MARCO ANTÔNIO FERREIRA CURI

SIMILAR

UMA APLICAÇÃO PARA CONSULTAS POR SIMILARIDADE EM ESTRUTURAS XML

FLORIANÓPOLIS
Junho 2012

MARCO ANTÔNIO FERREIRA CURI

SIMILAR

UMA APLICAÇÃO PARA CONSULTAS POR SIMILARIDADE EM ESTRUTURAS XML

Trabalho de conclusão de curso
apresentado pelo acadêmico
Marco Antônio Ferreira Curi
à banca examinadora do Curso
de Graduação em Sistemas de
Informação da Universidade Federal de
Santa Catarina como requisito parcial
para obtenção do título de Bacharel em
Sistemas de Informação.

Orientadora:
Prof. Carina Friedrich Dorneles
UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

MARCO ANTÔNIO FERREIRA CURI

SIMILAR

UMA APLICAÇÃO PARA CONSULTAS POR SIMILARIDADE EM ESTRUTURAS XML

Orientadora:

_____

Profª. Drª. Carina Friedrich Dorneles

Banca Examinadora:

_____

Prof. MSc. Rodrigo Goncalves

_____

Prof. Dr. Ronaldo dos Santos Mello

FLORIANÓPOLIS

Junho 2012

AGRADECIMENTOS

Agradeço aos meus pais pelo incentivo aos meus estudos, vocês são os grandes culpados pelo meu sucesso. Agradeço aos meus amigos e colegas de faculdade, pois sem eles a faculdade não teria sido a mesma. Agradeço à minha orientadora Carina pelo apoio e orientação durante a execução deste trabalho.

RESUMO

Sistemas de buscas por similaridade foram desenvolvidos a partir da necessidade da pesquisa por informação de uma maneira inteligente, ou seja, através de consultas aproximadas utilizando algoritmos de similaridade são aplicadas na busca dos dados. Sabendo que o formato dos dados nas bases não é fixo e que existe a possibilidade de que a palavra-chave utilizada na pesquisa não seja exatamente aquilo que esteja armazenado, a chance de que a busca não retorne registros compatíveis ou que retorne registros completamente diferentes é grande. Sabendo desta deficiência, foram criadas ferramentas de buscas tentando contornar este problema. Porém, as ferramentas de consulta atuais baseiam-se em palavras-chave para recuperar e ranquear o conteúdo desejado pelo usuário. Este trabalho propõe uma nova ferramenta de busca que permite efetuar pesquisas em documentos XML utilizando diversas funções de similaridade. A forma de pesquisa pode ser feita usando somente palavras-chave ou utilizando o conhecimento da estrutura XML armazenada, podendo pesquisar estritamente por uma ou mais *tags* XML. Os resultados destas funções de similaridade são exibidos ao fim da consulta de uma forma clara para o usuário, onde os mesmos são filtrados por um ponto de corte (*threshold*), trazendo os melhores resultados obtidos de cada documento XML. A ferramenta também disponibiliza uma forma de visualizar os melhores resultados obtidos em cada função de similaridade utilizada na pesquisa e ainda uma forma fácil de visualizar a *string* encontrada no documento original. A aplicação Similar foi criada e nela foram executados vários experimentos iniciais onde comprovaram sua eficiência que indicaram alguns trabalhos futuros para o projeto.

Palavras-chave: Buscas por similaridade, ferramenta de busca, *Documentos*, *XML*.

LISTA DE FIGURAS

Figura 1 - Métrica Jaro.....17
Figura 2 - Métrica JaroWinkler 18
Figura 3 - Arquitetura do Similar.....27
Figura 4 - Dewey.....29
Figura 5 – Tela principal, SIMILAR.....30
Figura 6 – Diagrama de Classes.....32
Figura 7 - Tela carga dos dados.....33
Figura 8 - Arvore dewey.....33
Figura 9 - Tela de consulta por palavra-chave.....34
Figura 10 - Tela de consulta por tag e palavra-chave.....34
Figura 11 - Tela de consulta por multi-tags e palavra-chave.....34
Figura 12 - Funções de similaridades básicas..... .36
Figura 13 - Menu de funções de similaridades avançadas.....36
Figura 14 - Funções de similaridades avançadas.....36
Figura 15 - Menu de Threshold.....37
Figura 16 - Tela de Threshold.....38
Figura 17 - Menu de número máximo de registros por arquivo.....38
Figura 18 - Tela de número máximo de registros por arquivo.....39
Figura 19 - Menu de número máximo de resultados selecionados.....39
Figura 20 - Tela de número máximo de resultados selecionados.....40
Figura 21 - Tela de resultados da consulta.....41
Figura 22 - Tela de visualização do documento..... 42

SUMÁRIO

1. Introdução..... 5
2. Fundamentação teórica.....8
2.1 Consulta por palavra-chave a documentos XML 8
2.2 Classificação das formas de busca 9
3. Busca por similaridade.....10
3.1 Métricas para cálculo de similaridade.....10
3.2 Trabalhos relacionados.....13
3.2.1 Suporte a argumentos de consulta vagos através da linguagem XPath.....13
3.2.2 Eris.....15
3.2.3 XSimilarity.....16
3.2.4 XSearch.....16
4. Aplicação Similar.....18
4.1 Visão geral 18
4.2 Arquitetura do Similar.....19
4.3 Dewey 20
4.4. Processo de consulta.....21
5. Implementação ..... .23
5.1. Projeto.....30
5.2 Diagrama de Classes.....31
5.3 Carga dos Dados.....32
5.4 Conceitos e tecnologias.....33
5.5 Interface de consulta..... ..35
5.6 Resultados..... .38
5.7 Visualização dos resultado no documento XML..... .43
5.8 Teste de desempenho..... .44
5.8 Experimentos..... .45
6. Conclusões e Trabalhos futuros.....48
7. Referências 50

1. Introdução

A intensificação do uso da Internet tem feito com que milhões de dados sejam trocados diariamente. Tais dados são, em sua maior parte, criados e manipulados diretamente por pessoas, contendo erros de vários tipos. O formato não é rígido e permite grandes variações: datas podem ser escritas de várias maneiras, como *dia/mês/ano* ou *mês/dia/ano*, *ano* com quatro ou dois dígitos, ou mês como *12* ou *dezembro*. Existem muitos exemplos de dados que podem ser expressos por diferentes formas, como nomes, principalmente de pessoas, que podem apresentar iniciais, abreviações, particularidades de cada idioma, etc. Todos esses formatos estão corretos, mas são difíceis de serem pesquisados por uma busca simples, utilizando somente palavras-chave.

A solução mais simples seria aplicar regras restringindo as possibilidades de armazenar uma informação, como normalmente é feito em bancos de dados, onde existe um formato fixo para o armazenamento de datas, por exemplo. No entanto, mesmo em bancos de dados tradicionais o armazenamento dos dados é uma tarefa que exige intenso planejamento e várias estruturas de controle. Uma solução desse tipo se torna praticamente impossível quando direcionada para um campo tão vasto e dinâmico como a Internet, dada à dificuldade que seria manter um formato fixo para os dados existentes.

A diversidade de formatos dos dados faz com que se procurem mecanismos de busca que facilitem cada vez mais a utilização pelo usuário para encontrar o que deseja em meio a grande quantidade de dados. Os mecanismos tradicionais de pesquisa em banco de dados utilizam uma busca por coincidência exata. Em uma base de dados de cadastros de usuários, por exemplo, “20/06/2010” e “06 - 20 - 2010” são considerados objetos diferentes. Mas ambos os objetos fazem referência a uma mesma data. A recuperação da informação por meio de consultas exatas pode ser ineficaz neste caso. Portanto, torna-se necessária a utilização de um mecanismo de pesquisa mais elaborado, com suporte a consultas imprecisas ou aproximadas, também conhecidas como consultas por similaridade.

Consultas por similaridade permitem o uso de argumentos vagos na pesquisa, isto é, a palavra-chave a ser consultada não precisa necessariamente ser idêntica à informação a qual o usuário esteja procurando. O resultado de tais consultas são valores

similares aos solicitados pelo usuário. As funções que calculam a similaridade entre dados usualmente retornam um valor numérico indicativo dessa similaridade. Este valor pode ser usado para ordenação e até mesmo para restringir o conjunto de resultados, através do uso de *threshold.* O cálculo de similaridade é realizado tendo em vista dois elementos distintos: elementos atômicos, que contêm como valor *strings* (nomes, datas, endereços, etc.) e elementos complexos, que possuem nos seus valores outros elementos; estes elementos são encontrados em documentos XML (Extensible Markup Language é uma recomendação da W3C para gerar linguagens de marcação com o propósito principal de facilitar o compartilhamento de informações através da internet) [W3C 2006].

A internet possui um grande volume de dados e isso motivou o desenvolvimento de mecanismos de busca de informações em meio a este grande volume. Surgiram várias empresas como Cadê, Yahoo, Google, entre outras. Porém os mecanismos tradicionais de busca não consideram a estrutura do documento XML. Nestes documentos, buscas são feitas por meio de consultas em XQuery [W3C s.d.] (linguagem que permite realizar consultas sobre documentos XML), ou buscas por coincidência exata (palavra-chave) que não suportam consultas por similaridade. Na XQuery, há trabalhos que introduzem funções de similaridade no processador da linguagem (XSimilarity [Silva, Borges e Galante 2008]). Em busca por palavra-chave não há propostas de consulta por similaridade a XML, considerando sua estrutura. Portanto, surge a necessidade de se desenvolver um mecanismo de pesquisa mais elaborado, com suporte a consultas aproximadas a documentos XML [Padilha 2005].

A proposta deste trabalho é desenvolver um sistema de busca por palavra-chave e por similaridade em XML, que auxilie o usuário a encontrar respostas que representem o mesmo objeto do mundo real daquele usado por ele na consulta. A aplicação proposta permite ao usuário carregar documentos XML, onde cada documento será mapeada em memória para utilizar os dados nas pesquisas. O principal objetivo do trabalho é desenvolver uma ferramenta de busca capaz de efetuar consultas por similaridade, usando um ou mais elementos (ou atributos) da estrutura do documento juntamente com a palavra-chave informada, que seria referente ao valor contido no elemento. Os parâmetros informados na consulta são processados utilizando uma ou mais funções de similaridade especificadas na aplicação, onde cada uma delas

determina o resultado a ser pesquisado. Como front-end da aplicação, deve ser disponibilizada uma tela com campos que permitam ao usuário efetuar suas ações. A aplicação carrega a base de dados XML em memória possibilitando um acesso rápido aos dados na consulta baseada na palavra-chave informada pelo usuário, utilizando métodos de pesquisa por similaridade que o usuário desejar efetuar. O resultado da consulta será uma lista ranqueada (lista dos maiores índices de semelhança) dos elementos encontrados na base de dados em questão.

O projeto está distribuído em diversas seções, onde as próximas se referem à fundamentação teórica do trabalho, seção 2, seguido da seção 3, que apresenta os principais temas e trabalhos que estimularam o desenvolvimento do projeto. A apresentação da ferramenta do projeto é feita na seção 4, enquanto as técnicas de carga e desenvolvimento da ferramenta Web são expostas na seção 5. Por fim, na seção 6 são apresentados as conclusões obtidas e os trabalhos futuros do projeto.

2. Consulta por palavra-chave a documentos XML

Consultas por palavras-chave permitem o acesso fácil a dados XML, uma vez que não exigem que o usuário aprenda uma linguagem de consulta estruturada nem estude possíveis esquemas de dados complexos. Com isso, vários motores de busca XML foram propostos para permitir a extração de fragmentos XML relevantes para consultas por palavras-chave. No entanto, algumas abordagens foram propostas para permitir que consultas aproximadas fossem executadas em SGBDs relacionais [Calado 2000, Borges e Cony 2005, Borges e Dorneles 2006]. Outros trabalhos estendem as linguagens de consulta a documentos XML [Padilha 2005].

