2003. p.86–93.

GOOGLE. Disponível em: <http://www.google.com.br>: Acesso em: nov. 2006.

HALEVY, A. Y.; IVES, Z. G.; MADHAVAN, J.; MORK, P.; SUCIU, D.; TATARINOV, I. The Piazza Peer Data Management System. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.16, n.7, p.787–798, 2004.

JARO, M. A. Advances in record linking methodology as applied to the 1985 census of Tampa Florida. **Journal of the American Statistical Society**, [S.l.], v.64, p.1183–1210.

JONES, K. S.; WILLETT, P. (Ed.). **Readings in information retrieval**. San Francisco, CA, USA: Morgan Kaufmann, 1997.

JOSHI, S. et al. A bag of paths model for measuring structural similarity in Web documents. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, KDD, 9., 2003. **Proceedings...** [S.l.: s.n.], 2003. p.577–582.

KILPELäINEN, P.; MANNILA, H. The tree inclusion problem. In: INTERNATIONAL JOINT CONFERENCE ON THEORY AND PRACTICE OF SOFTWARE DEVELOPMENT, TAPSOFT, 4., 1991. **Theory and practice of software development**. New York: Springer-Verlag, 1991. p.202–214.

KLEIN, P. Computing the Edit-Distance between Unrooted Ordered Trees. In: ANNUAL EUROPEAN SYMPOSIUM ON ALGORITHMS, 6., 1998, Venice, Italy. **Proceedings...** Berlin: Springer-Verlag, 1998. p.91–102. (Lecture Notes in Computer Science, v. 1461)

KUKULENZ, D.; HERGET, K.-P.; PAULI, J. Improving Retrieval by a Similarity Thesaurus based on Hyperlink Structure. In: INTERNATIONAL CONFERENCE ON DATABASES AND APPLICATIONS, IASTED, 2005, Austria. **Proceedings...** [S.l.: s.n.], 2005. p.29–34.

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions, and reversals. **Soviet Physics Doklady**, [S.l.], v.10, n.8, p.707–710, 1966.

LIANG, B. et al. Semantic Similarity Based Ontology Cache. In: ASIA PACIFIC WEB CONFERENCE, APWEB, 8., 2006. **Proceedings...** [S.l.: s.n.], 2006. p.250–262.

LUJÁN-MORA, S.; PALOMAR, M. Reducing Inconsistency in Integrating Data From Different Sources. In: INTERNATIONAL DATABASE ENGINEERING AND APPLICATIONS SYMPOSIUM, IDEAS, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.209–218.

LYCOS. Disponível em: <http://www.lycos.com>. Acesso em: nov. 2006.

MA, Y.; CHBEIR, R. Content and Structure Based Approach For XML Similarity. In: INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION TECHNOLOGY, CIT, 5., 2005. **Proceedings...** [S.l.: s.n.], 2005. p.136–140.

MARIAN, A.; ABITEBOUL, S.; COBENA, G.; MIGNET, L. Change-Centric Management of Versions in an XML Warehouse. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 27., 2001. **Proceedings...** [S.l.: s.n.], 2001. p.581–590.

MERGEN, S. L. S.; HEUSER, C. A. Carla: uma técnica para comparação de cadeias de caracteres. In: ESCOLA REGIONAL DE BANCO DE DADOS, ERBD, 1., 2005, Porto Alegre, Brazil. **Anais...** Porto Alegre: SBC, 2005. p.55–60.

NIERMAN, A.; JAGADISH, H. V. Evaluating Structural Similarity in XML Documents. In: INTERNATIONAL WORKSHOP ON THE WEB AND DATABASES, WEBDB, 5., 2002, Madison, USA. **Proceedings...** [S.l.: s.n.], 2002. p.61–66.

OGAWA, T.; INUZUKA, N. Similarity of Documents Using Reconfiguration of Thesaurus. In: KNOWLEDGE-BASED INTELLIGENT INFORMATION AND ENGINEERING SYSTEMS, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.95–102.

ORENGO, V.; HUYCK, C. A Stemming Algorithmm for the Portuguese Language. In: STRING PROCESSING AND INFORMATION RETRIEVAL, SPIRE, 8., 2001. **Proceedings...** [S.l.: s.n.], 2001. v.01, p.0186.

