UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MARIA ESTELA VIEIRA DA SILVA

XSimilarity : Uma ferramenta para consultas por similaridade embutidas na linguagem XQuery

Trabalho de Graduação.

Profa. Dra. Renata de Matos Galante
Orientador

Eng. Eduardo Nunes Borges
Co-orientador

Porto Alegre, novembro de 2007.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, por tudo de bom que já me aconteceu até hoje e por todas as pessoas boas que cruzaram o meu caminho.

Agradeço a meus pais pela vida em si. Agradeço aos meus avós que ocuparam o papel de pais em minha vida e sem dúvida são os grandes responsáveis pelas minhas melhores qualidades, incluindo a vontade de concluir a graduação. E também aos meus irmãos por serem bons irmãos, além de amigos com os quais eu sempre poderei contar.

Agradeço ao meu grande amor e namorado pelo companheirismo e incentivo para que eu chegasse até aqui, tão grandes que eu somente poderia encontrar em um grande amor. Agradeço por todos os momentos em que você me deixou tranqüila com a sua presença e o seu amor para que eu fizesse o melhor trabalho possível e não desistisse de nada.

Agradeço aos meus orientadores, Renata de Matos Galante e Eduardo Borges, por tudo em relação a esse trabalho de conclusão. Por terem acreditado na conclusão desse trabalho até o fim e me apoiado nisso, mesmo em horários que extrapolavam suas rotinas de trabalho ou estudo. E pelas idéias boas que deram no desenvolvimento desse trabalho e que me ajudaram a adquirir novos conhecimentos.

Agradeço aos amigos que fiz, aos colegas de faculdade e aos professores desta universidade que transmitiram seus conhecimentos e que tornaram esse trabalho possível. Agradeço a todos que contribuíram, de forma direta ou indireta, para que eu transpusesse essa barreira e construísse as bases para a nova etapa que se inicia com o término da graduação.

A todos vocês, os meus mais sinceros agradecimentos!

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

DTD	*Document Type Definition*
FLWOR	*For, Let, Where, Order By* e *Return*
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
UFRGS	Universidade Federal do Rio Grande do Sul
UML	*Unified Modeling Language*
W3C	*World Wide Web Consortium*
XML	*Extensible Markup Language*
XPath	*XML Path Language*
XQuery	*XML Query*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

RESUMO

Esse trabalho apresenta uma ferramenta para a realização de consultas por similaridade embutidas na linguagem XQuery. São implementadas funções de similaridade relevantes na área de banco de dados e recuperação de informações. É feita uma análise comparativa dessas funções e a validação da ferramenta através de experimentos com dados de artigos científicos da área da Ciência da Computação oriundos da biblioteca digital DBLP.

A principal contribuição desse trabalho é gerar uma ferramenta que permite consultas por similaridade em bases de dados no formato XML, possibilitanto a otimização da recuperação de informações nessas bases de dados. Além disso, como foram implementadas várias funções de similaridade diferentes, a ferramenta também permite comparar a eficácia dessas funções para diferentes contextos de utilização.

Palavras-Chave: XML, XQuery, Similaridade, Ferramenta, Consulta.

XSimilarity: A tool for similarity queries embedded in XQuery language

ABSTRACT

This work presents a tool for executing similarity queries embedded in XQuery language. Relevant similarity functions for database and information recovery fields are implemented in the tool. A comparative analysis of these functions and the tool validation by means of practical examples are also made. The examples are constructed over Computer Science scientific articles data provided by the DBLP digital library.

The main contribution of this work is generating a tool for similarity queries in XML databases. This allows for information recovery optimization in these databases. Besides that, as many different similarity functions were implemented, the tool allows comparing the effectiveness of these funcions in different contexts.

Keywords: XML, XQuery, Similarity, Tool, Query.

1 INTRODUÇÃO

Os mecanismos tradicionais de pesquisa em banco de dados (GROFF, 2002) utilizam uma busca por coincidência exata. Em uma base de dados de cidades, por exemplo, “Rio Grande” e “R. Grande” são considerados objetos diferentes. Mas ambos os objetos fazem referência a uma mesma cidade. A recuperação da informação por meio de consultas exatas pode ser ineficiente 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 (Jagadish, 1995).

A linguagem XML (W3C, 2006) vem ao longo do tempo provando ser a melhor forma de representação de conteúdo, pois oferece um método simples e ao mesmo tempo poderoso para a representação de documentos. Recorrendo à definição de um esquema, por exemplo, é possível efetuar a validação de documentos. Esta característica revela-se de extrema importância para aplicações em que a verificação estrutural de dados seja vital, como é o caso das bases de dados convencionais. Devido a suas inúmeras vantagens, a linguagem XML vem sendo aplicada, cada vez mais, nos mais diversos campos da informática.

O XQuery (W3C WORKING GROUP, 2005) é uma linguagem criada a partir da necessidade de se realizar consultas sobre arquivos XML. XQuery está para o XML assim como SQL está para os bancos de dados relacionais, pois embora suas sintaxes sejam diferentes, a semântica é a mesma. Cada expressão XQuery é chamada de uma expressão FLWOR. As expressões FLWOR possuem funcionalidade semelhante ao conjunto de instruções SELECT-FROM-WHERE do SQL, porém a sintaxe FLWOR permite a realização de pesquisas mais complexas e de leitura mais fácil.

Dado o exposto acima, torna-se evidente o interesse em desenvolver algum mecanismo de consulta por similaridade em bases XML. A linguagem X-Query, desenvolvida especificamente para trabalhar sobre XML, é uma excelente alternativa para a inclusão de tal mecanismo.

Este trabalho descreve o projeto e a implementação de uma ferramenta para executar consultas por similaridade em documentos XML. As funções de similaridade são embutidas na linguagem de consulta XQuery. A ferramenta é validada através de experimentos com dados de artigos científicos da área da Ciência da Computação oriundos da biblioteca digital DBLP (University of Trier, s.d.).

A principal contribuição desse trabalho é gerar uma ferramenta que permite consultas por similaridade em bases de dados no formato XML, possibilitanto a otimização da recuperação de informações nessas bases de dados. Além disso, como

foram implementadas várias funções de similaridade diferentes, a ferramenta também permite comparar a eficácia dessas funções para diferentes contextos de utilização.

Este trabalho está estruturado da seguinte forma:

- O capítulo 2, A linguagem de consulta XQuery, descreve a sintaxe e o funcionamento da linguagem XQuery e como ela pode ser estendida para incorporar funções de similaridade;
- O capítulo 3, Funções de Similaridade, apresenta os principais conceitos sobre consultas por similaridade, que serve de base para a compreensão da ferramenta proposta. Além disso, são descritas as principais funções de similaridades implementadas na ferramenta proposta neste trabalho;
- O capítulo 4, XSimilarity, discorre sobre a ferramenta implementada. Este capítulo apresenta funcionalidades, linguagem utilizada e bibliotecas de terceiros importadas. Também são apresentados *screenshots* da ferramenta como forma de exibir os resultados obtidos. Por fim, é realizado um estudo de caso que demonstra a validade da ferramenta através de experimentos com dados de artigos científicos da área da Ciência da Computação oriundos da biblioteca digital DBLP (University of Trier , s.d.);
- O capítulo 5, Conclusão, apresenta as conclusões obtidas com a realização do trabalho e propostas de trabalhos futuros.

2 A LINGUAGEM DE CONSULTA XQUERY

O presente capítulo apresenta as principais características da linguagem de consulta XQuery bem como detalhes de como a linguagem pode ser estendida para suportar o processamento de funções definidas pelo usuário. Primeiramente XQuery é conceituada e explicada com alguns exemplos práticos. Em seguida, é ilustrada a forma como um processador de consultas XQuery foi utilizado para embutir funções de similaridade na linguagem XQuery.

2.1 Principais Características da Linguagem XQuery

XQuery [referencia] é uma especificação do W3C (http://www.w3.org/TR/xquery) para uma linguagem de consulta sobre dados estruturados em XML. É declarativa, de maneira semalhante a SQL, e deriva do Quilt (CHAMBERLIN et al, 2000), uma linguagem que herda características de muitas outras, como XPath 1.0 (W3C, 1999), XQL (ROBIE et al., s.d.), XML-QL (DEUTSCH et al, s.d.), SQL (ISSO, 1999) e OQL (CATTELL et al, 1996).

Embora atualmente XML seja uma tecnologia padrão de fato para o intercâmbio de dados, a confiabilidade e a tradição dos sistemas de bancos de dados relacionais farão coexistir os padrões relacional e XML ainda por muito tempo. Dessa forma as aplicações modernas precisam estar preparadas para lidar com ambos os padrões. XQuery surgiu para prover uma interface que permite o acesso a dados em ambas as fontes sob uma visão unificada de modelo.

A sintaxe do XQuery baseia-se nas suas funções pré-definidas e em um conjunto de cinco operadores que lembram os do SQL. São eles: FOR, LET, WHERE, ORDER BY e RETURN. Por serem usados sempre juntos e nesta ordem (embora nem todos sejam obrigatórios), cada expressão XQuery é chamada de uma expressão FLWOR, uma abreviatura das instruções utilizadas. As expressões FLWOR possuem funcionalidade semelhante ao conjunto de instruções SELECT-FROM-WHERE do SQL. Entretanto, apresentam algumas construções a mais como o LET, por exemplo. Uma correspondência entre os operadores XQuery e os de SQL pode ser vista no Quadro 2.1.

For ⇔ SQL from
Where ⇔ SQL where
Return ⇔ SQL select
Order ByóSQL order by
Let (sem equivalência SQL) para variáveis temporárias, principalmente para execução de agregações

- FOR/LET associam valores às variáveis
- WHERE filtra o resultado vindo das cláusulas FOR/LET
- RETURN gera a saída da consulta

Quadro 2.1: Equivalências entre XQuery e SQL

XQuery é capaz de consultar documentos XML simples ou coleções de documentos XML. As consultas podem resultar na transformação dos resultados ou na criação de um novo documento XML.

A seguir serão apresentadas em mais detalhes as principais construções dessa linguagem. Os arquivos XML BDBComp (Laender , 2004) e DBLP (University of Trier, s.d.), apresentados no Quadro 2.2, serão utilizados para facilitar o entendimento. Tratam-se de dois arquivos contendo metadados sobre o mesmo artigo científico, mas com nomenclaturas diferentes.