Entretanto, existe uma lacuna nos SGBDs XML, como, por exemplo, eXist, Tamino, XIndice, entre outros, quanto ao suporte a consultas aproximadas. Torna-se evidente, portanto, o interesse em desenvolver mecanismos de consulta por similaridade em bases XML. As linguagens XPath [W3C 1999] e XQuery [W3C s.d.], desenvolvidas especificamente para trabalhar sobre XML, são uma alternativa para a inclusão de tal mecanismo. Porém, para fazer uma pesquisa em um banco de dados XML é necessário um conhecimento destas linguagens para poder criar uma consulta que permita efetuar a busca dos dados. Fazer uma pesquisa por palavra-chave usando elementos (ou atributos) do documento XML seria uma boa alternativa para usuários leigos, mas possui um problema: é necessário conhecer a estrutura do documento a ser pesquisado na base de dados.

Para buscas em busca XML, a estrutura do documento é usada para determinar fragmentos do documento que são mais significativos para retornar como resposta da consulta. Também é usada para especificar as condições de consulta sobre a estrutura que limitam o contexto da pesquisa para elementos específicos do documento XML, ao contrário de documentos inteiros.

Os mecanismos de consulta por similaridade podem ser classificados nas linguagens existentes para pesquisa em documentos XML, de acordo com o uso de condições de estrutura do documento: *Keyword-Only, Tag and Keyword Queries, Path and Keyword Queries, XQuery and Keyword Queries.* Este trabalho explora mais o tipo de consulta “Tag and Keyword Queries” (Consultas por palavras chave e tag ou

"rótulo"), porque não há necessidade do usuário da ferramenta ter o conhecimento de linguagens como o XPath ou XQuery para poder efetuar uma consulta, necessitando apenas o conhecimento da estrutura do documento XML, facilitando o acesso a estes usuários.

2.1 Classificação das formas de busca

As linguagens propostas para pesquisa em documentos XML variam de uma busca por palavra-chave simples a buscas envolvendo algoritmos de similaridades combinados com a estrutura do documento XML [Padilha 2005]. A estrutura do documento desempenha um papel importante na concepção e semântica destas linguagens. Quando se envolve mais de um documento a linguagem pode retornar da pesquisa fragmentos de documentos, em oposição aos documentos inteiros.

Para buscas em XML, a estrutura do documento é usada para determinar nome dos usuários, os fragmentos que são documentos mais significativos para retornar como respostas da consulta. É também usada para especificar condições de consulta sobre a estrutura que delimite o contexto da pesquisa de elementos específicos do XML, ao contrário aos documentos inteiros. A classificação das linguagens existentes para consultas em documentos XML variam de acordo com a estrutura do documento (Amer-Yahia, 2006):

- *Keyword-Only Queries*: Linguagem natural de evolução da tradicional linguagem de recuperação de informação que são expressas com um conjunto de palavras-chave. A principal característica desta linguagem é a sua capacidade de retornar partes do documento XML;
- *Tag and Keyword Queries*: Linguagens que incluem o XSEarch (Cohen, Mamou, Kanza e Sagiv; 2003), permite fazer anotações por palavras-chave na consulta utilizando *tags (labels)* do documento XML;
- *Path and Keyword Queries*: Linguagens como a XPath que apresentam condições mais sofisticadas sobre a estrutura de dados semi-estruturados;

- *XQuery and Keyword Queries*: Linguagens que combinam o poder do XQuery e uma série de elementos complexos (cadeia de *strings*) de texto completo (em oposição à simples booleanos).

A ferramenta *Similar* desenvolvida neste trabalho utiliza duas das classificações de linguagens apresentadas (*Keyword-Only Queries* e *Tag and Keyword Queries*) nas suas formas de pesquisa. Com estas linguagens, o *Similar* disponibiliza uma forma de busca mais completa permitindo a pesquisa ser feita somente por uma palavra-chave sem a necessidade de conhecer a estrutura do documento XML, ou utilizando o conhecimento da estrutura usando *tag* ou multi-*tags* (*labels*) na busca, restringindo o escopo dos dados pesquisados.

3. Busca por similaridade

Os mecanismos de busca em SGBDs relacionais deveriam recuperar como resultado todas as representações de um determinado objeto, independente do argumento de consulta utilizado. Além disso, é possível encontrar múltiplas convenções em atributos, como nome e endereço, onde a ordem das palavras pode estar invertida, dependendo da base a ser pesquisada. Da mesma forma, o usuário pode desejar conhecer uma determinada informação fornecendo uma lista ou conjunto de valores. Por exemplo, um usuário deseja saber quais artigos foram escritos por determinados autores. Se a ordem em que os autores aparecem é importante, então os argumentos da consulta formam uma lista, senão, um conjunto. É importante notar que, além de utilizar dois argumentos, a consulta exige que esses argumentos encontrem-se agrupados (lista ou conjunto), não obstante o fato de que eles podem ter sido informados com erros de digitação ou podem estar de forma abreviada na base. Existem casos em que a cadeia de caracteres, digitada na formulação da consulta, seja um padrão de fato, ou seja, não contenha nenhum tipo de erro. O erro pode estar na base a ser pesquisada. Diversos tipos de erros podem ser encontrados como, por exemplo, erros de datilografia, fonéticos ou na transmissão de dados.

3.1 Métricas para cálculo de similaridade

As métricas de similaridade apresentam um amplo conjunto, tanto para elementos atômicos, como para complexos (DORNELES et al., 2004). Métricas para elementos atômicos derivam do tipo de dado e a parte semântica da informação. Dentre as métricas para elementos complexos existe uma que trata da similaridade para acrônimos, onde os conjuntos de caracteres (conjunto, lista e tupla) são comparados, e a outra forma é na filtragem de textos. No segundo caso, as métricas utilizam palavras representativas, geralmente retiradas de um dicionário, ou baseiam-se em mecanismos de busca por palavra-chave. Nesse caso, é necessário que o texto a ser filtrado passe por um algum tipo de indexação, onde os termos mais representativos são selecionados e contados. Só depois desse procedimento é que são utilizadas métricas de filtragem. Algumas métricas são melhores explicadas abaixo.

N-grams

A técnica de n-grams consiste basicamente em se dividir uma cadeia de caracteres em subcadeias de caracteres de tamanho n e compará-la com as subcadeias de caracteres de outra palavra a ser pesquisada (ULLMANN, 1977). Porém, essa técnica foi proposta com as restrições de que as palavras não poderiam ter mais de duas diferenças (inserção, alteração ou exclusão de caracteres para transformar uma em outra) e que todas elas seriam comparadas a um dicionário com palavras de seis caracteres.

Mais adiante, em 1992, Ukkonen (UKKONEN, 1992) elabora uma técnica conhecida como q-grams, que visa localizar, com agilidade, cadeias de caracteres em grandes conjuntos de caracteres. Essa é uma das abordagens existentes para filtragem de textos e consiste em separar o texto em vários grânulos de tamanho q. Posteriormente, separa-se também uma palavra padrão nesses mesmos grânulos e então se procura por ocorrências dos grânulos dessa palavra no texto. A localização de m-q+1 grânulos do padrão no texto significa um casamento (match), onde m representa o tamanho da palavra, em caracteres.

A métrica ainda utiliza, além das duas cadeias de caracteres que serão comparadas, o tamanho do grânulo (gram), que é o valor de n, e o número de caracteres especiais que serão colocados no início e no fim das cadeias de caracteres, com a finalidade de melhorar a quantificação da similaridade. Com n=3, seria criado o seguinte conjunto de grânulos (o caracter "_" significa espaço em branco):

Porto Alegre, RS														
_po	por	ort	rto	to_	o_a	_al	ale	leg	egr	gre	re_	**e_r**	_rs	rs_
RS- Porto Alegre														
_rs	rs_	**s_p**	_po	por	ort	rto	to_	o_a	_al	ale	leg	egr	gre	re_

Acronym

Para tentar solucionar a comparação entre duas cadeias de caracteres, onde uma delas pode estar de forma abreviada, foram pesquisadas métricas que solucionassem o problema de acrônimos. Uma solução para esse problema foi encontrada em (LIMA, 2002), onde a métrica proposta, primeiramente, executa uma varredura em cada cadeia de caracteres eliminando caracteres, que não sejam alfanuméricos. Posteriormente, a(s) cadeia(s) de caracteres que não possui abreviaturas é reduzida a seus acrônimos, ou seja, ficam apenas as iniciais. Finalmente, é aplicada a métrica de Guth (GUTH, 1976) sobre as cadeias de caracteres e então um escore de similaridade é retornado.

Para formar os acrônimos, são levadas em consideração as letras maiúsculas que iniciam as palavras, bem como palavras maiores que dois caracteres (caso a cadeia de caracteres tenha mais de uma palavra). Com esta última medida, evita-se a ocorrência das preposições mais comuns em nomes próprios, e de artigos na composição de uma abreviatura. Caso a cadeia de caracteres seja composta por apenas uma palavra, a métrica a considerará um acrônimo. Um exemplo é mostrado a seguir:

GM :
General Maintenance
Great Mistake
Garbage Motors
Generally Miserable
Grossly Misconceived
Gluteus Maximus

Guth

A métrica criada por Gloria Guth (GUTH, 1976) compara duas cadeias de caracteres, letra a letra, procurando um casamento. Quando duas letras, na mesma posição, são diferentes, ela procura por essa letra em outras posições. Esse algoritmo é ideal para comparar nomes próprios e surgiu como alternativa para métricas que comparavam duas palavras levando em consideração o seu conteúdo fonético. No entanto, essa métrica retorna o número de casamentos feitos entre as duas cadeias de caracteres.

Números

Uma função de similaridade para números foi encontrada apenas em (DORNELES et al., 2004). Essa função calcula a similaridade entre dois números (positivos ou negativos), dividindo a sua diferença absoluta pelo *average*, que é a média dos números encontrados dentro do conjunto de números em questão.

Métricas de Datas

Da mesma maneira que a similaridade para acrônimos, praticamente não existem trabalhos na tentativa de solucionar o problema de similaridade entre datas. Uma das poucas alternativas encontradas em (LIMA, 2006) verifica a similaridade entre datas, mesmo estando em diferentes formatos. Existem algumas variações, yearSim, monthSim e daySim, que usam a mesma métrica de datas completas, entretanto aplicadas respectivamente a ano, mês e dia somente.

A métrica que realiza a comparação entre as datas, primeiramente, faz o reconhecimento da data, que se encontra na base e pode estar em formatos diferentes, transformando-a em um padrão reconhecido pela linguagem. É importante ressaltar que a data no formato "mm/dd/aaaa" só será reconhecida nesta forma se o dia for maior que 12, senão o padrão é "dd/mm/aaaa". Posteriormente, ambas as datas, juntamente com a data atual, são passadas para o calendário Juliano, através de um algoritmo que pode ser visto em (BAUM, Acesso em: 30 jun. 2005). Dessa forma todas as datas que serão utilizadas na métrica de similaridade para datas possuem um valor numérico linear.

<data>2005/3/14</data>
<data>14 de janeiro de 2005</data>
<data>14 janeiro, 2005</data>
<data>january 14, 2005</data>
<data>4/14/2005</data>
<data>12-28-2005</data>
<data>2000.12.10</data>
<data>31/07/2005</data>

Métricas de ligação de registros

Métricas de ligação de registros visam comparar um grande conjunto de dados com outro em busca de registros que façam parte de ambos. Para isso são considerados os pesos dos campos dos registros (ex. CPF possui um peso maior do que idade), bem como a possibilidade de existirem pequenos erros tipográficos.