PLAS, L. van der; TIEDEMANN, J. Finding Synonyms Using Automatic Word Alignment and Measures of Distributional Similarity. In: CONFERENCE OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 2006. **Proceedings...** [S.l.: s.n.], 2006.

PRESS, W. H. et al. **Numerical recipes in C**: the art of scientific computing. New York, NY, USA: Cambridge University Press, 1988.

RAHM, E.; BERNSTEIN, P. A. A survey of approaches to automatic schema matching. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 27., 2001, Rome, Italy. **Proceedings...** [S.l.: s.n.], 2001. p.334–350.

SCHÖNING, H. Tamino - A DBMS designed for XML. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 17., 2001, Heidelberg, Germany. **Proceedings...** [S.l.: s.n.], 2001. p.149–154.

SELLERS, P. H. The Theory and Computation of Evolutionary Distances: pattern recognition. **Proceedings of Journal of Algorithms**, [S.l.], v.1, n.4, p.359–373, 1980.

TANAKA, E.; TANAKA, K. The tree-to-tree editing problem. **International Journal Pattern Recognition and Atificial Intelligence**, [S.l.], v.2, n.2, p.221–240, 1988.

TATARINOV, I. et al. The Piazza peer data management project. **SIGMOD Record**, [S.l.], v.32, n.3, p.47–52, 2003.

VALIENTE, G. An Efficient Bottom-Up Distance between Trees. In: STRING PROCESSING AND INFORMATION RETRIEVAL, SPIRE, 8., 2001. **Proceedings...** [S.l.: s.n.], 2001. p.212–219.

VAN RIJSBERGEN, C. J. **Information Retrieval**. 2nd ed. [S.l.]: Dept. of Computer Science, University of Glasgow, 1979.

WANG, J. T.-L.; ZHANG, K. Finding similar consensus between trees: an algorithm and a distance hierarchy. **Pattern Recognition**, [S.l.], v.34, n.1, p.127–137, 2001.

WANG, L.; GUSFIELD, D. Improved Approximation Algorithms for Tree Alignment. In: ANNUAL SYMPOSIUM ON COMBINATORIAL PATTERN MATCHING, CPM, 7., 1996, London, UK. **Proceedings...** [S.l.]: Springer-Verlag, 1996. p.220–233.

WANG, Y.; DEWITT, D. J.; CAI, J. yi. X-Diff: an effective change detection algorithm for xml documents. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 21., 2003, Bangalore, India. **Proceedings...** [S.l.: s.n.], 2003. p.519–530.

WANG, Y. R.; MADNICK, S. E. The Inter-Database Instance Identification Problem in Integrating Autonomous Systems. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 5., 1989, Los Angeles, USA. **Proceedings...** [S.l.: s.n.], 1989. p.46–55.

WEN, G.; JIANG, L.; SHADBOLT, N. R. Ontology-Based Similarity Between Text Documents on Manifold. In: ASIAN SEMANTIC WEB CONFERENCE, ASWC, 1., 2006, Beijing, China. **Proceedings...** [S.l.: s.n.], 2006. p.113–125.

WIKIPEDIA. **Média Harmônica**. Disponível em: <http://pt.wikipedia.org/wiki/Média_harmônica>. Acesso em: dez. 2006.

XML. **Extensible Markup Language (XML)**. Disponível em: <http://www.w3.org/XML/>. Acesso em: dez. 2006.

ZHAI, Y.; LIU, B. Structured Data Extraction from the Web Based on Partial Tree Alignment. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.18, n.12, p.1614–1628, 2006.

ZHANG, K. A Constrained Edit Distance Between Unordered Labeled Trees. **Algorithmica**, [S.l.], v.15, n.3, p.205–222, 1996.

ZHANG, K.; SHASHA, D. Simple fast algorithms for the editing distance between trees and related problems. **SIAM Journal on Computing**, Philadelphia, PA, USA, v.18, n.6, p.1245–1262, 1989.

ZHAO, H.; RAM, S. Entity identification for heterogeneous database integration–a multiple classifier system approach and empirical evaluation. **Information Systems**, [S.l.], v.30, n.2, p.119–132, 2005.

APÊNDICE A CAMINHOS EXTRAÍDOS DAS FONTES XML

Este apêndice contém o caminhos extraídos dos treze documentos XML criados por alunos da UFRGS para armazenar informações sobre artigos científicos. Cada documento é representado por uma fonte abaixo, as fontes estão numeradas de 1 a 13 e cada fonte contém os caminhos extraídos do documento. Os caminhos foram utilizados como fonte para as consultas do experimento mostrado na seção 4.3, e como parte da fonte do experimentos mostrado na seção 4.4.