BDBComp

01 <oaidc> → atributos omitidos
02<title>Detecção de Sitios Replicados Utilizando Conteúdo e Estrutura </title>
03 <creator>André Luiz da Costa Carvalho</creator>
04 <creator>Allan José de Souza Bezerra</creator>
05 <creator>Edleno Silva de Moura</creator>
06 <creator>Altigran Soares da Silva</creator>
07 <creator>Patrícia Silva Peres</creator>
08 <description>Identifying replicated sites is an important task for search engines.It can reduce data storage costs, improve query processing time and remove noises that might affect the quality of the final answer given to the user . This paper introduces a new approach to detect replicated sites in search engines databases, using as replication evidences the websites' structure and the content of their pages. It is also depicted the result of experiments performed with a real search engine database. Our approach found 8.43% of the web pages stored in the database were in replicated web sites with 94.4% precision, result witch is more accurate than the ones found in other works.</description>
09 <date>2005</date>
10 <type>Text</type>
11 <identifier>sbbd2005article002</identifier>
12<identifier>http://www.sbbd-sbes2005.ufu.br/arquivos/artigo-02-novo_Carvalho.pdf</identifier>
13 <source>sbbd2005</source>
14 <language>por</language>
15 <coverage>Uberlândia, MG, Brasil</coverage>
16 <rights>Sociedade Brasileira de Computação</rights>
17 </oaidc>

DBLP

18 <inproceedings> _ atributos omitidos

```
19 <author>Andr&amp;eacute; Luiz da Costa Carvalho</author>
20 <author>Allan Jos&amp;eacute; de Souza Bezerra</author>
21 <author>Edleno Silva de Moura</author>
22 <author>Altigran Soares da Silva</author>
23 <author>Patr&amp;iacute;cia Silva Peres</author>
24<title>Detec&amp;ccedil;&amp;atilde;o de R&amp;eacute;plicas Utilizando Conte&amp; uacute;do e Estrutura.</title>
25 <pages>25-39</pages>
26 <year>2005</year>
27 <crossref>conf/sbbd/2005</crossref>
28 <booktitle>SBBD</booktitle>
29 <ee>http://www.sbbd-sbes2005.ufu.br/arquivos/artigo-02-novo_Carvalho.pdf</ee>
30 <url>db/conf/sbbd/sbbd2005.html#CarvalhoBMSP05</url>
31 </inproceedings>
```

Quadro 2.2: Metadados heterogêneos associados ao mesmo artigo científico.

2.1.1 Estrutura da Linguagem XQuery

Uma consulta XQuery pode ser dividida em três partes: declarações de esquema, definição de funções e expressões de consulta. O único fragmento obrigatório é a expressão de consulta.

A seção de declarações de esquema serve para indicar quais os esquemas utilizados nas seções posteriores. Um esquema, representado por sua URL, define o formato para um arquivo XML. Isto é feito indicando como as tags devem ser combinadas de modo a representar determinado domínio de informação e que informações podem estar associadas a cada tag, bem como qual o tipo destas informações.

Como pode haver esquemas que utilizem nomes repetidos para *tags* e seus atributos, cada esquema é associado a um *namespace*, que permite identificar de forma única, referências a um mesmo nome. No exemplo apresentado no Quadro 2.3, o esquema de URL "http://www.w3.org/2000/10/XMLSchema" é associado ao *namespace* xsd. Posteriormente, xsd poderá ser utilizado para referenciar itens definidos dentro do esquema.

A seção de definição de funções serve para definir as funções que serão utilizadas nas expressões de consulta XQuery. No exemplo definido no Quadro 2.3, a função conta_autores retorna o número de autores de um metadado referente a um artigo científico. Essa função recebe como parâmetros duas strings. A primeira é o nome do arquivo XML que armazena os metadados do artigo e a segunda trata da denominação para o criador do artigo utilizada.

Na terceira parte constam as expressões de consulta da XQuery. Tais expressões podem ser compostas por funções externas definidas na seção de definição de funções e por expressões da linguagem propriamente dita, como as expressões FLWOR. O resultado de uma consulta XQuery pode ser um documento XML ou um fragmento de um documento como no exemplo definido no Quadro 2.3. Uma expressão de consulta deve ser circundada por chaves para que o valor retornado possa ser interpretado de maneira adequada. Normalmente uma expressão de consulta está inserida entre *tags* XML que descrevem sua funcionalidade.

```
# Parte 1: Namespace e Declarações de Schema
namespace xsd = “http://www.w3.org/2000/10/XMLSchema”
# Parte 2: Definição de funções
declare function conta_autores (xsd:string $n, xsd:string $o) returns xsd:integer {
        let $t := doc($n)//$o
         return
        {count($t)}
}
# Parte 3: Expressões de consulta
<Results>
        <Description>Número de Autores do Artigo Científico </Description>
        <Value>{conta_autores(“BDBComp.xml”,”creator”)}</Value>
</Results>
```

Quadro 2.3: Estrutura XQuery

2.1.2 Expressão FLWOR

A expressão FLWOR é a principal estrutura do XQuery. Um exemplo simples é apresentado no Quadro 2.4. Nesse exemplo, a variável $b é utilizada para varrer todos os nodos oaidc existentes no arquivo de metadados do artigo científico (o for liga variáveis nodo). Para cada elemento em $b, é atribuído a $c o conjunto de autores de $b (o let liga a coleção de elementos à variável $c). O where é utilizado para filtrar, dentre os nodos identificados por $b, apenas aqueles que tiverem mais de 5 autores, conforme identificado por $c (a função count é utilizada para contabilizar o número de elementos no conjunto). Nos casos em que a condição for verificada, é retornado o valor do campo título, a data e o número de autores do artigo representado por $b. Cada resultado deve ser circundado pela tag denominada “result”.A saída completa será ordenada de forma decrescente pelo número de autores dos artigos e circundada pela tag “resposta_final”.

```
<resposta_final>
FOR $b in doc(“BDBComp.xml”)/oaidc
LET  $c:=$b/creator
WHERE count($c) > 5
ORDER BY count($c) DESCENDING
RETURN
<result>
    <titulo>{$b/title/text()}</titulo>
    <data>{$b/date/text()}</data>
    <num_autores>{count($c)}</num_autores>
</result>
</resposta_final>
```

Quadro 2.4: Expressão FLWOR

2.1.3 Expressões Condicionais

São análogas ao IF-THEN-ELSE das linguagens de programação. Se o teste for verdadeiro, o valor da primeira expressão é retornado. Caso contrário, o resultado da segunda expressão é retornado. É um diferencial importante da XQuery. No Quadro 2.5 é mostrada uma expressão que retorna a fonte se a data do artigo for “2005” ou a descrição em caso contrário.

```
for $l in doc("BDBComp.xml")/oaidc
return
  <Resultado>
    { $l/title,
     if ($l/date = "2005")
     then $l/source
     else $l/description
    }
  </Resultado>
```

Quadro 2.5: Expressão Condicional

2.1.4 Expressões de Quantificação

Expressões de quantificação em XQuery tem a mesma semântica do cálculo relacional. No Quadro 2.6 é mostrado de uso quantificador existencial. Nesse exemplo, são retornadas todas as descrições de artigos científicos que possuem algum autor que contenha a palavra "Silva" em seu nome. No Quadro 2.7, são retornadas todas as descrições de artigos em que todos os autores tem a palavra "Silva" em seus nomes.

```
for $b in doc("BDBComp.xml")/oaidc
where some $p in $b/creator satisfies
   (contains($p,"Silva"))
return $b/description
```

Quadro 2.6: Expressão SOME

```
for $b in doc("BDBComp.xml")/oaidc
where every $p in $b/creator satisfies
  (contains($p,"Silva"))
return $b/description
```

Quadro 2.7: Expressão EVERY

2.2 Extensão de XQuery para consultas de similaridade

Esta seção apresenta como um processador de consultas XQuery foi utilizado de forma a embutir funções de similaridade na linguagem. As funções de similaridade foram introduzidas na linguagem através do recurso de *namespaces*.

Um *namespace* é um recurso que permite a especialização e extensão de documentos XML e por isso também foi implementado na linguagem XQuery. Trata-se de um *container* abstrato que permite identificar de forma única seus itens, permitindo que nomes menores tenham um único significado. Assim, foi declarado um *namespace* para embutir, em XQuery, as funções de similaridade que serão apresentadas neste trabalho.

As funções supracitadas já estavam implementadas em Java. Com o objetivo de fazer a comunicação entre Java e XQuery, foi utilizada a bibilioteca NUX (BERKELEY LAB, s.d.) , a qual é uma implementação de XQuery em Java. Essa bibilioteca permite que funções Java declaradas em um *namespace* sejam chamadas e executadas a partir de consultas XQuery.

Utilizando *namespaces*, a inclusão dessas funções e a sua utilização em XQuery tornou-se simples. Veja no Quadro 2.8 um exemplo de utilização de *namespaces* com funções em Java.

```
declare namespace simi = "java:xsimilarity.Similaridade";
for $b in doc("bdbcomp.xml")/oaidc/creator
return
if ( simi:levenshtein($b,"Andre Costa Carvalho")>0.5)
then
$b
else()
```

Quadro 2.8: Consulta XQuery referenciando Java

Nesse exemplo foi declarado o *namespace* simi que referencia a classe Similaridade do pacote xsimilarity em java. A partir disso pôde-se usar a função levenshtein, implementada na referida classe, para comparar duas *strings* em XQuery.

2.3 Considerações Finais

No presente capítulo foi estudada a sintaxe da linguagem de consulta XQuery e foram apresentados alguns exemplos que ilustram o seu funcionamento. Também foi visto como é possível estender essa liguagem para comportar o uso de funções de similaridade.

XQuery, nesse trabalho, é a linguagem que possibilita a execução das funções de similaridade que são descritas no capítulo 3. Esse estudo foi importante para a construção da ferramenta porque a linguagem XQuery tornou-se parte central da mesma, juntamente às funções de similaridade.

3 FUNÇÕES DE SIMILARIDADE

Este capítulo apresenta os principais conceitos sobre consulta por similaridade que serve de base para a compreensão da ferramenta implementada neste trabalho. São apresentadas em detalhes as principais características das funções de similaridade mais conhecidas e mais utilizadas, sendo também justificados os motivos que levaram a escolha desses algoritmos. Em seguida, é apresentado um estudo comparativo sobre as funções de similaridade apresentadas. O capítulo é encerrado com as considerações finais.