A métrica de Jaro-Winkler (WINKLER, 1999) é um aprimoramento da métrica de Jaro (JARO, 1995). Em seu trabalho, Jaro propôs um método de ligação probabilístico com a finalidade de comparar dois arquivos para localizar registros iguais ou com alto grau de similaridade. Ela foi elaborada para realizar o casamento dos arquivos da saúde pública dos Estados Unidos. Resumidamente, para duas cadeias de caracteres *s* e *t*, considere *si;j* o conjunto de caracteres em *s* que também estão em comum em t e considere *ti;j* os caracteres em *t* os quais estão em comum em *s*. Um caractere em *s* está em comum em *t* quando ele está próximo à posição em que se encontra em *t*. Considere *Tsi;j ;ti;j* a medida do número de transposições de caracteres em *si;j* relativos a *ti;j* . A métrica de similaridade de Jaro para *s* e *t* é:

Figura 1. Métrica Jaro

$$Jaro(s,t) = \frac{1}{3} * (\frac{|s_{i,j}|}{|s|} + \frac{|t_{i,j}|}{|t|} + \frac{|s_{i,j}| - T_{s_{i,j},t_{i,j}}}{2|s_{i,j}|})$$

Winkler propôs uma extensão à métrica de Jaro, inserindo *P* como o tamanho do maior prefixo comum entre as cadeias de caracteres *x* e *y*, ficando da seguinte forma:

Figura 2. Métrica JaroWinkler

$$JaroWinkler(s,t) = Jaro(s,t) + \frac{P}{10} * (1 - Jaro(s,t))$$

Métricas de casamento de coeficientes

Algumas métricas são baseadas em coleções de palavras (ou tokens). Essas técnicas se destacam por serem aplicadas em cadeias de caracteres formadas por várias subcadeias de caracteres. Elas baseiam-se na distância vetorial das palavras e são muito utilizadas na comunidade de recuperação de informação. A métrica mais simples é o

coeficiente de casamento (Matching Coefficient). Essa métrica baseia-se em contar o número de palavras em comum, sem levar em consideração o tamanho das cadeias de caracteres.

A métrica de similaridade Cosine é largamente utilizada na comunidade cientifica (COHEN; RAVIKUMAR; FIENBERG, 2003). Ela se baseia no tamanho do ângulo formado entre os vetores utilizados na comparação. Quanto menor o ângulo, maior a similaridade entre as palavras. O tamanho dos vetores pode ser reduzido com a retirada das stop words (palavras que não identificam um documento como artigos, preposições, etc).

Métricas linguísticas

As métricas linguísticas pretendem resolver erros da compreensão humana das palavras, tanto no modo como elas são escritas (Smith ou Smyth), como na sua formação fonética (Sinclair ou St Clair) sendo muito utilizadas no auxílio à identificação de nomes. Aqui será descrita, brevemente, uma destas métricas.

A métrica Russel Soundex (LAIT; RANDELL, 1993) foi desenvolvida para resolver problemas fonéticos no casamento de nomes da língua inglesa. Ela converte cada nome em um código de quatro caracteres, sendo o primeiro uma letra e os outros três, números. O código é utilizado para identificar nomes equivalentes. Esse método possui adaptações para outras línguas como francês e alemão, porém não foram encontradas adaptações para o português.

3.2 Trabalhos relacionados

3.2.1 Suporte a argumentos de consulta vagos através da linguagem XPath

XML Path Language (XPath [W3C 1999]) é uma linguagem de programação que permite construir expressões que percorrem e processam um documento XML de modo parecido a uma expressão regular. Existem algumas propostas como em (MOTRO, 1988;GRAVANO, 2001) para suporte a argumentos de consulta vagos através da extensão da linguagem XPath. Para isso, são utilizadas expressões XPath que utilizam novas funções, as quais são, diretamente, adicionadas ao processador da linguagem de consulta.

Trabalhos como o de (Padilha 2005) propõem adicionar funções a um processador XPath que possuem uma ligação muito estreita com as métricas utilizadas. Como as métricas, as funções trabalham com valores simples (elementos atômicos) e compostos (elementos complexos). As funções que trabalham com elementos atômicos podem ser classificados tanto pelo tipo de dado que será analisado, como pelo tipo de análise que será feita. As funções para elementos complexos comparam conjuntos de elementos atômicos de acordo com a forma do agrupamento (conjunto, lista ou tupla).

As funções de similaridade que a linguagem XPath disponibiliza são definidas com base em algumas métricas de similaridade para elementos XML definidas em (DORNELES et al., 2004), que se aplicam tanto a elementos com valores atômicos, quanto a elementos com valores complexos. As funções para elementos atômicos comparam alguns tipos básicos de dados (cadeia de caracteres, data e número) através de métricas de similaridade, podendo haver mais de uma para cada tipo. As funções para elementos complexos consideram o tipo de estrutura em que os elementos atômicos podem ser encontrados. Através do casamento da estrutura contida na consulta, a qual pode ser uma lista, um conjunto ou uma tupla, é feita a comparação dos elementos atômicos, contidos nestas estruturas, por meio das métricas de similaridade atômicas. Para listas e conjuntos, não só os tipos de dados devem ser o mesmo, como também o tipo do elemento, enquanto que, para tuplas, ambas as restrições não são necessárias.

A extensão do conjunto de funções já disponíveis na linguagem XPath, tem como contribuições:

- suporte a consultas com condições de busca aproximada, através da adição de novas funções para uma linguagem;
- definição de uma extensão, sem alteração da sintaxe, para uma linguagem padrão já adotada pela comunidade (XPath).

3.2.2 Eris

Eris é uma ferramenta destinada a realizar consultas por similaridade em bases de dados XML. A ferramenta foi implementada em Java e possibilita a escolha das funções de similaridade, bem como o *threshold* (índice de similaridade, pode variar de 0,2 a 0,9, valores mais altos de *threshold* são mais restritivos) usado para a criação do ranking dos resultados. Todas as instâncias constantes em uma base XML são processadas e comparadas com uma dada consulta, na tentativa de encontrar aquelas instâncias que mais se aproximam da requisição do usuário.

O cálculo de similaridade da ferramenta Eris é realizado tendo em vista dois elementos distintos: elementos atômicos e elementos complexos, tais como encontrados em documentos XML. Os elementos atômicos contêm como valor strings (nomes, datas, endereços, etc), já os elementos complexos contêm como valor outros elementos. Para o cálculo de elementos atômicos, definiram-se métricas para cada tipo de elemento. O cálculo de elementos complexos leva em conta a estrutura do elemento e o que o usuário deseja com a consulta. Os elementos complexos podem ser divididos em tuplas ou coleções. Tuplas são os elementos que contém elementos com nomes diferentes, enquanto coleções contêm elementos de mesmo nome. As coleções podem ser tratadas como listas, se a ordem em que as informações aparecem é importante para o usuário, ou como conjuntos, caso a ordem não importe. Também foi definido o conceito de subestrutura, onde informações parciais podem ser utilizadas na consulta. Um exemplo seria um subconjunto. Neste caso, a similaridade é calculada usando como parâmetros um elemento XML da base, considerado conjunto, e a consulta como parte dos membros deste conjunto, por exemplo, contendo o nome de alguns autores de um artigo, mas não todos.

3.2.3 XSimilarity

XSimilarity é uma ferramenta para a realização de consultas por similaridade embutida na linguagem XQuery, implementando funções de similaridade relevantes na área de banco de dados e recuperação de informações. As funções de similaridade são embutidas no processador de consultas da linguagem de consulta XQuery.

O XSimilarity pode ser aplicado na consulta a informações em ambientes integrados na *Web* como CiteSeer e bibliotecas digitais. Tais ambientes integram meta-dados de várias fontes, os quais geralmente estão estruturados em documentos XML, e consultas exatas não seriam suficientes para mostrar resultados satisfatórios. Além disso, a ferramenta desenvolvida soluciona uma das deficiências dos SGBDs XML no que tange à execução de consultas por similaridade utilizando XQuery.

A ferramenta permite a execução de consultas XQuery em que funções de similaridade podem ser embutidas através de um mecanismo de *namespace* (identificador único, usando para evitar conflitos de nomes nos documentos XML). O uso das funções de similaridade atreladas ao *namespace* possibilita a melhora da recuperação de informações nessas bases de dados.

3.2.4 XSearch

XSearch é um motor de busca semântico para XML e possui uma linguagem de consulta simples, uma extensão da linguagem descrita. A linguagem de consulta de um motor de busca padrão é simplesmente uma lista de palavras-chave. Em alguns motores de busca, cada palavra-chave pode, opcionalmente, ser precedido por um sinal de mais ("+"). Palavras-chave com um sinal de ("+") deve aparecer em um documento, enquanto palavras-chave sem um ("+") sinal pode ou não pode aparecer (mas o aparecimento de tais palavras-chave é desejável).

O XSearch permite especificar palavras-chave somente ou um combinado de palavras-chave com específicos *labels* (rótulos) que devem ou podem aparecer em um documento XML. O resultado de uma consulta é classificado usando técnicas de recuperação da informação e são ranqueados de acordo com o grau de similaridade.

Sobre a semântica da consulta, ela deve satisfazer cada uma das condições exigidas. Além disso, todos os elementos que satisfazem a condição da consulta devem possuir significados quando estão juntos. Para satisfazer de um termo de pesquisa nos modelos de documentos XML como árvores no formato padrão, cada nó interior é associado a uma etiqueta e cada nó folha é associado com uma sequência de

palavras-chave. Isto permite determinar que sub-árvores de um documento são mais relevantes, permitindo-nos a classificar corretamente os resultados da consulta.

3.2.5 Tabela comparativa entre as abordagens:

A Tabela 1 a seguir separa algumas características das propostas analisadas.

Tabela 1: Tabela comparativa entre os trabalhos.

	ERIS	XSIMILARITY	XSEARCH	SIMILAR
Busca por similaridade	X	X	X	X
Busca por elementos complexos	X	X	X	X
Busca por formato de palavra chave				X
Busca por diferentes tipos de funções de similaridade				X
Busca por Labels(tags)	X	X	X	X
Busca multi-níveis de labels (tags)		X		X
Suporte a XQuery		X		
Utiliza threshold	X			X
Interface visual	X	X	X	X
Múltiplos documentos XML		X		X
Ranking dos melhores resultados obtidos dentre as funções de similaridade				X
Interface de visualização dos resultados no documento XML				X

A Tabela 1 apresenta características importantes tratadas por cada um dos trabalhos, conforme descritas a seguir:

Busca por similaridade: Tecnologias que suportam a busca por similaridade.

Busca por elementos complexos: Tecnologias que suportam a busca por elementos complexos, que são os conjuntos, listas ou tuplas.

Busca por formato de palavra chave: Tecnologias que suportam a busca por palavra-chave, ou seja, efetuar uma busca procurando uma determinada string.

Busca por diferentes tipos de funções de similaridade: Tecnologias que suportam fazer uma busca com até dez tipos diferentes de funções de similaridade.

***Busca por Labels(tags)*:** Tecnologias que suportam a busca por labels(tags), restringindo na busca aos rótulos informadas.

Busca multi-níveis de labels (tags): Tecnologias que suportam a busca com mais de um nível de label, isto é, poder restringir uma pesquisa entre os níveis da estrutura do documento XML. Por exemplo; em uma estrutura XML onde são armazenados os dados cadastrais de alunos, necessita-se fazer uma busca entre os endereços dos alunos. Aluno -> endereço -> “palavra-chave”.

Suporte a XQuery: Tecnologias que suportam a linguagem XQuery.

Utiliza threshold: Tecnologias que suportam a utilização de threshold, fazendo com que a busca se restrinja aos índices indicados pelas funções de similaridade.

Interface visual: Tecnologias que disponibilizam uma interface visual para efetuar as consultas.

Múltiplos documentos XML: Tecnologias que permitem efetuar uma busca em diversos documentos XML.

Ranking dos melhores resultados obtidos dentre as funções de similaridade: Tecnologias que permitam a visualização do ranking (melhor colocação) dos melhores resultados obtidos no processo de busca dentre todas as funções de similaridade utilizadas.

Interface de visualização dos resultados no documento XML: Tecnologias que permitam a visualização exata do resultado obtido na busca em relação ao documento XML, sua localização na estrutura.