Fonte 1

congressos/congresso/nome
congressos/congresso/ano
congressos/congresso/local/pais
congressos/congresso/local/UF
congressos/congresso/local/cidade
congressos/congresso/artigos/artigo/autores/nome
congressos/congresso/artigos/artigo/autores/email
congressos/congresso/artigos/artigo/autores/inst
congressos/congresso/artigos/artigo/cep
congressos/congresso/artigos/artigo/local/pais
congressos/congresso/artigos/artigo/local/UF
congressos/congresso/artigos/artigo/local/cidade
congressos/congresso/artigos/artigo/titulo

Fonte 2

eventos/evento/titulo
eventos/evento/edicao
eventos/evento/ano
eventos/evento/local/cidade
eventos/evento/local/estado
eventos/evento/local/pais
eventos/evento/artigos publicados/artigo/titart
eventos/evento/artigos publicados/artigo/autores/autor/nome
eventos/evento/artigos publicados/artigo/autores/autor/email
eventos/evento/artigos publicados/artigo/instituicoes/instituicao/nomeinst
eventos/evento/artigos publicados/artigo/instituicoes/instituicao/endereco/caixa postal
eventos/evento/artigos publicados/artigo/instituicoes/instituicao/endereco/cep
eventos/evento/artigos publicados/artigo/instituicoes/instituicao/endereco/local/cidade
eventos/evento/artigos publicados/artigo/instituicoes/instituicao/endereco/local/estado
eventos/evento/artigos publicados/artigo/instituicoes/instituicao/endereco/local/pais

Fonte 3

congressos/congresso/nome
congressos/congresso/artigo/titulo
congressos/congresso/artigo/autores/nome
congressos/congresso/artigo/autores/instituicao/nome
congressos/congresso/artigo/autores/instituicao/endereco
congressos/congresso/artigo/autores/emails

Fonte 4

congressos/evento/nome
congressos/evento/ano
congressos/evento/local/pais
congressos/evento/local/cidade
congressos/evento/trabalhos/artigo/nome
congressos/evento/trabalhos/artigo/autores/filiacao/nome
congressos/evento/trabalhos/artigo/autores/filiacao/endereco
congressos/evento/trabalhos/artigo/autores/filiacao/pessoas/autor/nome
congressos/evento/trabalhos/artigo/autores/filiacao/pessoas/autor/email

Fonte 5

congressos/congresso/nome
congressos/congresso/pais
congressos/congresso/UF
congressos/congresso/cidade
congressos/congresso/artigo/nome
congressos/congresso/artigo/autor/nome
congressos/congresso/artigo/autor/email
congressos/congresso/artigo/instituicao/nome
congressos/congresso/artigo/instituicao/depto
congressos/congresso/artigo/instituicao/cidade
congressos/congresso/artigo/instituicao/uf
congressos/congresso/artigo/instituicao/pais

Fonte 6

congressos/congresso/nome
congressos/congresso/local
congressos/congresso/data
congressos/congresso/artigos/artigo/nome
congressos/congresso/artigos/artigo/autores/autor
congressos/congresso/artigos/artigo/intituicao/nome
congressos/congresso/artigos/artigo/intituicao/univ
congressos/congresso/artigos/artigo/intituicao/cep
congressos/congresso/artigos/artigo/intituicao/cidade
congressos/congresso/artigos/artigo/intituicao/estado
congressos/congresso/artigos/artigo/intituicao/pais
congressos/congresso/artigos/artigo/intituicao/emails/email

Fonte 7

artigos/artigo/tituloartigo
artigos/artigo/congresso
artigos/artigo/autores/autor/nomeautor
artigos/artigo/autores/autor/instituicaoautor
artigos/artigo/autores/autor/emailautor

artigos/artigo/autores/autor/enderecoinstituicao

Fonte 8

conferencias/evento/nome
conferencias/evento/ano
conferencias/evento/local
conferencias/evento/artigo/nome_art
conferencias/evento/artigo/autores/autor/nome_aut
conferencias/evento/artigo/autores/autor/email
conferencias/evento/artigo/autores/autor/inst_aut/nome_inst
conferencias/evento/artigo/autores/autor/inst_aut/endereco_inst