3.1 Conceitos básicos

As funções de similaridade servem para determinar o quão parecidas são duas *strings*. Elas retornam um valor no intervalo [0;1] onde 0 implica que são totalmente diferentes e 1 que são idênticas. Esse valor é calculado com base em uma métrica de distância a ser escolhida dependendo da função de similaridade. O cálculo é feito de acordo com a fórmula do exemplo abaixo (1), onde *d* é o valor obtido pela métrica de distância e *maxlength* é o maior valor que pode ser obtido por esta métrica.

s(Rio Grande, Rio Graande) = 1 – d / maxlength = 1 – 1/10 = 0.9 = 90% (1)

As funções de similaridade podem ser classificadas em baseadas em *token*, baseadas em caractere ou híbridas. As funções baseadas em *token* dividem as *strings* de entrada em palavras separadas por espaços e usam essas palavras como elementos principais de suas métricas. Nestas funções, os *tokens* de uma *string* são comparados com os *tokens* de outra e apenas serão considerados similares se forem totalmente idênticos. As funções baseadas em caractere são inspiradas na distância de edição para comparar todos os caracteres de uma *string* com os caracteres de outra. A distância de edição é definida como o número de operações necessárias para transformar uma *string* na outra. Por fim, as funções híbridas usam ambos os conceitos mencionados anteriormente para comparar duas *strings*.

Funções de similaridade podem ser utilizadas em diversos contextos. Há registros de sua utilização para auxiliar na análise de DNA, na detecção de plágio, na busca de textos quando não se sabe a grafia correta, cruzamento de registros em banco de dados, entre outros exemplos. Neste trabalho, pretende-se utilizá-las para executar consultas por similaridade em arquivos XML. Dependendo da finalidade desejada, uma ou outra função pode ser mais adequada.

Estas funções podem ser embutidas em linguagens de consulta, como a XQuery, ou qualquer linguagem de programação de alto nível. Ao fazer isto, permite-se a construção de consultas ou sistemas que auxiliem nas tarefas mencionadas anteriormente.

3.2 Funcões de similaridade

Esta seção apresenta algumas das principais funções de similaridade utilizadas nas áreas de bancos de dados e recuperação de informação. Para cada função, será apresentada sua definição e seu modo de uso. Também será dado um exemplo prático de utilização e mencionada a aplicabilidade de cada função apresentada. Ao final, será feita uma análise comparativa entre as funções abordadas.

3.2.1 Dice's Coefficient

O coeficiente de Dice (DICE , 1945) é uma métrica de distância baseada em *tokens*. Cada *token* de uma *string* é comparado com os de outra *string* e somente são considerados para essa métrica os *tokens* iguais. A medida de similaridade é definida como duas vezes o número de *tokens* comuns das strings dividida pelo total de número de *tokens* de ambas conforme a equação abaixo (2).

$$dice = \frac{2 \times |a \cap b|}{|a| + |b|} \qquad (2)$$

Sejam a=" University of California" e b="California University of" então o número de tokens que são intersecção nessas strings é três. A equação (2) aplicada ao exemplo está descrita abaixo (3).

Dices("University of California", "California University of")=2 x 3 / (3 + 3)
=1=100% (3)

Em outro exemplo (4), vemos que o coeficiente de Dice pode ser muito ineficaz, pois em duas *strings* com grande similaridade, a função retorna nada de similaridade.

Dices("University of California", "Universities in Callifornia")= 2 x 0 /(3 + 3)
=0=0% (4)

De acordo com o exposto anteriormente, *strings* com os mesmos *tokens* possuem 100% de similaridade, não importando a ordem. Entretanto, esta métrica falha quando *strings* muito parecidas não apresentam *tokens* iguais.

3.2.2 Jaccard Similarity

A métrica de similaridade de Jaccard (Cohen, 2003) é baseada em *tokens* e é descrita pela equação a seguir (5) . Jaccard considera as *strings* de entrada como palavras separadas por espaços para realizar a comparação.

$$jaccard = \frac{|S \cap T|}{|S \cup T|} \qquad (5)$$

Sejam S e T conjuntos de *tokens* derivados da quebra em palavras menores de duas *strings* s e t. A métrica de Jaccard retorna o quociente do número de *tokens* que

representam a intersecção dos conjuntos S e T pelo número de *tokens* que representam a união desses conjuntos.

No exemplo a seguir (6), há dois *tokens* comuns entre as duas *strings* (University e California), e há três *tokens* no total, desconsiderando-se as repetições. Desta forma, basta realizar a divisão 2/3 para obter o resultado final.

Jaccard("University of California", "California University") = 0,6666667 (6)

A principal aplicabilidade é o casamento de registros onde a ordem dos *tokens* não é levada em consideração no cálculo da similaridade.

3.2.3 Hamming distance

A distância de Hamming (HAMMING , 1950) é definida como o menor número de elementos que precisam ser modificados para transformar uma palavra em outra. Neste contexto, o ideal é considerar cada caractere como um elemento. Em outros casos, um elemento pode ser um bit, ou outro conceito qualquer. O cálculo da distância de Hamming tem complexidade O(n), sendo a forma mais simples de medir a distância entre duas *strings*.

A função Hamming retorna um valor entre 0 e 1 que representa a proporção de caracteres iguais entre as duas *strings*, indicando uma medida de similaridade. Caso as *strings* não sejam do mesmo tamanho, faz-se a comparação levando em conta o tamanho da menor seqüência de caracteres. O excedente da maior cadeia é descartado para a comparação. A função Hamming de similaridade é classificada como baseada em caractere.

A principal aplicabilidade é a análise de seqüências binárias, pelo fato de serem do mesmo tamanho. É muito relevante nas disciplinas de teoria da informação, códigos de detecção e correção de erros e criptografia. Entretanto, esta função é pouco usada para avaliar a similaridade entre palavras de tamanhos distintos, onde espera-se a inserção ou deleção de substrings.

O problema com esta métrica é que aparentemente *strings* muito similares podem ter uma distância de *Hamming* muito grande. As duas *strings* abaixo são muito semelhantes. Entretanto, como a maior parte dos caracteres semelhantes de uma não está em posição correspondente na outra, *Hamming* não detecta sua grande semelhaça, conforme exemplo a seguir (7) .

Hamming(“THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG”,
“MY QUICK BROWN FOX JUMPED OVER THE LAZY DOGS”)=0.0 (7)

As *strings* “aaaaaaaa” e “abbaabab” têm uma distância de *Hamming* de 4 caracteres, (4 caracteres são diferentes em posições correspondentes entre elas).

As cadeias de DNA abaixo têm mesmo tamanho e diferem de dois nucleotídeos (posições 2 e 8), tendo uma distância de Hamming igual a dois.

AAGTACATTGC
AGGTACACTGC

Esta métrica é eficiente para comparar *strings* que sofram apenas substituições de caracteres em relação a outras *strings*. Dessa forma, uma aplicação é comparar seqüências de *string* de DNA.

3.2.4 Levenshtein distance

Distância de Levenshtein (Levenshtein, 1966) foi nomeada em homenagem ao cientista russo Vladimir Levenshtein, que desenvolveu o algoritmo em 1965 (Borges, 2005). Também sendo conhecida pelo nome de edit distance. É uma função baseada em caractere e serve com base para o cálculo da função de similaridade de mesmo nome.

A distância de Levenshtein entre duas *strings* é dada pelo menor número de operações necessárias para transformar uma *string* em outra, onde uma operação pode ser uma inserção, deleção ou substituição de um caractere. Pode ser considerada uma generalização da distância de Hamming, a qual é usada para *strings* do mesmo tamanho e considera apenas a edição de substituição.

A distância de Levenshtein é um algoritmo de complexidade $\Theta(m \times n)$ que tem como função calcular a distância entre duas *strings*, onde m e n são os tamanhos das mesmas.

		-1	0	1	2	3	4
			g	a	t	o	s
-1		0	1	2	3	4	5
0	a	1	1	1	2	3	4
1	m	2	2	2	2	3	4
2	a	3	3	2	3	3	4
3	t	4	4	3	2	3	4
4	a	5	5	4	3	3	4

Figura 3.1: Algoritmo de Levenshtein

Na Figura 3.1, a distância de Levenshtein é calculada para as strings “gatos” e “amata”. Cada célula contém um valor associado que representa o custo de executar as operações de transformação até então. A computação começa preenchendo-se com zero a célula (-1,-1). As demais células são computadas a partir das células adjacentes superior, diagonal superior esquerda e esquerda, em um processo de escolha. Computa-se um possível valor resultante a partir de cada célula e escolhe-se, ao final, aquele que representar o menor custo para a célula sendo calculada. O custo para transformar “gatos” em “amata” está no canto inferior direito, cujo valor é 4.

O custo calculado para uma célula depende da célula utilizada como base. No caso da célula base ser a diagonal, se os caracteres associados à célula sendo computada forem iguais, então o valor resultante será igual ao da célula diagonal. Se os caracteres forem diferentes, deve-se somar um ao valor da célula da diagonal para obter o valor resultante. Neste caso, o custo é derivado de uma substituição dos caracteres. No caso da célula superior, também deve-se somar um, pois o custo é derivado de uma remoção

de caracter na string “gatos”. Por fim, no caso da célula esquerda, o custo é derivado de uma inserção de caractere na *string* “amata”, também exigindo que se some um para obter o valor resultante.

A principal aplicação é determinar o quão similar duas *strings* são, como feito em verificadores de ortografia e gramática. Distância de Levenshtein tem sido usada também em reconhecedores de voz, análise de DNA e detecção de plágio.

Esta métrica é muito usada para comparar *strings* que são relativamente pequenas e que não precisam necessariamente ter o mesmo tamanho.

3.2.5 Jaro distance metric

O método de Jaro (Jaro , 1989) compara os caracteres comuns entre duas *strings*. Dadas as *strings* $s=a_1...a_k$ e $t=b_1...b_k$, um caractere a_i em s é comum com t se existe um $b_j=a_i$ em t tal que $i-H<=j<=i+H$, onde $H=\min(|s|,|t|)/2$. Sejam $s'=a'_1...a'_k$ os caracteres em s que são comuns com t (na mesma ordem em que aparecem em s) e seja $t'=b'_1...b'_k$, construído de forma análoga a s’. Uma transposição para s’ e t’ corresponde a uma posição i tal que $a'_i \neq b'_i$. Assim, $T_{s',t'}$ é definido como a metade do número de transposições para s’ e t’. A fórmula de Jaro é apresentada em (8).

$$Jaro(s,t) = \frac{1}{3} \times \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|} \right) \quad (8)$$

O exemplo a seguir mostra o valor de similaridade entre duas *strings* de acordo com a função de similaridade de Jaro. No exemplo, s’ é calculado como “Tiago Silv”, sendo igual a t’. Calculando a fórmula de Jaro temos:

$$\text{Jaro("Thiago Silva","Tiago Silvio")} = (1/3)*(10/12 + 10/12 + (10-0)/10) = 32/36 = 0.88888884 \quad (9)$$

Essa função de similaridade serve principalmente para fazer a integração de bases de dados heterogêneas e o casamento de metadados. No geral, tudo que o que é possível fazer com as demais funções baseadas em caracteres é factível com a Jaro. Essa função tem uma complexidade O(m x n) e é boa para detectar erros de grafia.