É importante enfatizar que o diferencial existente entre os trabalhos relacionados e a ferramenta desenvolvida neste trabalho (*Similar*) é a capacidade de efetuar uma busca por similaridade usando somente palavras-chave ou *tags* do documento juntamente com a palavra chave desejada, fazendo com que a busca seja mais precisa. Além da forma de busca, o *Similar* permite também uma comparação entre as funções de similaridade disponibilizadas, exibindo separadamente os resultados encontrados dentre cada função de similaridade utilizada; e ainda disponibiliza uma visualização dentro do aplicativo do resultado encontrado em relação ao documento XML.

4. Aplicação *Similar*

Este capítulo apresenta o funcionamento da aplicação *Similar*, uma ferramenta de busca que permite consultas por similaridade em documentos XML. A principal contribuição da ferramenta é permitir consultas não exatas, isto é, efetuar a pesquisa com certas variações da palavra-chave informada e permitir a construção de consultas com a utilização de combinações de elementos do documento XML com palavra-chave.

O capítulo está organizado como segue. Na seção 4, é apresentada uma visão geral do funcionamento da aplicação, a arquitetura do *Similar* e os processos de consulta de similaridades. Na seção 5, são apresentadas as etapas do processo de implementação da ferramenta, o processo de carga dos documentos, como é feito o processo de consulta e os resultados disponibilizados ao usuário.

4.1 Visão geral

A aplicação consiste em três partes principais. A primeira é a carga dos dados de uma base XML para a memória. A segunda é criar uma estrutura de indexação dos dados que facilite a busca. Finalmente a terceira é a busca dos dados com diferentes funções de busca dos dados por similaridade.

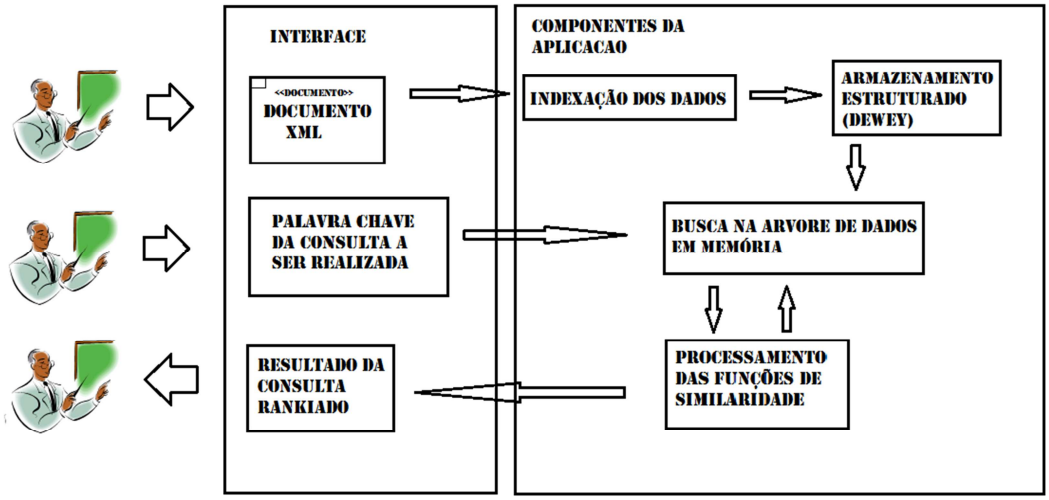
A primeira parte consiste em fazer a carga dos dados de uma base XML para memória. Caso a base não seja um documento XML ou esteja corrompida de alguma forma a aplicação mostra uma mensagem “Não foi possível fazer a carga da base de dados selecionada.”. Se a base estiver consistente é feita a carga dos dados com a biblioteca do Java chamada SAXParserFactory. Ela permite uma maior facilidade em manipular os elementos e tags do XML, fazendo com que seja possível criar uma estrutura de árvore DOM. Com os dados pré-carregados pela biblioteca SAXParserFactory, é iniciada a segunda parte do tratamento dos dados, onde é feito uma estrutura Dewey (Seção. 4.3) de rotulagem ID, para que os dados possam ser resgatados mais rapidamente quando for efetuada a busca por similaridade na estrutura. A terceira e última parte consiste dos diferentes tipos de pesquisa por similaridade que podem ser feitos na aplicação *Similar*. As funções utilizadas são as disponibilizadas pela

biblioteca do Java Simmetrics (SimMetrics, 2005), onde na primeira versão da ferramenta *Similar* utilizadas as seguintes: Jaro Winkler, Jaccard, Cosine, Levenshtein, QGrams Distance, Dice, Euclidean Distance, Needleman Wunch, Overlap Coefficient, Smith Waterman.

4.2 Arquitetura do *Similar*

A arquitetura da aplicação *Similar* baseia-se em seis etapas principais, mostradas na Figura 3.

Figura 3. Arquitetura do *Similar*.



A intervenção do usuário é feita de duas formas: na carga dos dados pelo arquivo XML e na inserção dos parâmetros da busca, seja pela palavra-chave apenas ou pela “tag” seguida da palavra-chave. Após o caminho do documento ser inserido pelo usuário, a aplicação faz a leitura do documento XML e faz o armazenamento do mesmo em uma estrutura de árvore (numeração Dewey) para aumentar a velocidade nas buscas dos dados.

Com os parâmetros da busca inseridos, o usuário escolhe o tipo de mecanismo de busca por similaridade que vai ser utilizado. Com os parâmetros e o mecanismo definidos, é feita a pesquisa dos dados na árvore de dados que está em memória caso a

pesquisa seja feita usando o elemento, então só será percorrido os dados que estão dentro do elemento informado ou em seus sub-elementos. Caso contrário é feita a varredura de todo o conteúdo do documento XML em busca de algum dado com grau de similaridade da palavra-chave. Ao final do processamento da consulta é retornado para o usuário uma lista com o ranking dos dados com maior grau de similaridade.

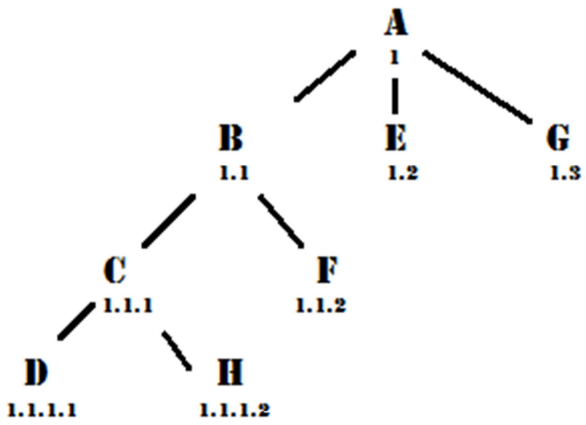
4.3 Dewey

O sistema Dewey ID de rotulagem proposto por (Tatarinov et al. 2002) apresenta a posição de um elemento em ocorrência a um documento XML. Em Dewey ID, cada elemento é apresentado por um vetor: (*i*) a raiz é rotulada por um string vazia €, (*ii*) para um elemento rótulo *u* non-root (que não é raiz), rótulo (*u*) = rótulo (*s*).*x*, onde *u* é o filho *x*-th de *s*. Dewey ID faz uma avaliação de relações estruturais entre os elementos do documento. Isto é, elemento *u* é um ancestral de elemento *s* se e somente se o rótulo (*u*) é um prefixo *x* de rotulo (*s*).

Dewey ID tem uma propriedade interessante: pode derivar ancestrais de um elemento a partir do seu rótulo. Para exemplificar, suponha que o elemento *u* é rotulado "1.2.3.4", então o pai de "*u*" é "1.2.3" e o avô é "1.2" e assim por diante. Com o conhecimento dessa propriedade, considera-se que os nomes de todos os ancestrais de "*u*" pode ser derivado do rótulo (*u*) sozinho, logo o caminho XML correspondente pode ser diretamente reduzido para comparação de strings. Por exemplo, se sabemos que o rótulo "1.2.3.4" apresenta o caminho "a / b / c / d", então é fácil identificar se o elemento corresponde a um padrão de caminho (por exemplo, " / c / d ").

A Figura 4 mostra um documento XML com exemplo de estrutura por rótulos de Dewey. Dado o rótulo "/ 1.1.1.2" do elemento do texto, podemos deduzir que o caminho da raiz ao elemento do texto é "/A/B/C/H/".

Figura 4. Dewey



O sistema Dewey ID de rotulagem é utilizado na ferramenta *Similar* na etapa de carga dos documentos XML usados para alimentar o aplicativo. Ao fazer a leitura dos dados do documento e o pré-processamento, os dados ficaram armazenados em memória na estrutura Dewey ID. Ela permite uma busca mais performática dos dados, pois quando a pesquisa é feita utilizando *tags* do documento o caminho da estrutura é percorrido até encontrar a *tag(s)* definida, eliminando assim os outros caminhos (“galhos”) da estrutura.

4.4 Processador de consulta por similaridade

A aplicação disponibiliza um campo texto para a inserção da palavra-chave que será usada na busca. O *Similar* permite ao usuário utilizar dois tipos de busca (Keyword-Only e Tag and Keyword Queries).

As três formas de consultas na aplicação são:

- A pesquisa por texto, por toda a árvore DOM, por exemplo, “Marco”;

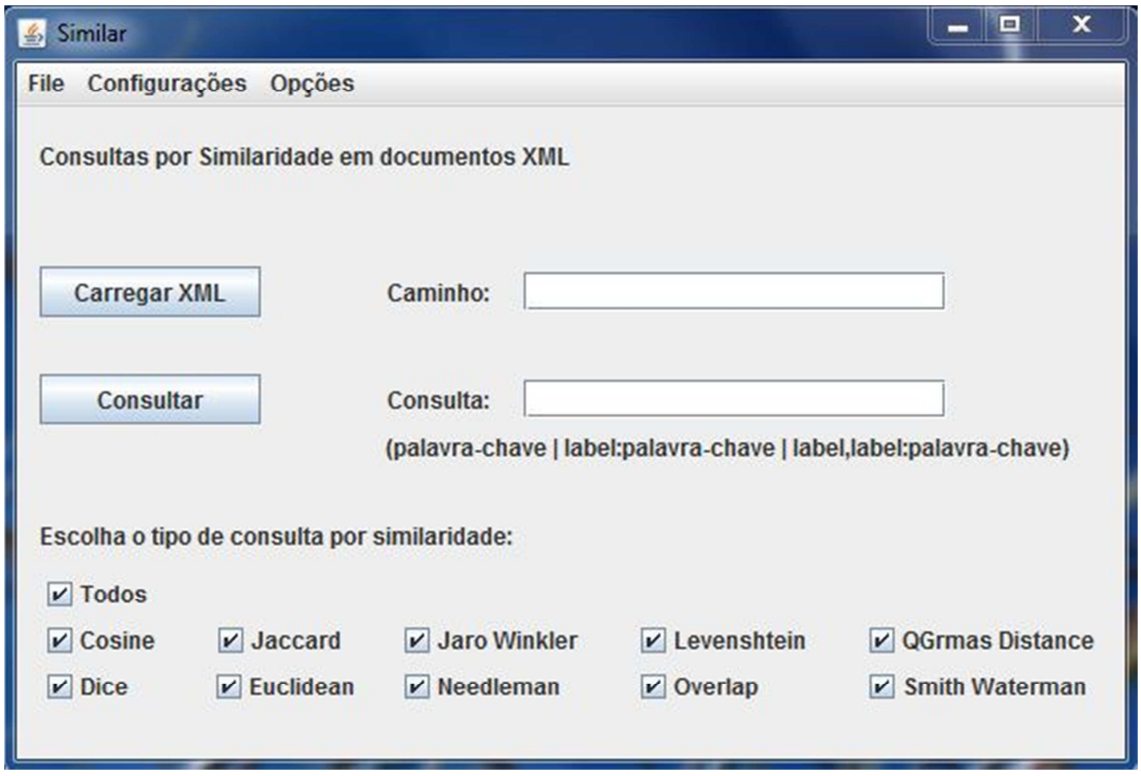
- A pesquisa direcionada a tags definidas pelo usuário, que por sua vez precisam estar presente no documento XML carregado, por exemplo, “Nome:Marco”;

- A pesquisa multi – tags, por exemplo, “Aluno,Nome:Marco”;

Existe a possibilidade de fazer a consulta usando dez tipos de funções de similaridade, sendo elas a Jaro Winkler, Jaccard, Cosine, Levenshtein, QGrams Distance, Dice, Euclidean Distance, Needleman Wunch, Overlap Coefficient, Smith Waterman. As funções possuem seu específico mecanismo de busca, isto é, possui sua própria lógica de similaridade onde cada uma delas possui vantagens em diferentes aspectos, por exemplo, uma função pode possuir um coeficiente maior quando se trata de textos mais extensos, enquanto a outra é melhor em buscas por números, assim por diante (seção 3.3).