Fonte 9

artigos/evento/nome
artigos/evento/ano
artigos/evento/pais
artigos/evento/UF
artigos/evento/cidade
artigos/evento/artigo/titulo
artigos/evento/artigo/autor/nome
artigos/evento/artigo/autor/email
artigos/evento/artigo/instituicao/nome
artigos/evento/artigo/instituicao/depto
artigos/evento/artigo/instituicao/localizacao/cidade
artigos/evento/artigo/instituicao/localizacao/uf
artigos/evento/artigo/instituicao/localizacao/pais
artigos/evento/artigo/instituicao/localizacao/cep
artigos/evento/artigo/instituicao/localizacao/cpostal

Fonte 10

congressos/congresso/nomecongresso
congressos/congresso/anocongresso
congressos/congresso/localcongresso
congressos/congresso/artigo
congressos/congresso/autores/autor/nomeautor
congressos/congresso/autores/autor/sobrenomeautor
congressos/congresso/autores/autor/mailautor
congressos/congresso/autores/instituicao/setor intit
congressos/congresso/autores/instituicao/nome instit
congressos/congresso/autores/instituicao/caixa postal
congressos/congresso/autores/instituicao/cep
congressos/congresso/autores/instituicao/local instit

Fonte 11

congr/nome
conferencias/evento/titulo
conferencias/evento/ano
conferencias/evento/pais
conferencias/evento/estado
conferencias/evento/cidade
conferencias/evento/artigos/artigo/titulo
conferencias/evento/artigos/artigo/autores/autor/nome

conferencias/evento/artigos/artigo/autores/autor/e-mail
conferencias/evento/artigos/artigo/instituicao/nome
conferencias/evento/artigos/artigo/instituicao/departamento
conferencias/evento/artigos/artigo/instituicao/cidade
conferencias/evento/artigos/artigo/instituicao/estado
conferencias/evento/artigos/artigo/instituicao/pais
conferencias/evento/artigos/artigo/instituicao/cep
conferencias/evento/artigos/artigo/instituicao/cxpostal

Fonte 12

documento/evento/artigo
documento/evento/nomeevento
documento/evento/autores
documento/evento/local
documento/evento/contato

Fonte 13

congr/local
congr/tema
congr/participante
congr/insti
congr/endinsti
congr/emails

APÊNDICE B RESULTADOS DETALHADOS DOS EXPERIMENTOS

Este apêndice apresenta em detalhes as avaliações das funções de similaridade realizadas nos experimentos detalhados no capítulo 4. Cada tabela apresenta os resultados das três funções ("baseline", PathSim e PathSim$_C$) utilizando as mesmas funções de pré-processamento e consultando a mesma fonte de caminhos. As duas últimas linhas das tabelas indicam o percentual de ganho das funções PathSim e PathSim$_C$ em relação ao "baseline" na medida de avaliação indicada pela coluna. O significado das colunas das tabelas são:

- Função - Indica qual a função de similaridade foi utilizada nas consultas;
- Encontrados - Indica quantos caminhos foram encontrados corretamente pelas consultas;
- Existentes - Indica quantos caminhos deveriam ser encontrados pelas consultas;
- Retornados - Indica quantos caminhos foram retornados nos conjuntos de respostas às consultas;
- Precisão - Indica a precisão média das consultas utilizando a função de similaridade;
- Revocação - Indica a precisão média das consultas utilizando a função de similaridade;
- Média Harmônica - Indica o valor de média harmônica médio das consultas utilizando a função de similaridade;
- Medida F(P=0.2,R=0.8) - Indica o valor médio da medida F, com pesos de 20% para a precisão e 80% para a revocação, utilizando a função de similaridade
- Precisão Média por Resultado - Indica o valor médio de Precisão Média por resultado retornado corretamente alcançado utilizando a função de similaridade.

As tabelas B.1, B.2, B.3, B.4 e B.5 são, respectivamente, as avaliações com as funções de pré-processamento remoção de caracteres especiais, remoção de afixos, remoção de nomes de elementos duplicados e todas as funções; efetuadas em uma fonte onde apenas existem caminhos de domínio igual ao domínio do caminho consultado.