3.2.6 Jaro Winkler

A função de Jaro Winkler (Winkler, 1990) é uma extensão da distância de Jaro. Essa métrica adiciona uma bonificação ao resultado da Jaro no sentido de valorizar *strings* que tenham um prefixo em comum. Jaro winkler tem uma complexidade de O(m x n) e seu resultado é muito semelhante à função de similaridade Jaro, conforme pode ser visto na equação a seguir (10).

$$Jaro-Winkler(s,t) = Jaro(s,t) + \frac{P'}{10} \times (1 - Jaro(s,t)) \quad (10)$$

Na equação (10) , P’ =min(P, 6), onde P é o tamanho do maior prefixo comum entre s e t. O exemplo a seguir (11) ilustra o cálculo da função Jaro Winkler para o mesmo exemplo calculado com a função Jaro.

JaroWinkler("Thiago Silva","Tiago Silvio") = (32/36) + (1/10)*(4/36) = 0,9 (11)

As métricas de Jaro e Jaro-Winkler parecem ser mais adequadas à comparação de *strings* pequenos como primeiros e últimos nomes (Cohen , 2003). A Jaro-Winkler pode ser utilizada nas mesmas aplicações da função de Jaro.

3.2.7 Monge Elkan distance

A Monge Elkan distance (Monge , 1996) é uma função de distância híbrida, pois trabalha tanto com *tokens*, dividindo as *strings* de entrada em palavras separadas, quanto com os caracteres individuais dessas palavras. A ordem dos *tokens* não é levada em consideração no cálculo da similaridade.

Como está representado na equação a seguir (12), a função opera quebrando ambas as *strings* de entrada (A e B) em *tokens* (A_i e B_j), e buscando, através de similaridade, qual *token* na *string* B melhor corresponde a cada *token* na *string* A. Ao final, é retornada uma média dos valores de similaridade obtidos para cada *token* da *string* A. Para calcular a similaridade entre os *tokens*, uma função secundária de similaridade baseada em caractere é utilizada ($match_2$). A função implementada na ferramenta que será apresentada utiliza a Gotoh Distance como função secundária.

$$match(A,B) = \frac{1}{|A|} \times \sum_{i=1}^{|A|} \max_{j=1}^{|B|} \text{match}_2\,(\text{Ai},\text{Bj}) \qquad (12)$$

O exemplo abaixo mostra o valor de similaridade entre as duas *strings* de acordo com a função de similaridade Monge Elkan. Cada token da primeira *string* (por exemplo "University") é comparado é comparado com cada *token* da segunda *string*. Nesse caso, o token mais similar a ser encontrado na segunda string será "Univ.", com valor de similaridade computado em 0.33. A média das similaridades para todos os *tokens* da primeira *string* é 0.90833336.

MongeElkan ("University of California, San Diego, Computer Science Department" , "Dept. of Computer Science, California Univ., San Diego, USA") = 0.90833336

Esta função é eficiente para comparar texto longos, com a finalidade de detecção de plágio, por exemplo. A principal aplicabilidade é o casamento de registros.

3.3 Análise comparativa

Esta seção apresenta uma análise comparativa entre as funções de similaridade apresentadas na seção 3.2. Primeiramente, são apresentados os critérios utilizados para comparação. Em seguida, é feita a análise de cada critério. A análise é finalizada com uma tabela comparativa (Tabela 3.1) que resume as características comentadas.

Os seguintes critérios foram utilizados para realizar a comparação das funções:

- comparação de *strings* de tamanhos diferentes: avalia a capacidade das funções em comparar strings de tamanhos diferentes;
- adaptação a pequenas variações: avalia se a função detecta quando apenas um ou outro caractere foi modificado entre duas *strings*;

- comparação de frases: avalia se a métrica aceita a troca de ordem entre palavras em uma frase;
- comparação de palavras: avalia se a função detecta diferentes graus de similaridades entre duas palavras (mais de dois);
- desempenho: avalia se a função tem um desempenho superior ou inferior em relação as demais e é baseado no custo computacional das funções.

As funções foram classificadas nas seguintes categorias:

- baseada em *token*: indica que a métrica de similaridade faz a comparação entre as *strings* usando como elemento menor *tokens*;
- baseda em caractere: indica que a métrica faz a comparação usando como elemento menor caracteres;
- híbrida: indica que a função usa *tokens* e caracteres em suas comparações.

Uma característica muito relevante em funções de similaridade é poder comparar *strings* com tamanhos diferentes. Observa-se que apenas a distância de Hamming não satisfaz este quesito.

A limitação de comparar apenas *strings* de tamanhos iguais, por outro lado, permite melhorar o desempenho da função de Hamming. Seu algoritmo fica com complexidade linear em decorrência de sua simplicidade. Por não precisarem comparar caractere a caractere, em geral, espera-se que as funções basedas em *tokens* sejam mais eficientes que as baseadas em caractere ou hibridas. Em resumo, no quesito desempenho, a função mais bem classificada é a Hamming. Logo em seguida estão as funções baseadas em *tokens*, seguidas pelas baseadas em caractere (exceto Hamming).

Idealmente, as funções de similaridade devem ser capazes de aceitar pequenas variações de grafia em *strings* sem comprometer o nível de similaridade. Nos algoritmos de similaridade baseados em caractere apresentados, com exceção ao algoritmo de Hamming, esta característica está presente. Somente comparando-se caractere a caractere duas *strings*, é possível detectar a deleção, inserção ou substituição de caracteres. As funções de similaridade Levenshtein, Jaro , Jaro Winkler e a função híbrida de Monge Elkan tratam adequadamente variações de grafia.

Outra característica relevante de uma função de similaridade é a capacidade de comparar frases de maneira eficaz. As funções baseadas em *tokens* são melhores nesse sentido, pois é necessário quebrar as *strings* em suas partes constituintes para permitir uma análise que leve adequadamente em consideração alterações na ordem das partes.

Para comparar palavras, as funções baseadas em caractere e as híbridas são mais eficazes. Essas funções possibilitam uma comparação mais apropriada para a similaridade entre elas. Por exemplo, duas palavras, quando comparadas usando-se uma métrica de *token*, são consideradas similares ou não, sem meio termo. Entretanto, as classes de funções mencionadas inicialmente nesse parágrafo conseguem identificar graus intermediários de similaridade entre essas duas palavras.

As informações apresentadas nessa seção estão resumidas na Tabela 3.1.

Tabela 3.1: Comparação entre as funções de similaridade abordadas neste trabalho.

	Dice	**Jaccard**	**Hamming**	**Levenshtein**	**Jaro**	**Jaro winkler**	**Monge Elkan**
Compara *strings* de tamanho diferente	x	x		x	x	x	x

Adaptação a pequenas variações de grafia				x	x	x	x
Adequada para comparação de frases	x	x					x
Adequada para comparação de palavras				x	x	x	x
Desempenho (1=maior, 2=intermediário, 3=menor)	2	2	1	3	3	3	3
Baseada em tokens	x	x					
Baseada em caracater			x	x	x	x	
Híbrida							x

O conjunto de funções de similaridade descritas nesta seção envolve métricas baseadas tanto em caracteres, palavras (tokens) ou híbridas. Ainda possuem diferentes custos computacionais e aplicabilidade variada. Estas diferentes características fornecem maior flexibilidade para experimentos dos mais diversos tipos, o que justifica a escolha destas funções para a implementação da ferramenta detalhada no capítulo 4.

3.4 Considerações finais

Neste capítulo foram apresentados os conceitos básicos referentes às funções de similaridade implementadas na ferramenta que está descrita no capítulo 4. Para cada função, foram abordados seu modo de funcionamento, vantagens e/ou desvantagens e identificada a sua aplicabilidade. Exemplos foram apresentados para ilustrar a utilização das funções. Ao final, foi realizada uma análise comparativa para retomar as características das funções.

Terminado este capítulo, espera-se que o leitor tenha adquirido a fundamentação teórica que irá facilitar o entendimento da ferramenta desenvolvida e da importância de cada função implementada.

4 XSIMILARITY

Este capítulo descreve a ferramenta XSimilarity desenvolvida para a realização de consultas por similaridade embutidas na linguagem XQuery. As seções que seguem apresentam uma visão geral da ferramenta e uma descrição mais detalhada de seu funcionamento e interface. Por fim, são apresentados exemplos de consultas por similaridade com dados de artigos científicos da área da Ciência da Computação oriundos da biblioteca digital DBLP com o objetivo de comprovar a validade da ferramenta, e são feitas as considerações finais.

4.1 Visão geral da ferramenta

A ferramenta XSimilarity foi desenvolvida para suprir a necessidade de um mecanismo de consulta por similaridade em bases de dados XML. Para tanto, a ferramenta permite a execução de consultas XQuery em que funções de similaridade podem ser embutidas através de um mecanismo de *namespace*. O uso das funções de similaridade possibilita a otimização da recuperação de informações nessas bases de dados.

A ferramenta basicamente implementa uma interface de usuário para a biblioteca de execução XQuery NUX (BERKELEY LAB, s.d.). As funções de similaridade são disponibilizadas através de um *namespace* na linguagem XQuery. Em sua maioria, as funções já estavam implementadas na biblioteca *Open Source* SimMetrics (CHAPMAN, s.d.). As diversas funções de similaridade disponibilizadas na ferramenta permitem atender a propósitos de busca variados.

As funcionalidades incluídas na interface de usuário podem ser vistas na Figura 4.1, a qual apresenta o diagrama de casos de uso da ferramenta.