O resultado da consulta da aplicação retorna os valores com os mais altos graus de similaridade. O grau de similaridade varia entre 0 e 1, onde 1 é quando o valor é exatamente igual à palavra-chave escolhida. Foi definido um limiar do grau de similaridade de 0.6, onde qualquer resultado que estiver abaixo deste grau será descartado da consulta.

Figura 5. Tela principal do *SIMILAR*.



5. Implementação

A implementação do projeto utilizou a IDE Eclipse e todo o código foi desenvolvido na linguagem de programação JAVA 1.6 com várias bibliotecas externas para auxiliar a manipulação dos dados. As bibliotecas SAX e JDOM, foram usadas para extração e carga dos dados do XML para aplicação, e a biblioteca SIMMETRICS foi utilizada várias funções para as buscas por similaridade.

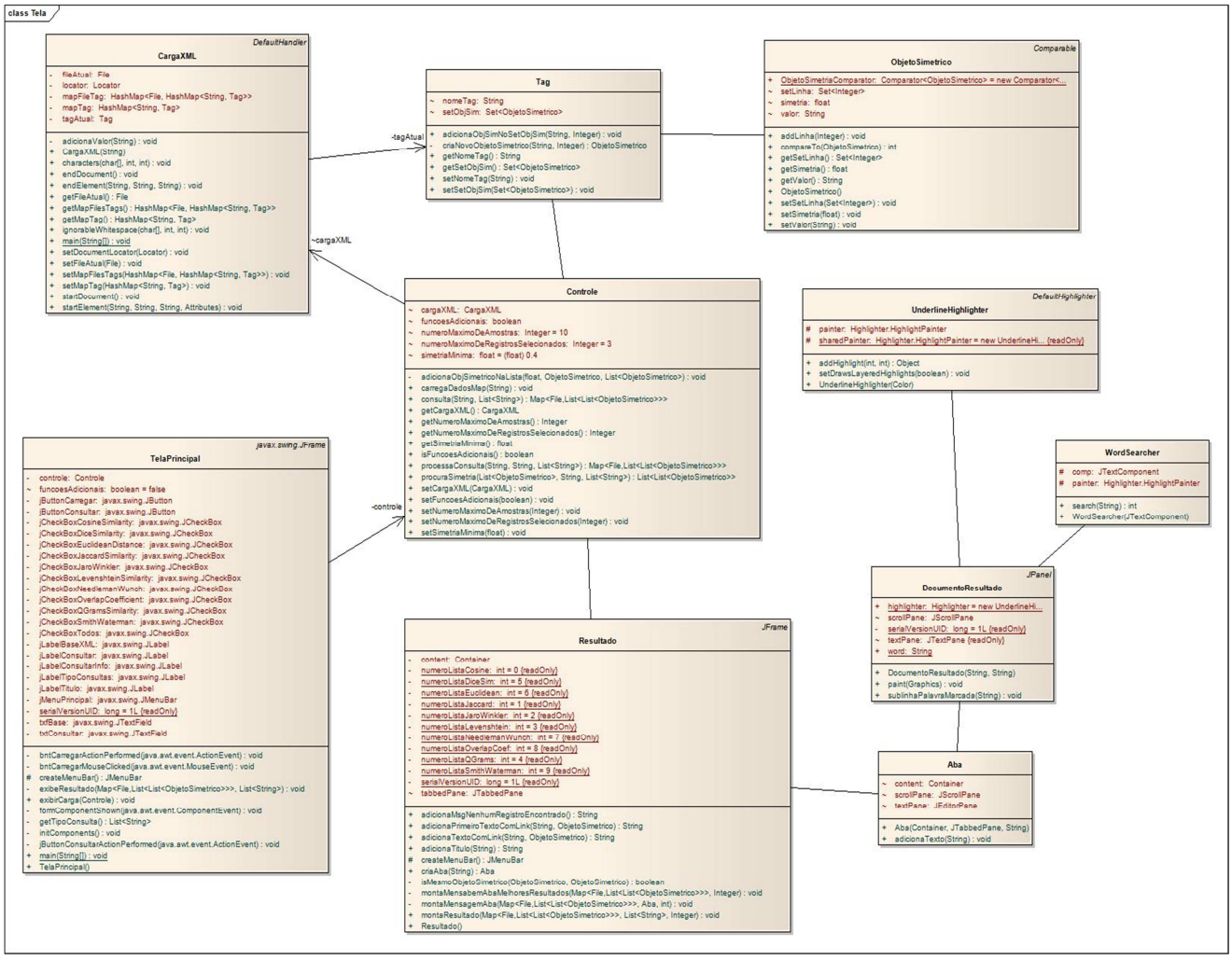
5.1 Projeto

A aplicação *Similar* foi desenvolvida em três etapas. A primeira etapa foi o desenvolvimento da carga dos dados, que envolve a leitura dos arquivos XML e o *parsing* dos dados para a estrutura Dewey. A segunda etapa foi implementar a busca nos dados em memória, incluindo todas as funções de similaridade que o aplicativo disponibiliza; fazer o *ranking* dos registros encontrados na busca, e também configurações de personalização da aplicação, como exemplo, informar o Threshold utilizado na busca. A terceira e última etapa foi o desenvolvimento da visualização do resultado, a divisão das funções de similaridade por abas, a aba de seleção dos melhores resultados dentre as funções selecionadas e a visualização do resultado no arquivo XML.

Todas as etapas foram desenvolvidos em Java SE através da IDE Eclipse.

5.2 Diagrama de Classes

Figura 6. Diagrama de Classes.



O diagrama de classe da aplicação *Similar* (Figura 6) mostra as suas principais classes e suas relações. A *TelaPrincipal* consiste em definir as características visuais da tela principal da ferramenta. O *Controle* é a classe que gerência as principais funções que a aplicação disponibiliza. A *CargaXML* é responsável pela leitura e *parsing* do documento XML e a classe *Tag* e *ObjetoSimetrico* são responsáveis por armazenar em memória os dados tratados. A classe *Resultado* tem como responsabilidade manipular as funções que dizem respeito à tela de resultados, onde as classes *Aba* e *DocumentoResultado* são classes auxiliares que ajudam a obter todas as funções que dizem respeito ao resultado final.

5.3 Carga dos Dados

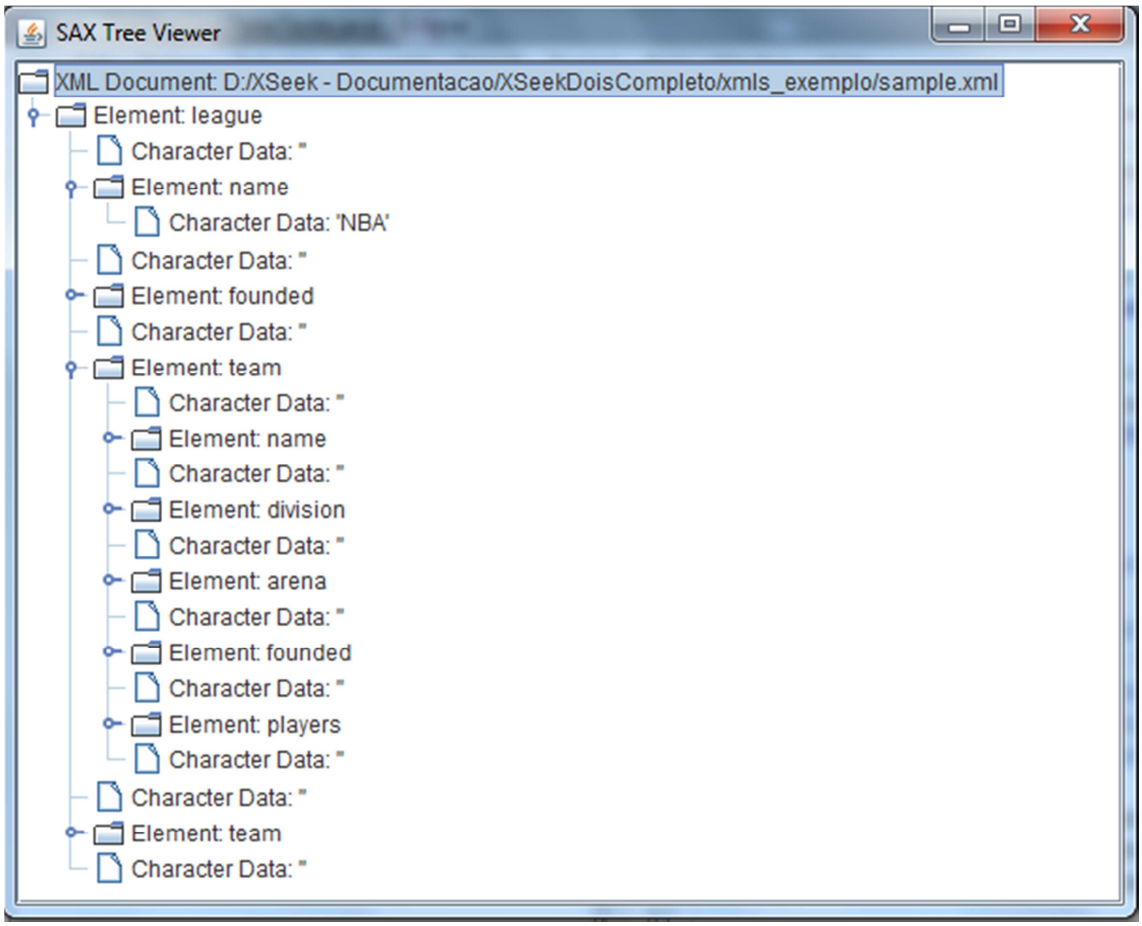
A carga de dados é feita usando uma biblioteca do Java chamada SAXParserFactory. Ela permite uma maior facilidade em manipular os elementos e tags do XML. Ela faz o trabalho do *parsing* do arquivo, fazendo com que seja possível criar uma estrutura de árvore DOM, onde é possível uma melhor busca dos dados na parte de pré-processamento.

A carga dos arquivos XML é processada após informar o caminho onde se encontra o arquivo ou uma pasta com vários arquivos (Figura 7).

Figura 7. Tela carga dos dados.

Após a leitura dos dados no documento em XML, os mesmos ficaram armazenados em memória em uma estrutura de dados Dewey (Figura 8).

Figura 8. Árvore Dewey.



5.4 Interface de consulta

A interface da aplicação permite ao usuário fazer a busca por conteúdo de três maneiras. Uma é definindo somente a palavra-chave (Figura 9), onde a aplicação percorre todos os nós da árvore para tentar achar algum valor similar ao da palavra-chave escolhida.

Figura 9. Tela de consulta por palavra-chave.

As outras formas de efetuar a busca são quando o usuário já conhece a estrutura do XML carregado em memória e deseja efetuar uma busca sobre a estrutura do

arquivo. A segunda forma consiste em especificar a *tag* onde o valor a ser pesquisado se encontra (Figura 10). A aplicação percorre todas as *tags* com o mesmo nome especificado na consulta e assim compara o valor com a palavra-chave.