As tabelas B.6, B.7, B.8, B.9 e B.10 são, respectivamente, as avaliações com as funções de pré-processamento remoção de caracteres especiais, remoção de afixos, remoção de nomes de elementos duplicados e todas as funções; efetuadas em uma fonte onde existem caminhos de vários domínios.

Tabela B.1: Resultados dos algoritmos sem pré-processamento para consultas efetuadas em fontes com caminhos do mesmo domínio do caminho

Função	Encontrados	Existentes	Retornados	Mean Precision	Revocação	Mean Harmonic	Mean F Measure	Mean Average Precision
Baseline	1913	1913	16227	0,119304049	1	0,210822215	0,394639715	0,263882084
PathSim	1597	1913	11327	0,153994296	0,84199515	0,249991234	0,417165136	0,297020499
PathSimc	1617	1913	11281	0,157335378	0,85050594	0,255145745	0,424644314	0,297118086
			% Ganho PathSim	29,07717499	-15,800485	18,57917075	5,707844339	12,55803899
			% Ganho PathSimc	31,87765177	-14,949406	21,02412711	7,603035863	12,59502028

Tabela B.2: Resultados dos algoritmos com remoção de caracteres especiais para consultas efetuadas em fontes com caminhos do mesmo domínio do caminho

Função	Encontrados	Existentes	Retornados	Precisão	Revocação	Média Harmônica	Medida F (P=0.2,R=0.8)	Precisão Média Por Resultado
Baaseline	1913	1913	16234	0,119264158	1	0,210742081	0,394493364	0,265302528
PathSim	1599	1913	11334	0,154079175	0,8428182	0,250153302	0,417488597	0,29868288
PathSimc	1619	1913	11174	0,158791848	0,85132898	0,257449275	0,42784362	0,29922846
			% Ganho PathSim	29,19151683	-15,71818	18,70116368	5,829054388	12,58199545
			% Ganho PathSimc	33,14297492	-14,867102	22,16320267	8,453945953	12,78763982

Tabela B.3: Resultados dos algoritmos com remoção de afixos para consultas efetuadas em fontes com caminhos do mesmo domínio do caminho

Função	Encontrados	Existentes	Retornados	Precisão	Revocação	Média Harmônica	Medida F (P=0.2,R=0.8)	Precisão Média Por Resultado
Baaseline	1913	1913	16301	0,119193174	1	0,210433285	0,393677504	0,275124293
PathSim	1629	1913	11381	0,156328927	0,8577903	0,254986578	0,425978554	0,315688035
PathSimc	1649	1913	11226	0,160468349	0,86653876	0,261548366	0,435690196	0,312613619
			% Ganho PathSim	31,15593921	-14,22097	21,17217027	8,204952076	14,74378786
			% Ganho PathSimc	34,62880774	-13,346124	24,29039733	10,67185503	13,62632343

Tabela B.4: Resultados dos algoritmos com remoção de nomes de elementos duplicados para consultas efetuadas em fontes com caminhos do mesmo domínio do caminho

Função	Encontrados	Existentes	Retornados	Precisão	Revocação	Média Harmônica	Medida F (P=0.2,R=0.8)	Precisão Média Por Resultado
Baaseline	1913	1913	15914	0,122005888	1	0,214915087	0,400257272	0,273251336
PathSim	1459	1913	9253	0,206846005	0,77431753	0,279960864	0,431068531	0,330772825
PathSimc	1489	1913	9645	0,188303074	0,78739288	0,270604546	0,428417761	0,329566578
			% Ganho PathSim	69,53772346	-22,568247	30,26580297	7,697863568	21,05076222
			% Ganho PathSimc	54,3393332	-21,260712	25,91230787	7,035597095	20,60931993

Tabela B.5: Resultados dos algoritmos com todas as funções de pré-processamento para consultas efetuadas em fontes com caminhos do mesmo domínio do caminho

Função	Encontrados	Existentes	Retornados	Precisão	Revocação	Média Harmônica	Medida F (P=0.2,R=0.8)	Precisão Média Por Resultado
Baaseline	1913	1913	15939	0,121905768	1	0,214757777	0,400035847	0,28616497
PathSim	1499	1913	9307	0,20987396	0,79400385	0,28696773	0,443142143	0,346671757
PathSimc	1525	1913	9625	0,197123198	0,80500103	0,281862871	0,442666699	0,34684572
			% Ganho PathSim	72,16081153	-20,599615	33,62390631	10,77560842	21,14402297
			% Ganho PathSimc	61,70128865	-19,499897	31,24687508	10,65675802	21,20481427