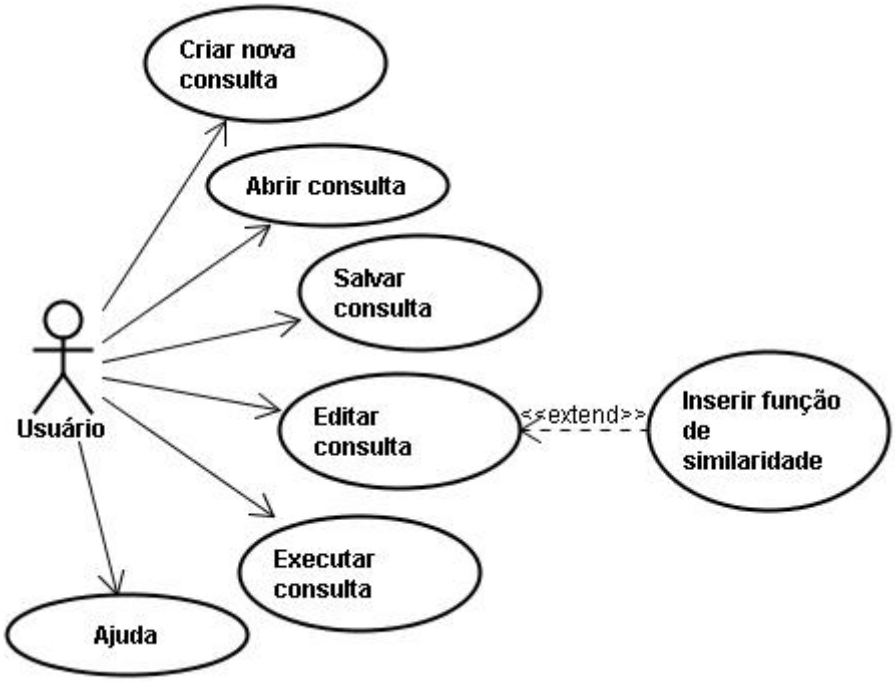

Figura 4.1: Funcionalidades da ferramenta

Todas as funcionalidades constantes no diagrama de casos de uso foram implementadas, com exceção da funcionalidade de "Ajuda". As três primeiras funcionalidades referem-se à manipulação de arquivos de consulta, os quais são salvos com a extensão "xq". As três funcionalidades restantes são centrais na ferramenta.

A funcionalidade de "Editar consulta" permite que a consulta seja editada, como em um editor de texto simples. É possível realizar todas as operações básicas de edição, incluindo copiar, recortar, colar, selecionar e deletar as *strings* desejadas.

A funcionalidade de "Inserir função de similaridade" estende a funcionalidade de "Editar consulta", permitindo que sejam inseridas funções de similaridade na consulta sendo editada. Mesmo sem utilizar esta funcionalidade, o usuário pode editar a consulta diretamente e utilizar as funções de similaridade. Entretanto, ao utilizar a funcionalidade de "Inserir função de similaridade", o usuário terá auxílio da ferramenta na montagem da função a ser inserida, incluindo a descrição das funções disponíveis, suas assinaturas, parâmetros e *namespace* que deve ser utilizado.

Por fim, a funcionalidade de "Executar consulta" permite que a consulta XQuery sendo editada na ferramenta seja executada. Consultas com a sintaxe adequada são executadas e seu resultado é retornado na própria ferramenta.

4.1.1 Recursos utilizados

A ferramenta foi implementada em Java, sendo utilizada a biblioteca NUX como mecanismo de execução XQuery e a biblioteca SimMetrics como fonte de funções de similaridade. Também é utilizado o pacote Java *swing* para a definição da interface.

Internamente é construída uma classe para permitir a disponibilização das funções de similaridade em um *namespace* da XQuery. O nome desta classe aparece na declaração do *namespace*, que fica como "java:xsimilarity.Similaridade".

Na Figura 4.2 é apresentada de forma gráfica a arquitetura da ferramenta. Considerando-se que a ferramenta utiliza a API Java Swing para a sua interface e a Nux como mecanismo de execução XQuery e ambas são externas a ferramenta , elas são representadas abaixo do módulo XSimilarity. Como Swing e Nux não se comunicam pois não têm relação entre si foi colocado um espaço em branco entre elas na figura da arquitetura. A ferramenta implementa um namespace denominado Similaridade que utiliza a bibilioteca de funções SimMetrics que é externa à ferramenta.

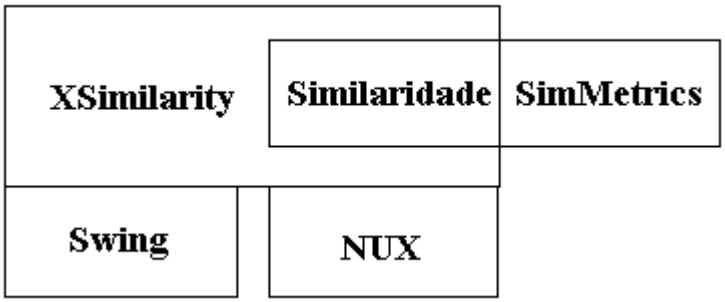

Figura 4.2: Arquitetura da Ferramenta

4.2 Descrição da ferramenta

Esta seção apresenta em detalhe os recursos da interface gráfica da ferramenta e seu modo de uso. Com o objetivo de validar o uso da ferramenta, são apresentados exemplos de consultas XQuery que utilizam funções de similaridade. As consultas são executadas sobre um subconjunto de registros da base de dados XML da bibilioteca digital DBLP.

4.2.1 A interface principal do programa

A interface principal do programa consiste basicamente de duas caixas de texto, conforme pode ser observado na Figura 4.3. A primeira caixa serve para o usuário editar a consulta a ser executada, enquanto a segunda caixa serve para apresentar o resultado de uma execução. A segunda caixa é preenchida quando o usuário solicita a execução de sua consulta. Por flexibilidade, a segunda caixa também permite edição.

Na parte superior da interface principal há uma série de botões. Os três primeiros apresentam as funções clássicas “novo”, “abrir” e “salvar”, respectivamente. No caso desta ferramenta, estes botões atuam sobre a consulta. Ao clicar no botão “novo”, a caixa que contém a consulta a ser executada é esvaziada e o usuário poderá criar uma nova consulta do zero. Os arquivos de consulta são salvos em formato texto. A extensão “xq” é utilizada para diferenciá-los de outros arquivos.

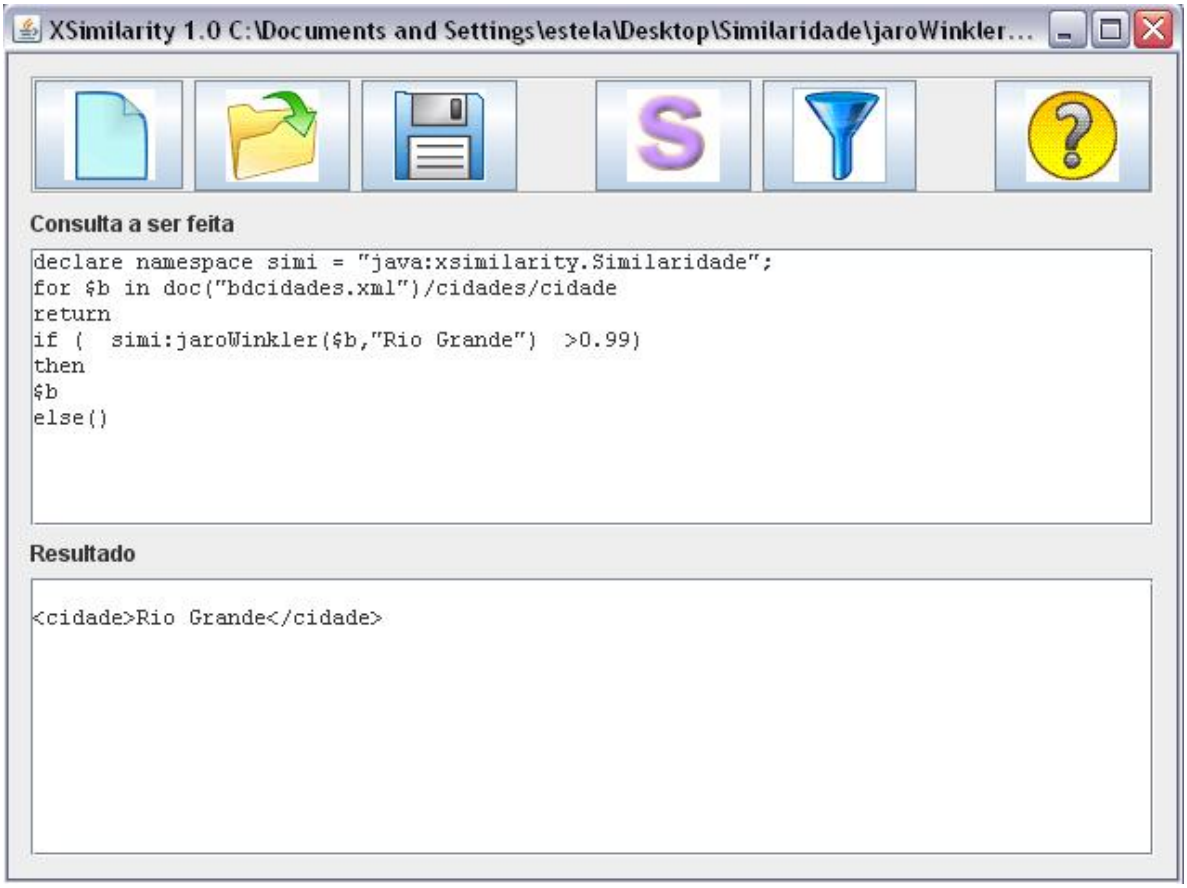

Figura 4.3: Interface principal da ferramenta

Dois botões oferecem as funcionalidades centrais da ferramenta. O botão com a imagem de um " S" é o que permite inserir funções de similaridade. O efeito deste botão será detalhado mais adiante. O botão com a figura de um funil é o que permite executar uma consulta XQuery. Os arquivos a serem consultados são referenciados na própria consulta XQuery, assim como as funções de similaridade utilizadas.

O último botão da interface, que contém a imagem de uma interrogação, permite ao usuário acessar a ajuda da ferramenta. Embora seja útil para esclarecer dúvidas sobre a utilização da XSimilarity, a ajuda da ferramenta ainda não foi disponibilizada.

4.2.2 A tela de abrir consulta

Quando o usuário clica no botão com a imagem de uma pasta amarela sendo aberta na interface principal, a ferramenta exibe a tela de abrir consulta, que pode ser visualizada na Figura 4.4. Pode-se clicar sobre o nome do arquivo com extensão "xq" ou digitar o nome do arquivo desejado diretamente na caixa "File Name". Ao clicar no botão "Open", a consulta selecionada será aberta na ferramenta e estará pronta para ser editada na interface principal. Para abrir arquivos em outros diretórios, o usuário pode utilizar os recursos de navegação existentes na parte superior da tela.