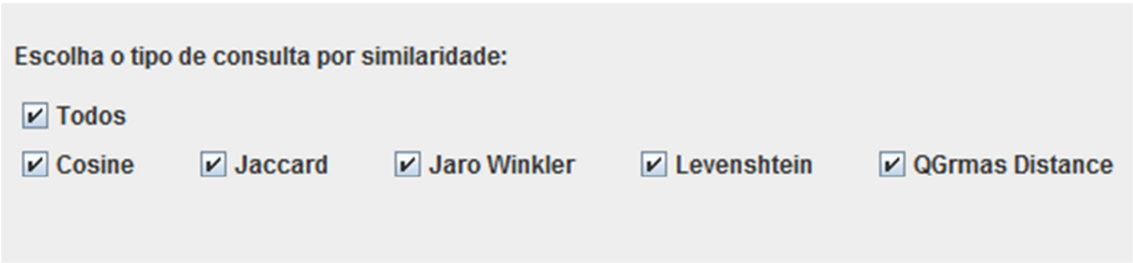
Figura 10. Tela de consulta por tag e palavra-chave.

E terceira forma possível de busca é a multi-tags(labels), onde acontece a maior restrição de dados, pois são pesquisados apenas os valores que estão dentro de todas as *tags* informadas (Figura 11). Por exemplo, dentro da tag "aluno" existe a *tag* "nome" e dentro dela são pesquisados todos os valores. Mesmo se a estrutura "aluno,nome" exista em diferentes níveis no documento, a pesquisa será efetuada em todos os níveis encontrados.

Figura 11. Tela de consulta por multi-tags e palavra-chave.

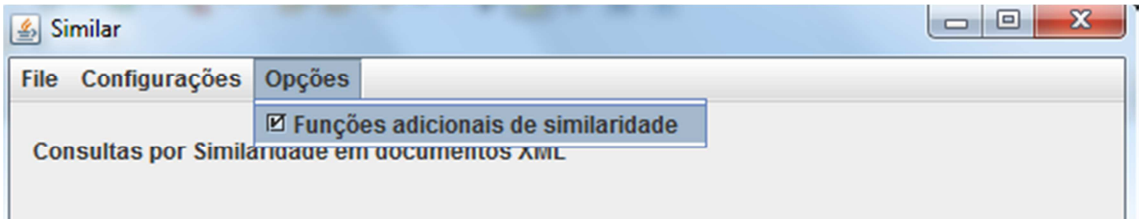
Após definir a palavra-chave da busca, é necessário escolher um ou mais mecanismos de busca por similaridade entre os disponíveis na aplicação, que são: Jaro Winkler, Jaccard, Cosine, Levenshtein, QGrams Distance (Figura 12).

Figura 12. Funções de similaridades básicas.



Existe uma opção avançada no aplicativo para que abra novas opções de funções de similaridade para o usuário utilizar. No menu opções habilite a opção de “Funções adicionais de similaridade” (Figura 13).

Figura 13. Menu de funções de similaridades avançadas.



Após habilitada a opção, as novas funções são visualizadas: Dice, Euclidean Distance, Needleman Wunch, Overlap Coefficient, Smith Waterman (Figura 14).

Figura 14. Funções de similaridades avançadas.

Depois de efetuada a consulta dos dados, a aplicação retornar os resultados obtidos separando por função de similaridade escolhida, onde cada função mostra os resultados de cada arquivo XML carregado ordenados de acordo com o grau de similaridade. Os resultados são melhor explicados na seção seguinte.

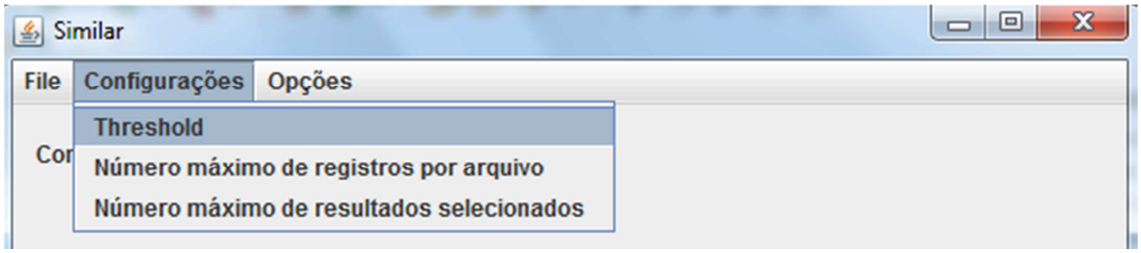
5.5 Apresentação dos Resultados

A aplicação *Similar* permite fazer algumas configurações que personalizam o aplicativo de acordo com a necessidade do usuário, dando uma maior flexibilidade de uso. Uma das configurações é a opção de “*Threshold*” que consiste em definir um ponto de corte para as funções de similaridade utilizadas. As funções de similaridade retornam

um valor de similaridade que varia de 0 a 1. Logo, os resultados que retornarem maior ou igual ao valor de *Threshold*, são exibidos ao usuário.

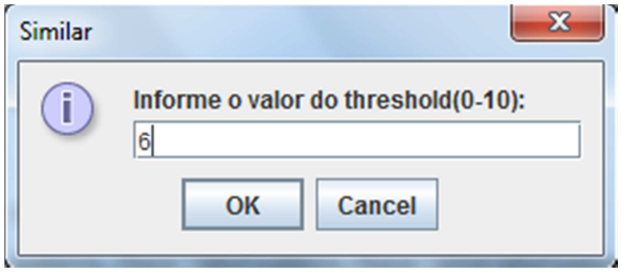
Para poder configurar o valor de *Threshold*, basta ir ao menu “Configurações” e clicar na opção “*Threshold*” (Figura 15).

Figura 15. Menu de Threshold.



Após clicar na opção “*Threshold*”, o sistema abre uma janela onde o usuário informar um valor de 0 a 10 (Figura 16), onde o 0(zero) seria setar nenhum ponto de corte, ou seja, o sistema iria retornar todos os resultados encontrados para o usuário e 10 (equivalente a 1) o sistema retornaria somente à *string* idêntica a palavra-chave digitada. Assim, se o usuário digitar o número 5, por exemplo, quer dizer que o sistema setaria um ponto de corte equivalente a 0.5 já que trabalhamos somente com a variação de 0 a 1.

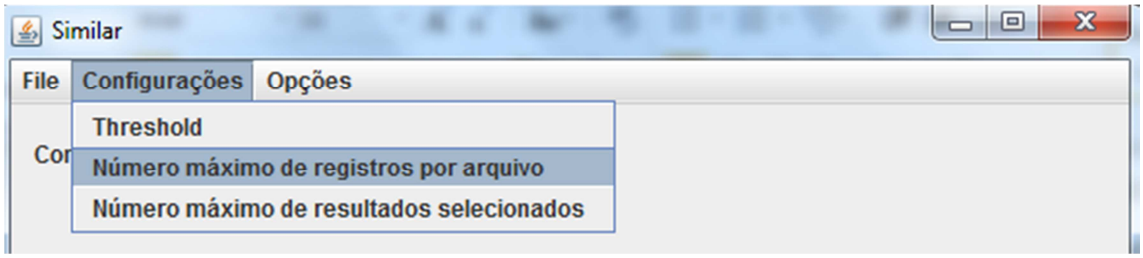
Figura 16. Tela de Threshold



Outra configuração do sistema é a escolha do número máximo de registros a serem exibidos. Quando os resultados forem visualizados, o número de registros mostrados serão os *x* (número escolhido pelo usuário) registros com os maiores índices de similaridade encontrados.

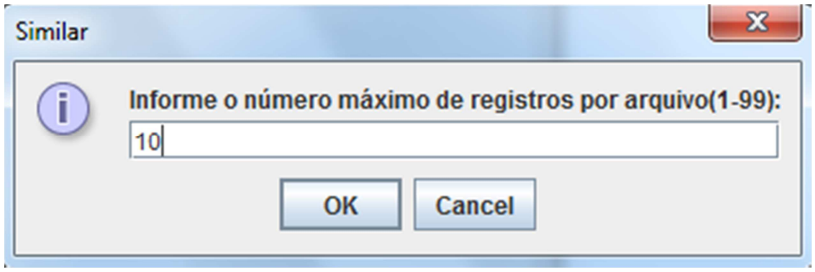
Para configurar o número máximo de registros por arquivo, basta ir ao menu “Configurações” e clicar na opção “*Número máximo de registros por arquivo*” (Figura 17).

Figura 17. Menu de número máximo de registros por arquivo.



Após clicar na opção “*Número máximo de registros por arquivo*” o sistema abre uma janela onde o usuário informa um valor de 1 a 99 (Figura 18), onde o 1 irá considerar somente o registro de maior valor de similaridade, e ao setar um valor muito alto, dependendo do número de arquivos xml carregados, o resultado pode ficar muito extenso.

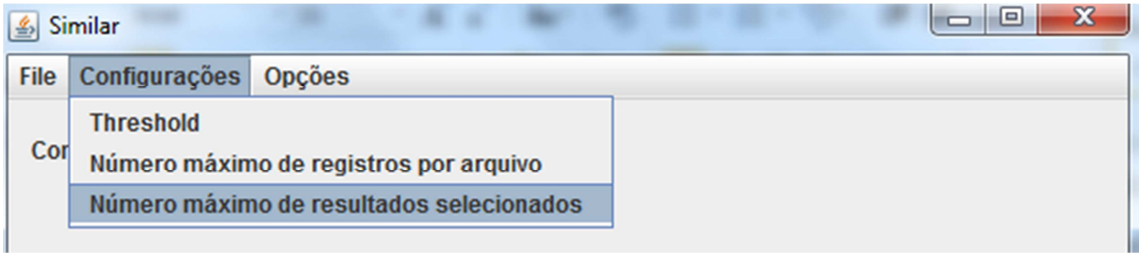
Figura 18. Tela de número máximo de registros por arquivo.



A última opção de configuração da aplicação *Similar* é parecida com a opção anterior, porém ela refere-se somente a aba “*Melhores Resultados*”. Esta aba consiste em fazer uma seleção dos melhores resultados obtidos (maior valor de similaridade) entre todas as funções de similaridade utilizadas sobre todos os arquivos xml. A opção “*Número máximo de resultados selecionados*” possibilita escolher o número máximo de registros selecionados entre cada arquivo.

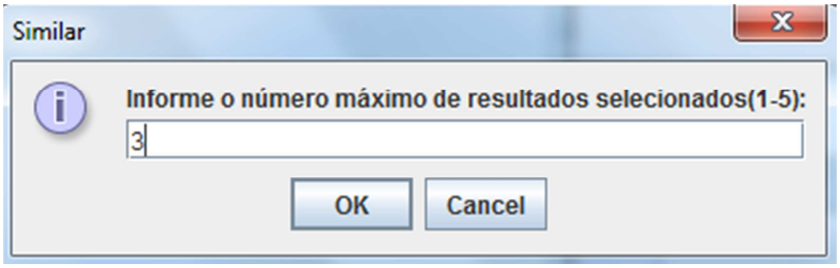
Para poder configurar o número máximo de registros por arquivo basta ir ao menu “Configurações” e clicar na opção “*Número máximo de resultados selecionados*” (Figura 19).

Figura 19. Menu de número máximo de resultados selecionados.