Tabela B.6: Resultados dos algoritmos sem pré-processamento para consultas efetuadas em fontes com caminhos de vários domínios

Função	Encontrados	Existentes	Retornados	Mean Precision	Revocação	Mean Harmonic	Mean F Measure	Mean Average Precision
Baaseline	1913	1913	41213	0,059341655	1	0,109879868	0,227346218	0,255091269
PathSim	1597	1913	11760	0,147458958	0,841995153	0,241564919	0,40777886	0,299092281
PathSimc	1617	1913	12209	0,147755331	0,850505936	0,241908647	0,408336883	0,29859288
			% Ganho PathSim	148,4914821	-15,8004847	119,8445661	79,36469917	17,2491249
			% Ganho PathSimc	148,9909171	-14,9494064	120,1573877	79,61014993	17,05335144

Tabela B.7: Resultados dos algoritmos com remoção de caracteres especiais para consultas efetuadas em fontes com caminhos de vários domínios

Função	Encontrados	Existentes	Retornados	Precisão	Revocação	Média Harmônica	Medida F (P=0.2,R=0.8)	Precisão Média Por Resultado
Baaseline	1913	1913	41751	0,058673674	1	0,108690762	0,225142833	0,256106291
PathSim	1599	1913	11775	0,147478365	0,842818199	0,241621927	0,407958776	0,300178376
PathSimc	1619	1913	12289	0,148231531	0,851328981	0,24263343	0,409192863	0,299787033
			% Ganho PathSim	151,3535541	-15,7181801	122,3021739	81,19998338	17,20851324
			% Ganho PathSimc	152,6372066	-14,8671019	123,2327988	81,74811854	17,05570833

Tabela B.8: Resultados dos algoritmos com remoção de afixos para consultas efetuadas em fontes com caminhos de vários domínios

Função	Encontrados	Existentes	Retornados	Precisão	Revocação	Média Harmônica	Medida F (P=0.2,R=0.8)	Precisão Média Por Resultado
Baaseline	1913	1913	41317	0,060042157	1	0,110727556	0,227906609	0,266617582
PathSim	1629	1913	11876	0,148569127	0,857790305	0,245082245	0,415051035	0,316079421
PathSimc	1649	1913	12049	0,150532149	0,866538758	0,248144343	0,419721448	0,312827577
			% Ganho PathSim	147,4413552	-14,2209695	121,3380787	82,1145235	18,55160452
			% Ganho PathSimc	150,7107614	-13,3461242	124,1035131	84,16378965	17,33193837

Tabela B.9: Resultados dos algoritmos com remoção de nomes de elementos duplicados para consultas efetuadas em fontes com caminhos de vários domínios

Função	Encontrados	Existentes	Retornados	Precisão	Revocação	Média Harmônica	Medida F (P=0.2,R=0.8)	Precisão Média Por Resultado
Baaseline	1913	1913	40697	0,06248232	1	0,114794899	0,234742248	0,262814203
PathSim	1459	1913	9668	0,196027456	0,774317527	0,268478607	0,420272945	0,329242333
PathSimc	1489	1913	10462	0,175735961	0,787392881	0,254774897	0,410975418	0,330186289
			% Ganho PathSim	213,7326783	-22,5682473	133,8767744	79,03592071	25,27570019
			% Ganho PathSimc	181,2570996	-21,2607119	121,9392144	75,07518204	25,63487256

Tabela B.10: Resultados dos algoritmos com todas as funções de pré-processamento para consultas efetuadas em fontes de vários domínios

Função	Encontrados	Existentes	Retornados	Precisão	Revocação	Média Harmônica	Medida F (P=0.2,R=0.8)	Precisão Média Por Resultado
Baaseline	1913	1913	41578	0,062407084	1	0,114440317	0,233388431	0,275556232
PathSim	1499	1913	9792	0,197413326	0,794003849	0,273332493	0,430201122	0,343957474
PathSimc	1525	1913	10442	0,183018293	0,805001034	0,265090687	0,424688601	0,344402582
			% Ganho PathSim	216,3315979	-20,5996151	138,8428311	84,32838344	24,82297043
			% Ganho PathSimc	193,2652533	-19,4998966	131,6409933	81,96643218	24,98450117