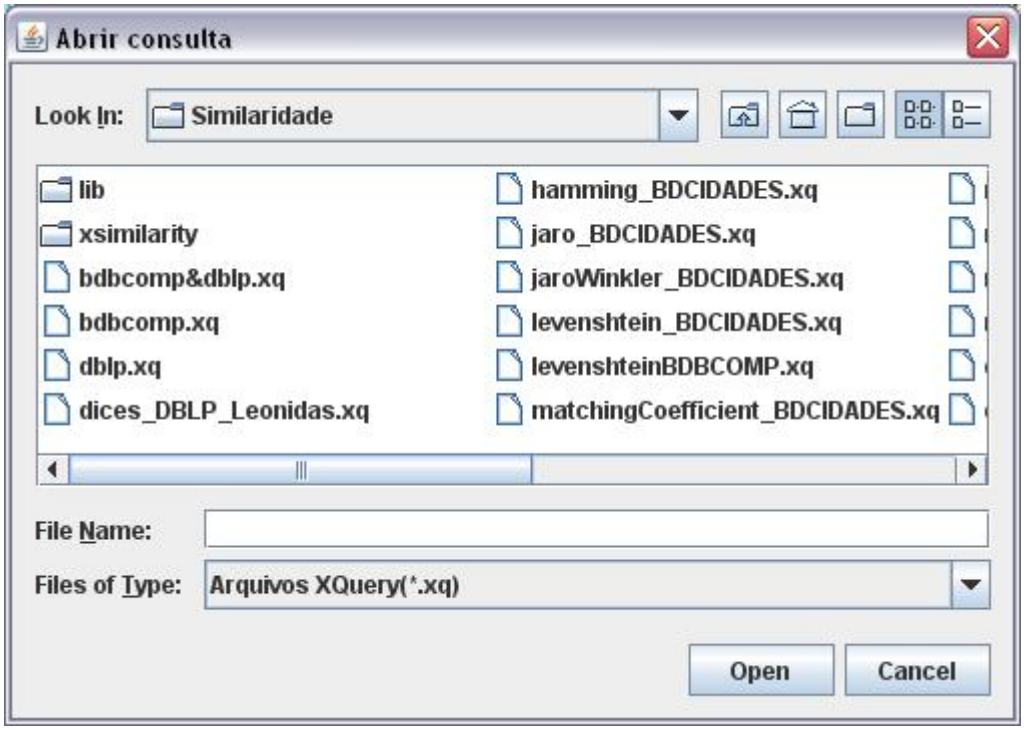

Figura 4.4 : Janela de abrir consulta

4.2.3 A tela de salvar consulta

Quando o usuário clica no botão com a imagem de um disquete azul na interface principal, a ferramenta exibe a tela de salvar consulta, que pode ser visualizada na Figura 4.5. Pode-se clicar sobre o nome do arquivo com extensão “xq” ou digitar o nome do arquivo desejado diretamente na caixa “File Name”. Ao clicar no botão “Save”, a consulta sendo editada será salva no diretório escolhido com o nome selecionado. A escolha do diretório onde a consulta será salva é feita através dos recursos de navegação existentes na parte superior da tela.

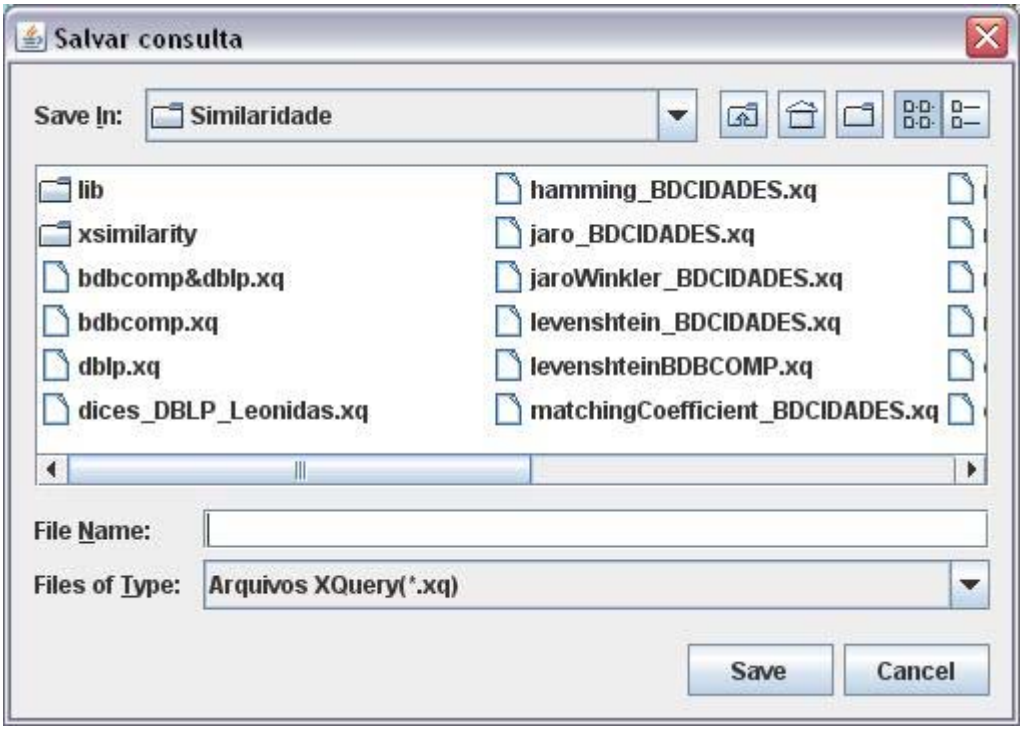

Figura 4.5 : Janela de salvar consulta

4.2.4 A tela de inserção de função de similaridade

Quando o usuário clica no botão com a imagem de um "S" na interface principal, o sistema abre uma nova janela para permitir a escolha da função de similaridade a ser inserida. Esta janela pode ser vista na Figura 4.6.

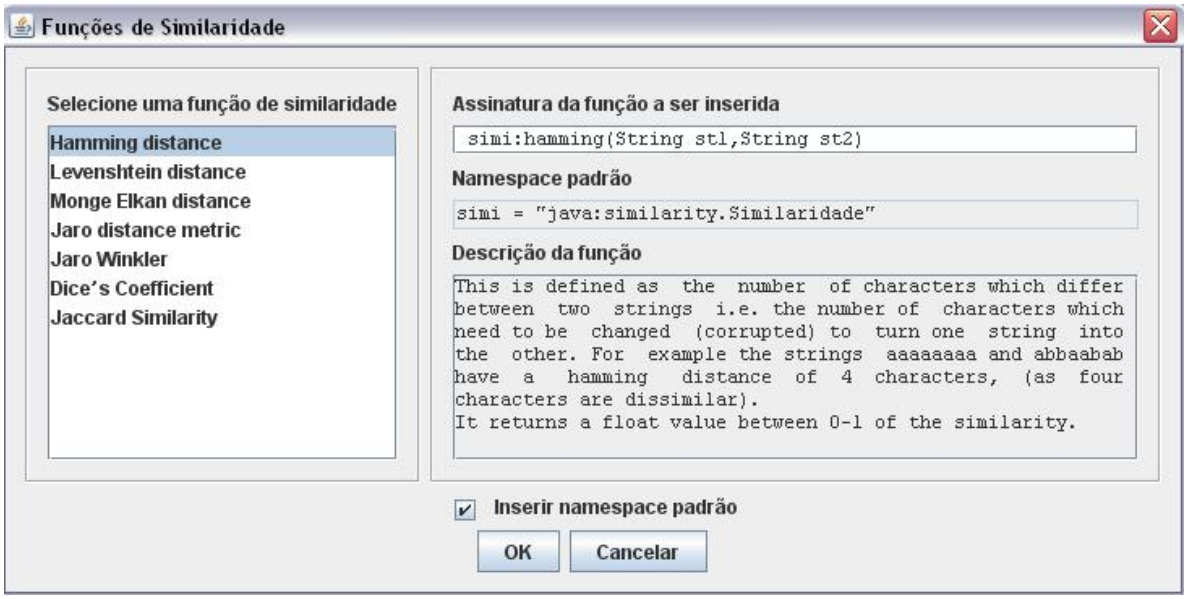

Figura 4.6: Janela de inserção de função de similaridade

Esta janela permite que o usuário visualize a assinatura da função de similaridade que deseja utilizar. O usuário seleciona a função desejada na lista à esquerda e o sistema preenche a assinatura, o *namespace* e a descrição da função nas caixas de texto à direita. Todas as funções descritas na seção 3.2 foram disponibilizadas.

Antes de confirmar clicando em OK, o usuário pode editar a assinatura para adaptá-la a sua consulta. Ao editar a assinatura na janela de inserção de função, o usuário terá à sua disposição uma descrição detalhada da função escolhida e de seus parâmetros, o que facilita a adaptação da função. Ao navegar por várias funções na lista à esquerda, o usuário poderá aprender mais sobre cada função de similaridade.

Outra funcionalidade importante dessa caixa de diálogo é permitir que o usuário marque a inserção do *namespace* padrão da função em sua consulta. Como o *namespace* das funções de similaridade é implementado em uma única classe Java, é necessário inseri-lo apenas na primera vez que se insere uma função. Dessa forma, o *checkbox* responsável por habilitar essa inserção permance desabilitado por *default*. Quando esse *checkbox* é habilitado, o *namespace* padrão é inserido na primeira linha do texto da consulta.

Ao clicar em OK, a assinatura da função, como consta na caixa de assinatura, é inserida na posição do cursor na caixa de consulta da interface principal. Se desejar, o usuário também pode cancelar a inserção da função de similaridade. Nesse caso, a caixa de diálogo é encerrada sem nenhum efeito colateral.

4.2.5 Exemplos de consultas

Nesta seção são apresentados alguns exemplos de consultas que validam o funcionamento da ferramenta desenvolvida. As consultas são executadas sobre parte da base de dados DBLP (University of Trier , s.d.), que é fornecida em formato XML. Foram utilizados 163 elementos incollection da DBLP (que representam artigos) e o tamanho em bytes total da base XML consultada era de 187 KB. Assim o tamanho médio de cada elemento foi de aproximadamente 1,15 KB.

A base DBLP possui um arquivo DTD que define o esquema XML adotado no arquivo de dados. O arquivo DTD foi posicionado juntamente ao arquivo da base XML para que a ferramenta pudesse interpretar corretamente a base XML. A validação e interpretação da base de dados conforme a DTD será feita pelo processador XML da NUX. Uma parte do arquivo de dados pode ser vista no Quadro 4.1.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE dblp SYSTEM "dblp.dtd">
<dblp>
<incollection mdate="2002-01-03"
key="books/acm/kim95/AnnevelinkACFHK95">
<author>Jurgen Annevelink</author>
<author>Rafiul Ahad</author>
<author>Amelia Carlson</author>
<author>Daniel H. Fishman</author>
<author>Michael L. Heytens</author>
<author>William Kent</author>
<title>Object SQL - A Language for the Design and
Implementation of Object Databases.</title>
<pages>42-68</pages>
<year>1995</year>
<booktitle>Modern Database Systems</booktitle>
<url>db/books/collections/kim95.html#AnnevelinkACFHK95</url>
</incollection>
```

```
<incollection mdate="2002-01-03"
key="books/acm/kim95/Blakeley95">
<author>Jos&eacute; A. Blakeley</author>
<title>OQL[C++]: Extending C++ with an Object Query
Capability.</title>
<pages>69-88</pages>
<booktitle>Modern Database Systems</booktitle>
<url>db/books/collections/kim95.html#Blakeley95</url>
<year>1995</year>
</incollection>

...