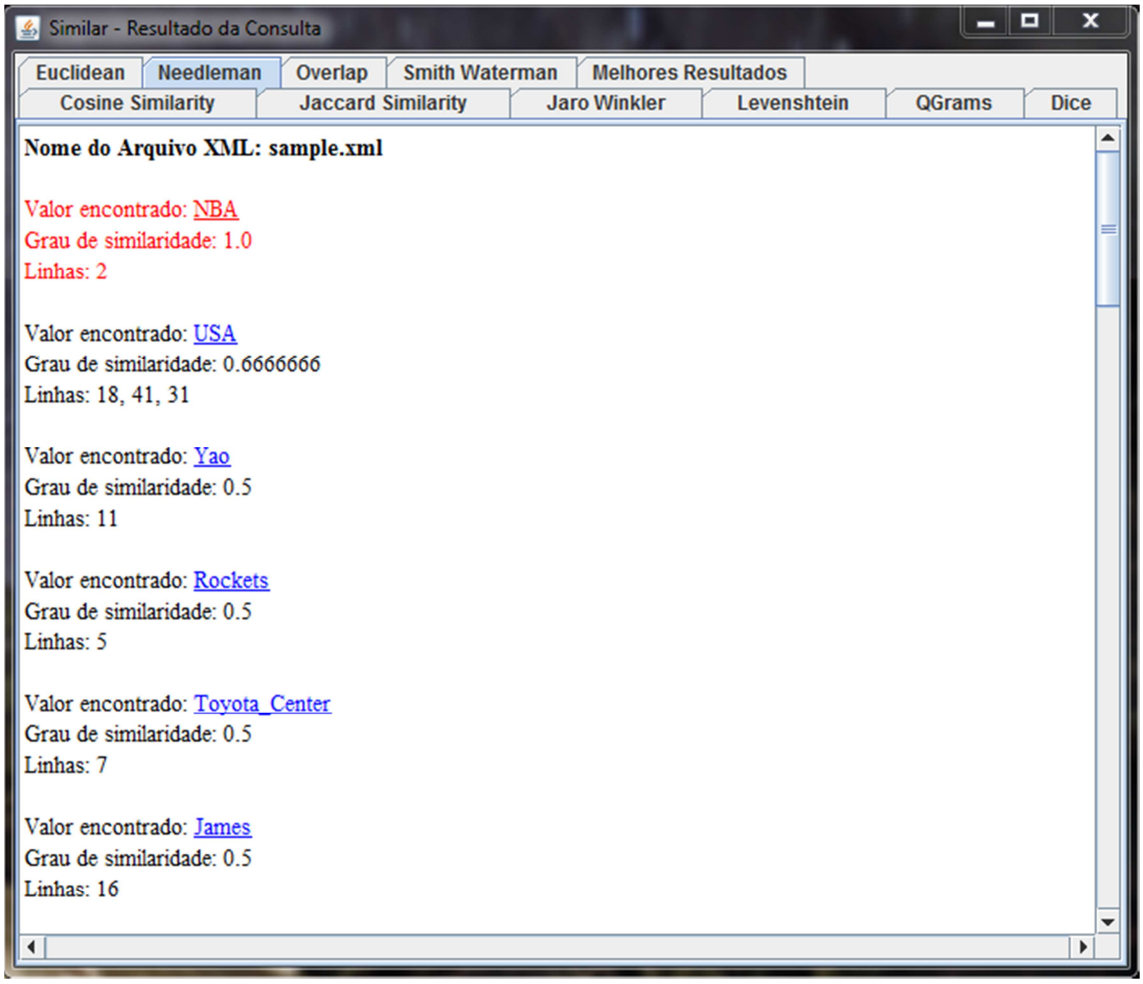
Após clicar na opção "*Número máximo de resultados selecionados*" o sistema abre uma janela onde o usuário pode informar um valor de 1 a 5 (Figura 20). É normal que a aplicação retorne valores de similaridade iguais para diferentes *strings* encontradas no mesmo arquivo, pois são utilizados diversos métodos de similaridade. Então o sistema seleciona por ordem alfabética estes registros. Este procedimento só acontece nesta aba "*Melhores Resultados*".

Figura 20. Tela de número máximo de resultados selecionados**.**



A interface principal do resultado da consulta é dividida em abas onde cada aba é referente a um método de similaridade escolhido pelo usuário mais a aba de "*Melhores Resultados*" (Figura 21). Em cada aba é exibido o nome do arquivo xml em negrito e logo abaixo é listado os resultados encontrados em ordem decrescente ao índice de similaridade. O primeiro registro encontrado é exibido na cor vermelha, para identificar o resultado de maior expressão entre os encontrados. Para cada registro encontrado é exibido o "*Valor encontrado*", que consiste na *string* localizada no documento, o "*Grau de similaridade*", que é o valor retornado das funções de similaridade, e onde este valor não pode ser abaixo do *Threshold* definido e as "*Linhas*" onde esta mesma *string* é encontrada neste documento.

Figura 21. Tela de resultados da consulta.



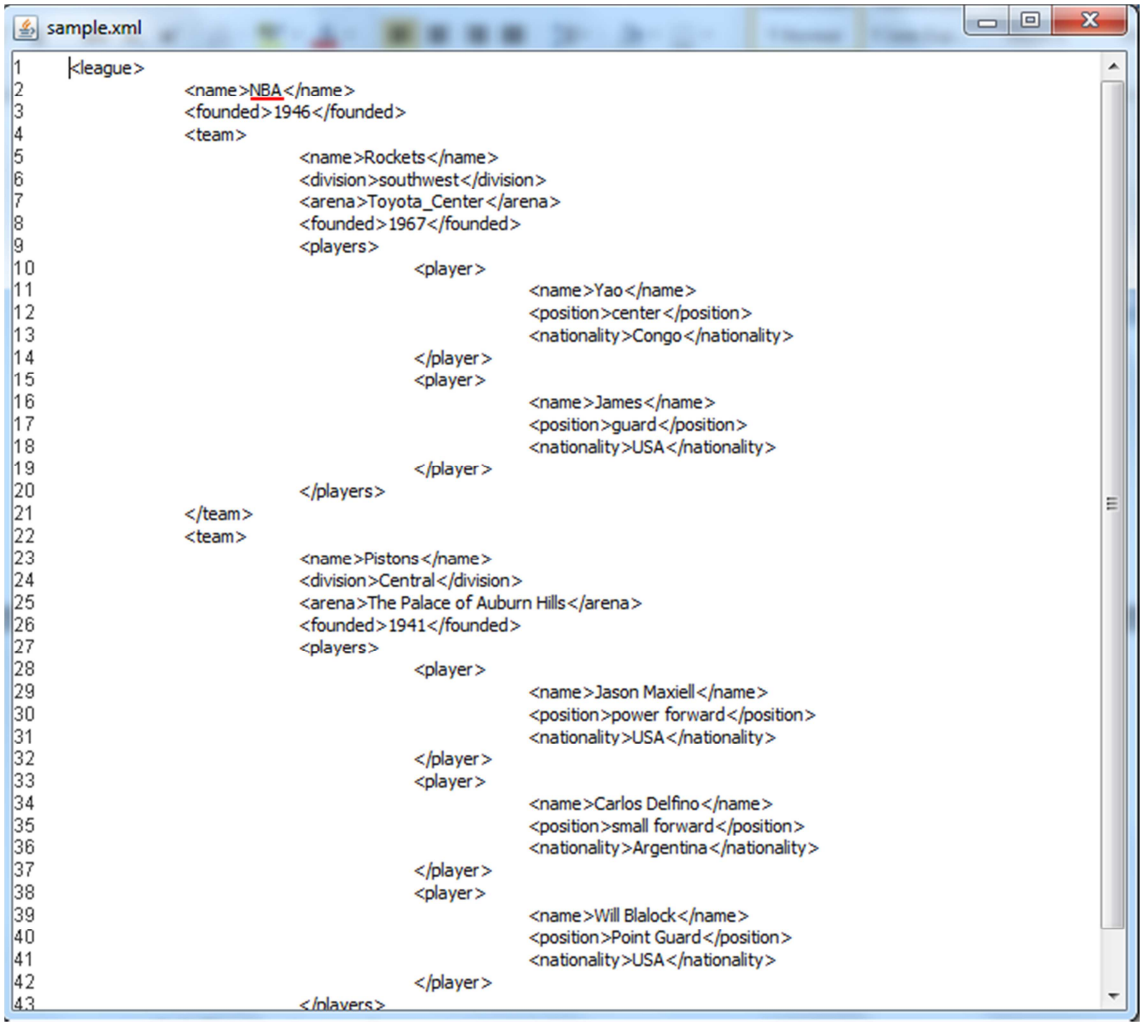
Os valores encontrados em cada documento referente à *string* localizada são também um link para a visualização do valor no documento, onde cada linha em que a *string* localizada existir será marcada, para uma melhor identificação da mesma.

5.6 Visualização do resultado no documento XML

Uma das partes do resultado da consulta é a visualização do resultado encontrado no documento xml onde foi feita a busca. Na tela principal do resultado da busca onde são exibidas todas as abas das funções de similaridade, existem links no valor dos "*Valores encontrados*" que abrem uma janela de visualização da *string* encontrada no documento. O título desta janela é o nome do documento aberto, onde é mostrada a primeira ocorrência do valor encontrado. Todas as linhas que esta mesma *string* existir no mesmo documento também são destacadas (sublinhada com a cor

vermelha). Esta funcionalidade facilita para o usuário identificar em qual nível da árvore o valor encontrado se localiza, não tendo a necessidade de procurar nos documentos carregados manualmente.

Figura 22. Tela de visualização do documento.



5.7 Teste de desempenho

Para executar os testes de desempenho com a aplicação *Similar* foram carregados 240 documentos XML diferentes, que ao todo consiste em 103mb de dados e demorou 03:16 min para efetuar toda a carga. Após a carga, a aplicação *Similar* Java consumia do sistema operacional 550mb de memória ram.

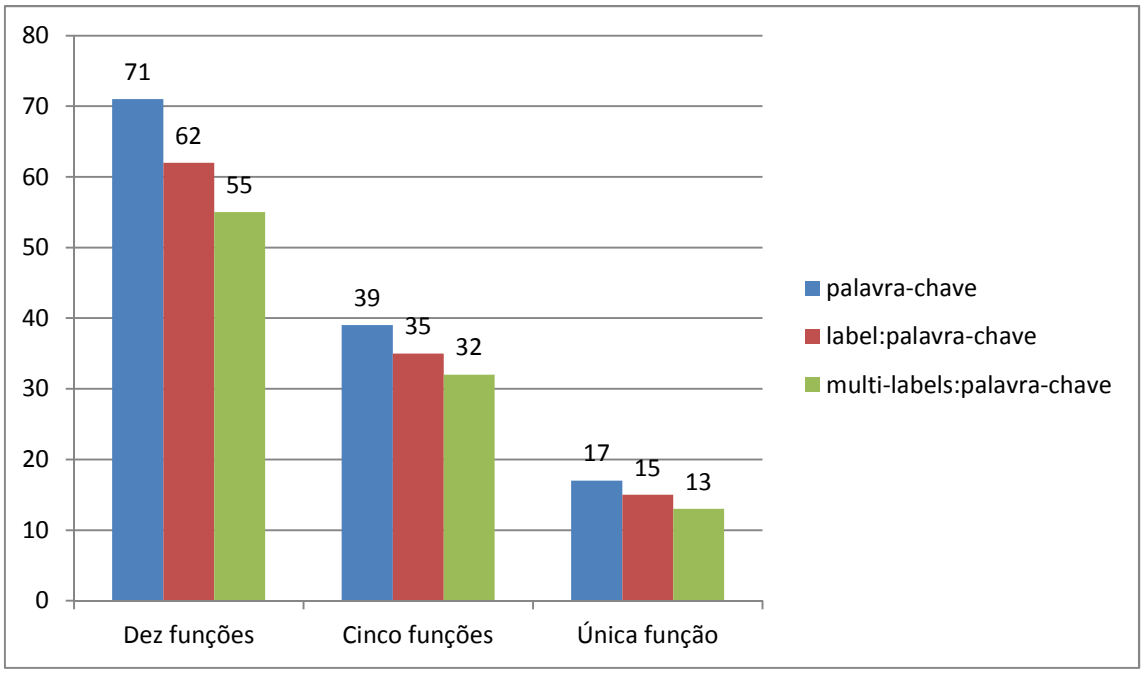
Para efetuar a consulta foram feitos três tipos de testes diferentes em relação à forma de consulta, somente por palavra-chave, *label* e palavra-chave e múltiplos *labels* e palavra-chave. Em cada uma das formas foram efetuados testes com todos os dez métodos de similaridade, com cinco métodos (*Jaro Winkler, Jaccard, Cosine, Levenshtein, QGrams Distance*) e somente com uma função de similaridade, onde para efeitos de teste foi escolhido o método *Cosine*.

Para o primeiro teste foi executado a consulta somente por palavra-chave fazendo com que ocorra o maior processamento possível, pois não há nenhuma restrição por labels e com todos os dez métodos diferentes de busca por similaridade a pesquisa foi efetuada em 71 segundos. Utilizando cinco métodos a pesquisa foi executada em 62 segundos e somente um único método de similaridade, a pesquisa foi feita no tempo de 55 segundos.

O segundo teste foi executado a consulta por um único *label* e uma palavra-chave (label:palavra-chave), fazendo uma pequena restrição dos dados carregados. O teste com todos os dez métodos diferentes de busca por similaridade a pesquisa foi efetuada em 39 segundos. Utilizando cinco métodos a pesquisa foi executada em 35 segundos e somente um único método de similaridade, a pesquisa foi feita no tempo de 32 segundos.

O terceiro e último teste foi executado a consulta com múltiplos *labels*, onde para efeitos de teste foram usados três *labels* e uma palavra-chave (label,label,label:palavra-chave), onde com todos os dez métodos diferentes de busca por similaridade a pesquisa foi efetuada em 17 segundos. Utilizando cinco métodos a pesquisa foi executada em 15 segundos e somente um único método de similaridade, a pesquisa foi feita no tempo de 13 segundos.

Gráfico 1: Testes de desempenho



Dados do computador utilizado:

Processador: Intel Core 2 duo (2Ghz)

Memoria Ram: 4Gb

Sistema Operacional: Windows 7

Máquina virtual Java (JVM), versão 1.6.0

5.8 Experimentos

Para determinar quais foram as “melhores” funções de similaridade disponibilizada na aplicação foram executados alguns testes para avaliar a sua precisão. Arquivos XML com diversos tipos diferentes de dados (caracteres, cadeia de caracteres, números, datas, etc.) foram escolhidos para servir de base de dados para as buscas. As buscas foram efetuadas de diversas formas, pesquisando por somente uma *string*, um conjunto de *strings*, números, datas, números e *strings*, *strings* incompletas (Tabela 2).

Tabela 2: Tabela de consultas.

Consulta	**Melhor função** (*k*=10,15,20; soma dos *k* grau de similaridade / *k* = média de similaridade)
“JAeMs”	Needleman
“02:10MP”	Jaro winkler
“AirbSU A319”	Jaro winkler
“149 or 150”	Smith Waterman
“Haryr K. T. Wong”	Jaro winkler
“01 - 07 - 2006”	Smith Waterman
“CasLTe Donington, EnglNAd”	Euclidean Distance
“Problems of Optimistic Concurrency ConRTol in Distributed Database SSYt”	Euclidean Distance
“Jon D. Clark, 27”	Jaro winkler

Com base nestas buscas, foi ranquiado os dez, quinze e vinte primeiros resultados, isto é, os resultados que obtiveram números maiores no grau de similaridade. Destas *k* amostras (10,15 e 20) foram feitas as médias dos valores encontrados para cada consulta (soma dos *k* grau de similaridade / *k* = média de similaridade), com a média de cada consulta foi feito a média das consultas, e assim as melhores funções de similaridade foram escolhidas.

Para *k*=10 amostras, as três melhores funções de similaridade foram:

Jaro Winkler: 0,71576

Euclidean Distance: 0,69037

Needleman Wunch : 0,64951

Para *k*=15 amostras, as três melhores funções de similaridade foram:

Jaro Winkler: 0,70187

Euclidean Distance : 0,679527

Needleman Wunch: 0,61084

Para *k*=20 amostras, as três melhores funções de similaridade foram :

Jaro Winkler: 0,68615

Euclidean Distance: 0,65933

Needleman Wunch: 0,61628

O resultado encontrado nos testes para as funções disponibilizadas na aplicação *Similar* demonstrou que as funções que tiveram os melhores desempenhos em retornar valores similares foram em primeiro lugar a Jaro Winkler, em segundo a Euclidean Distance, e em terceiro a Needleman Wunch. Este resultado não avalia se os valores encontrados na busca são relevantes à consulta, analisam somente o grau de similaridade da função utilizada, ou seja, mesmo que a função de similaridade tenha um alto grau de similaridade não quer dizer que a *string* encontrada seja de interesse ao escopo busca.

Este resultado não significa que estas são as melhores funções de similaridade para todos os escopos (strings, datas, números, etc.), significa é que de um modo geral ou na média dos resultados, estas funções se destacaram com diferentes escopos (tipos de consulta).

6. Conclusões e Trabalhos futuros

Os mecanismos tradicionais de busca não consideram a estrutura do documento XML. Nestes documentos buscas são feitas por meio de consultas em XQuery [W3C s.d.], ou buscas por coincidência exata (palavra-chave) que não suportam consultas por similaridade. Na XQuery, há trabalhos que introduzem funções de similaridade no processador da linguagem (XSimilarity [Silva, Borges e Galante 2008]); em busca por palavra-chave não há propostas de consulta por similaridade a XML, considerando sua estrutura.

A proposta do trabalho é desenvolver um sistema de busca por palavra-chave, por similaridade, em XML, que auxilie o usuário a encontrar respostas que representem o mesmo objeto do mundo real daquele usado por ele na consulta. A aplicação *Similar* desenvolvida é uma ferramenta de busca capaz de efetuar consultas por similaridade utilizando a estrutura do arquivo. As formas de pesquisas disponibilizadas na ferramenta é a busca somente a palavra-chave e a outra forma consiste em especificar a *tag* ou multi-*tags*(labels) do documento onde acontece a maior restrição de dados, pois são pesquisados apenas os valores que estão dentro de todas os *labels* informadas. O *Similar* permite efetuar buscas com até dez tipos diferentes de funções de similaridade paralelas, podendo assim traçar um comparativo entre os resultados obtidos em cada função. O resultado da consulta é a seleção dos melhores resultados em cada método de similaridade escolhido.

Como resultado final, alguns experimentos foram efetuados a fim de verificar a confiabilidade do resultado obtido através da proposta implementada do *Similar*. Os testes executados avaliaram o desempenho da ferramenta, onde foi carregada uma grande quantidade de arquivos XML e efetuados diversos tipos diferentes de combinações entre a forma de consulta e o número de métodos utilizados. Outro experimento avaliou as funções de similaridade que mais se destacaram em diferentes escopos, ou seja, as que tiveram maior grau de similaridade nos registros encontrados, definido assim as três melhores funções (Euclidean Distance, Jaro Winkler e Needleman Wunch). Algumas limitações foram identificadas a partir deste resultado e incluídas aos trabalhos futuros.

Os principais trabalhos futuros identificados neste projeto são listados conforme segue:

1. Fazer a análise das funções de similaridade para identificar quais que são mais eficientes para determinado tipo de palavra-chave, sendo ela uma string, data, números, etc. Sendo assim é possível implementar uma aplicação mais "inteligente", onde é possível identificar o tipo de palavra-chave utilizada pelo usuário e assim restringir o número de funções para aquelas mais especializadas.

2. Estender a aplicação para que ela suporte XQuery.

3. Adicionar um banco de dados na aplicação para que os dados carregados dos documentos XML fiquem gravados em um banco de dados XML.

4. Fazer um estudo para que a aplicação Similar fique mais performática para grande quantidade de dados.

5. Estender a aplicação Similar para que ela suporte uma busca por dados temporais, isto é, dentro dos documentos fornecidos pelo usuário para a busca, quais deles se referem a uma data determinada, ou ao um período de tempo.

6. Desenvolver novas funções de similaridade e acoplar a aplicação Similar, ou ainda, fazer um estudo para melhorar alguma das funções já existentes para aprimorar sua similaridade.

7. Implementar uma busca por fonemas na aplicação Similar.

7. Referências

Amer-Yahia, S.;Lalmas, M. XML Search: Languages, INEX and Scoring. University of London.2006.

BAUM, P. Date Algorithms. 1998. Disponível em: <http://vsg.cape.com/pbaum/date/date0.htm>. Acesso em: 30 jun. 2005.

C. F. Dorneles, R. M. Galante: Aplicação de Funções de Similaridade e Detecção de Diferenças em Grandes Volumes de Dados Distribuídos. In: Tomasz Kowaltowski; Karin Breitman. (Org.). Jornadas de Atualização em Informática - JAI. 1 ed. Rio de Janeiro: Editora PUC-Rio, 2008, v. 1, p. 233-272.

COHEN, S. et al. XSEarch: a semantic search engine for xml. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 2003, Berlin, Germany. Proceedings... [S.1.]: Morgan Kaufmann, 2003. p.45-56.

COHEN, W.; RAVIKUMAR, P.; FIENBERG, S. A Comparison of String Metrics for Matching Names and Records. In: KDD-2003 WORKSHOP ON DATA CLEANING AND OBJECT CONSOLIDATION, 2003, Washington, DC. Proceedings...[S.1.: s.n.], 2003. p.13-18.

GUTH, G. Surname Spellings and Computerized Record Linkage. Historical Methods Newsletter, [S.1.], v.10, n.1, p.10-19,1976.

JACCARD, P. The distribution of flora in the alpine zone. The New Phytologist, [S.1], v.11, n.2, p.37-50, 1912.

JARO, M. Probabilistic Linkage of Large Public Health Data Files. Statistics in Medicine, [S.1], v14, n.(5-7), p.491-498,1995.

Jiaheng Lu, Tok Wang Ling, Chee-Yong Chan, Ting Chen. From Region Encoding To Extended Dewey: On Efficient Processing of XML Twig Pattern Matching. Proceedings

of the international conference on Very large data bases, 2005. < http://www.vldb2005.org/program/paper/tue/p193-lu.pdf >Acesso em 11/12/2011

LAIT, A.; RANDELL, B. An Assessment of Name Matching Algorithms. [S.1.]: Department of Computing Science of University of Newcastle upon Tyne, 1993. Technical report.

LEVENSHTEIN, V. Binary codes capable of correcting spurious insertions and deletions of ones. Problems of Information Transmission, [S.1.], v.1, n.1, p.8-17, 1965.

Lima A. E. N.; Dorneles C. F.; Heuser C. A. Eris: Um protótipo para consulta por similaridade a bases de dados XML. Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS).2006.

Silva M. E. V.; Borges E. N.; Galante R. M. XSimilarity : Uma Ferramenta para Consultas por Similaridade embutidas na Linguagem XQuery. Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS).2008.

SimMetrics – Similarity Metric Library. Disponível em < http://www.aktors.org/technologies/simmetrics/index.html > Acesso em 28/02/2012.

Suporte a argumentos de consulta vagos através da linguagem XPath / Alvaristo Bernardes do Amaral Padilha. – Porto Alegre: PPGC da UFRGS, 2005.

Tatarinov, I.; Viglas, S.; Beyer, K. S.; Shanmugasundaram, J.; Shekita, E. J.; Zhang, C. Storing and querying ordered XML using a relational database system. In: Sigmod Conference, 2002. ACM. p.204–215.

WINKLER, W. E. The state of record linkage and current research problems. Wachington, DC: Statistical Research Division, U.S. Bureau of the Census, 1999.(R99-04).

XML Data Repository. University of Washingthon, Seattle – USA. < http://www.cs.washington.edu/research/xmldatasets/www/repository.html > Acesso em 01/12/2010.

XQuery: An XML Query Language. Disponível em < http://www.w3.org/TR/xquery/ > Acesso em 28/11/2010.

XSeek - Intelligent Structured Search Engine. : Disponível em < http://xseek.asu.edu/ > Acesso em 25/11/2010.

Ziyang Liu, Jeffrey Walker, Yi Chen. XSeek: a semantic XML search engine using keywords. Proceedings of the 33rd international conference on Very large data bases, 2003.