</dblp>
```

Quadro 4.1 : Base de dados DBLP

Na Figura 4.7 tem-se um exemplo de consulta XQuery onde é utilizada a função de similaridade Monge Elkan para recuperar um registro na base da DBLP. Existia uma diferença entre o nome do autor consultado e o nome real desse autor armazenado na DBLP. Entretanto, devido ao uso da função de similaridade Monge Elkan, foi possível recuperar o registro mesmo sem o nome adequado de um dos autores.

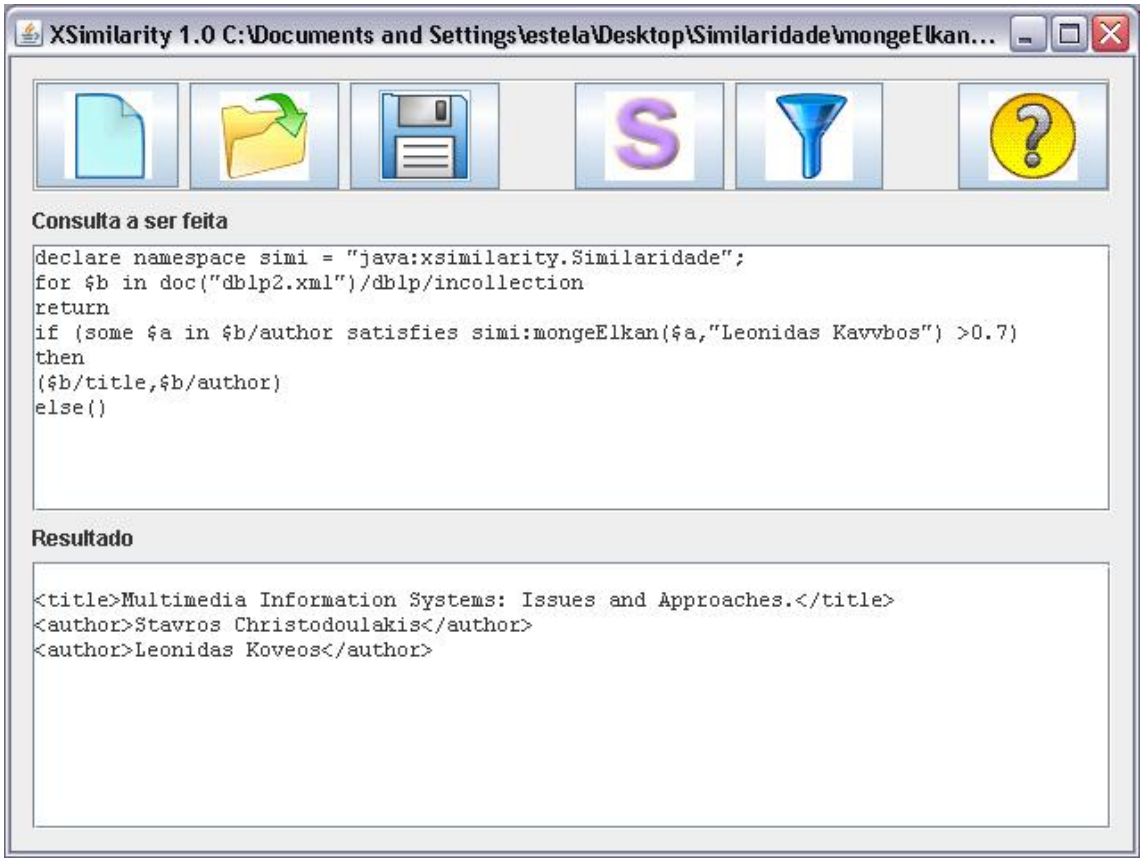

Figura 4.7 : Exemplo Monge Elkan

A Figura 4.8 ilustra um exemplo que mostra a melhor eficácia da função de similaridade Monge Elkan em relação à Dice. Nesse exemplo, a mesma consulta anterior (Figura 4.7), quando realizada com a função Dice, não obtém o resultado

esperado. Já na Figura 4.9, onde o grau de similaridade exigido da função Dice é menor, é retornado o mesmo resultado da função Monge Elkan obtido anteriormente.

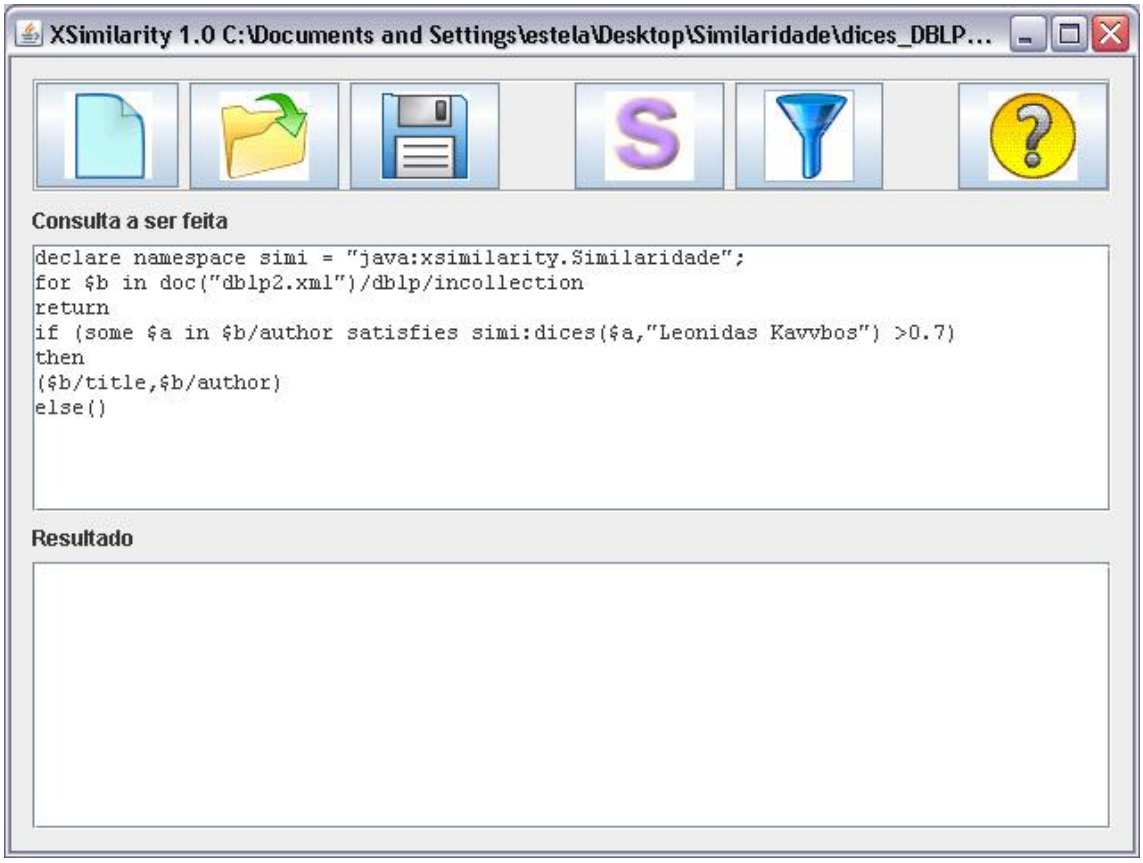

Figura 4.8: Dice's Similarity não retorna resultado

Outro resultado possível ao se utilizar consultas por similaridade é a obtenção de elementos não requisitados. Na Figura 4.10, na falta de elementos que correspondam ao nome consultado, utilizou-se o limiar de similaridade 0.6 para buscar os registros com a função Monge Elkan. A função Monge Elkan retornou nomes que são considerados semelhantes ao nome “maria”. Entretanto, alguns nomes não eram exatamente esperados.

Com os resultados apresentados, verificou-se o correto funcionamento da ferramenta e sua utilidade para testar funções de similaridade.

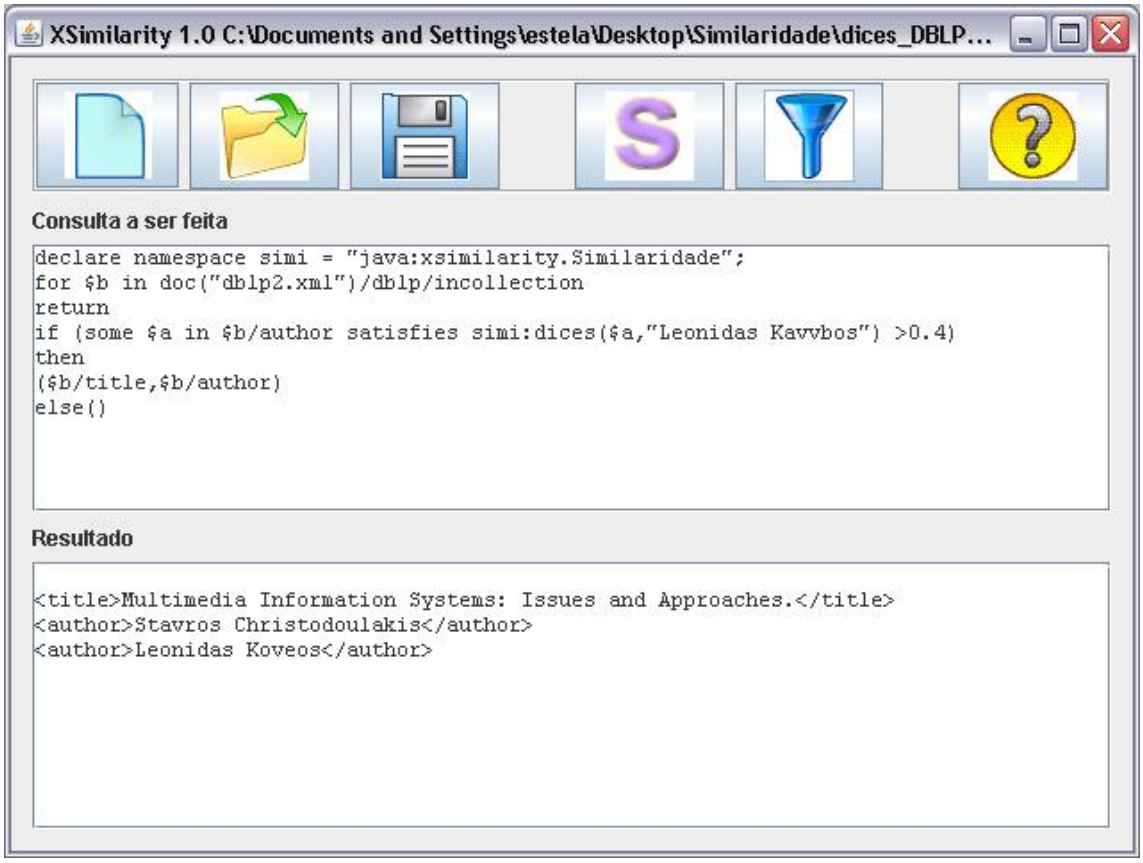

Figura 4.9: Dice's Similarity recupera resultado com similaridade menor

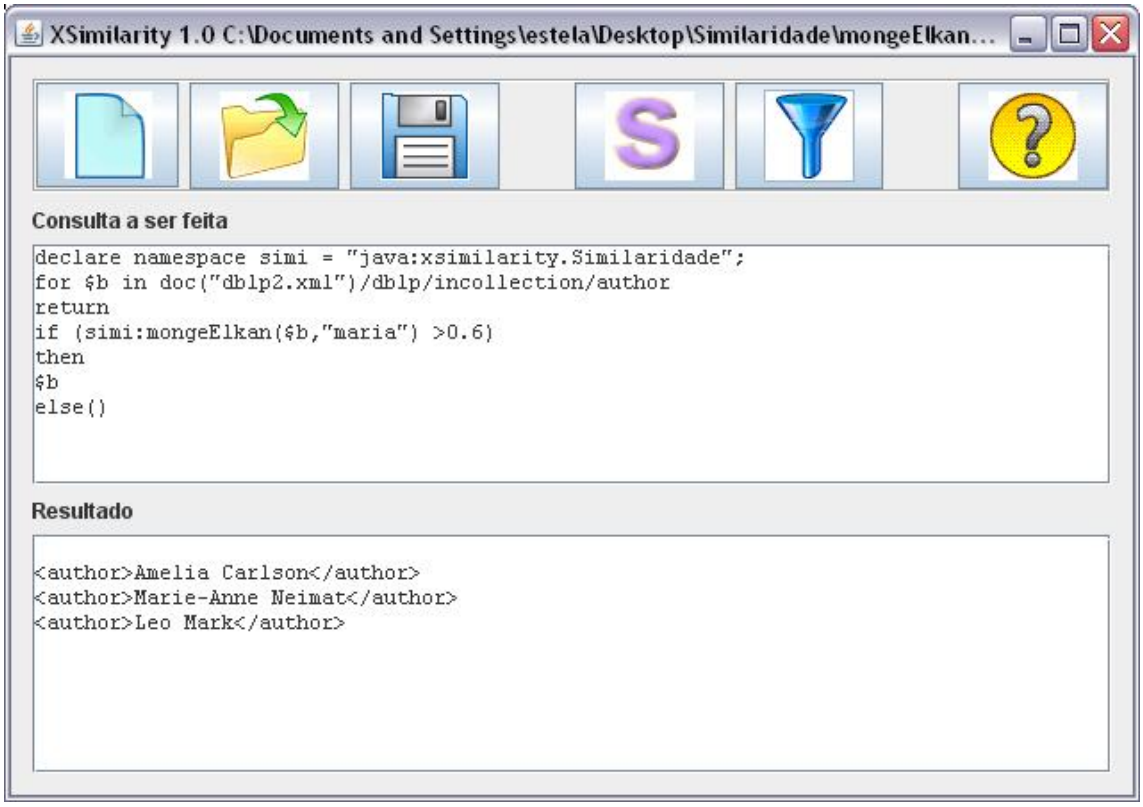

Figura 4.10: Monge Elkan recupera elementos não desejados

4.3 Considerações Finais

Este capítulo descreveu a XSimilarity, que é uma ferramenta para consultas por similaridade embutidas na linguagem XQuery. A ferramenta desenvolvida fornece uma alternativa simples e ágil para executar consultas por similaridade em bases de dados XML. Ela pode apresentar grande utilidade nesse contexto, facilitando a recuperação de informações em bases de dados XML. Outra vantagem da ferramenta é facilitar a comparação entre as funções de similaridade para diferentes contextos de utilização.

A ferramenta foi planejada de modo que possa ser facilmente estendida para a inclusão de novas funções de similaridade. Acredita-se que isto deva ocorrer futuramente, com a maior divulgação dessa ferramenta na comunidade científica.

5 CONCLUSÃO

Este capítulo apresenta as conclusões obtidas durante a realização deste trabalho e propõe trabalhos futuros. Algumas conclusões foram obtidas durante a implementação da ferramenta, e outras puderam ser observadas já na fase de estudos preliminar.

Quanto à linguagem de consulta XQuery, observa-se que é uma linguagem bastante flexível, pois pôde-se facilmente estendê-la para suportar funções de similaridade. Seu poder de expressão e simplicidade são muito úteis para realizar consultas sobre bases de dados XML.

Funções de similaridade são muito importantes para encontrar registros quando não se conhece precisamente a *string* a ser encontrada. Também se observou sua importância para detectar semelhanças ou diferenças em documentos maiores. Entretanto, funções de similaridade podem gerar resultados irrelevantes em contextos onde não são apropriadas.

O XSimilarity implementa funções de similaridade que atuam de formas variadas, sendo uma ferramenta muito útil para realizar buscas. O conjunto de funções de similaridade implementado envolve métricas baseadas em caracteres, palavras (*tokens*) ou híbridas. Isto torna a ferramenta bastante abrangente, permitindo maior flexibilidade para experimentos dos mais diversos tipos. O XSimilarity implementa algumas das principais funções de similaridade utilizadas nas áreas de bancos de dados e recuperação de informação.

Como trabalhos futuros sugere-se a implementação de novas funções de similaridade na ferramenta e a utilização dessa ferramenta para comparar as funções em termos de eficácia na recuperação de informações. Também se pode aprimorar a funcionalidade de edição de consultas da ferramenta, incluindo recursos como *Syntax Highlight* e a funcionalidade de desfazer. A implementação de mensagens de erro e da funcionalidade de ajuda da ferramenta também são melhorias possíveis.

6 REFERÊNCIAS

BERKELEY LAB **Nux.** Disponível em <http://dsd.lbl.gov/nux/>. Acesso em: nov. 2007

BORGES, E. N.; CONY, C. A. **Consultas por Similaridade em SGBDS Comerciais: Estendendo o PostgreSQL**. 2005. 90f. Projeto de Diplomação (Bacharelado em Engenharia da Computação) – FURG, Rio Grande

CATTELL, R. et al. **The Object Database Standard**: ODMG-93, Release 1.2. Morgan Kaufmann Publishers, San Francisco, 1996.

CHAMBERLIN, D. et al. **Quilt: an XML Query Language for Heterogeneous Data Sources**. In: Lecture Notes in Computer Science, Springer-Verlag, Dez. 2000. Disponível em: <http://www.almaden.ibm.com/cs/people/chamberlin/quilt_lncs.pdf>

CHAPMAN, S. **Sam's string metrics.** Disponível em: < http://www.dcs.shef.ac.uk/~sam/stringmetrics.html>. Acesso em: nov. 2007

Cohen, W., Ravikumar, P., and Fienberg S. (2003): A Comparison of String Distance Metrics for Name-Matching Tasks in *IIWeb 2003*: 73-78.

DEUTSCH, A. et al. **A Query Language for XML**. Disponível em: <http://www.research.att.com/~mff/files/final.html>

Dice, Lee R. (1945). Measures of the Amount of Ecologic Association between Species. Journal of Ecology, 26: 297-302.

GROFF, J.R; WEINBERG,P.N. **SQL:** the complete reference. Berkeley,CA: Osborne/ McGrawHill, 2002.

HAMMING, R. W. Error detecting and error correcting codes. In: Bell Syst. Tech. J., 29:147–160, 1950.

ISO. **Information Technology-Database Language SQL**. Standard No. ISO/IEC 9075:1999. (Available from American National Standards Institute, New York, NY 10036, (212) 642-4900.)

Jagadish, H. V.; Mendelzon, A. O.; Milo, T. Similarity-Based Queries. Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of

Database Systems, May 22-25, 1995, San Jose, California. ACM Press. pages: 36-45.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association* 84:414–420.

Laender, A.; Gonçalves, M.; Roberto, P. BDBComp: Building a Digital Library for the Brazilian Computer Science Community. In: Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries, Tuscon, AZ, USA, pp. 23-24, 2004.

Levenshtein, I. V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory*, 10(8):707–710.

Monge, A. E., and Elkan, C. (1996). The Field Matching Problem: Algorithms and Applications. *KDD 1996*: 267-270.

ROBIE, J. et al. **XML Query Language (XQL)**.Disponível em: <http://www.w3.org/TandS/QL/QL98/pp/xql.html>

University of Trier. Digital Bibliography & Library Project (DBLP). Disponível em <http://dblp.uni-trier.de/>. Acesso: novembro, 2007.

Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research Methods*, American Statistical Association. 354-359.

W3C. **XML Path Language (XPath) Version 1.0**. W3C Recommendation, Nov. 1999. Disponível em: <http://www.w3.org/TR/xpath.html>

W3C. **Extensible Markup Language (XML) 1.0 (Fourth Edition)**. Disponível em: <http://www.w3.org/TR/2006/REC-xml-20060816/>. Acesso em: setembro 2006.

W3C WORKING GROUP. **XML Query (XQuery)** Draft. World Wide Web Consortium (W3C). Nov. 2005. Disponível em <http://www.w3.org/XML/Query/>. Acesso em: Novembro 2006.

APÊNDICE A ARQUITETURA DO XSIMILARITY

Este apêndice apresenta a arquitetura da ferramenta desenvolvida, o XSimilarity. São apresentados o Diagrama de Casos de Uso e o Diagrama de classes. Para os casos de uso mais relevantes, são apresentados uma descrição e um diagrama de seqüência.

A.1 Diagrama de Casos de Uso

O Diagrama de Casos de Uso apresentado a seguir ilustra as funcionalidades da ferramenta desenvolvida. A funcionalidade de Inserir função de Similaridade é uma extensão de Editar Consulta que facilita a utilização das funções de similaridade desenvolvidas. Os casos de uso são auto-explicativos.

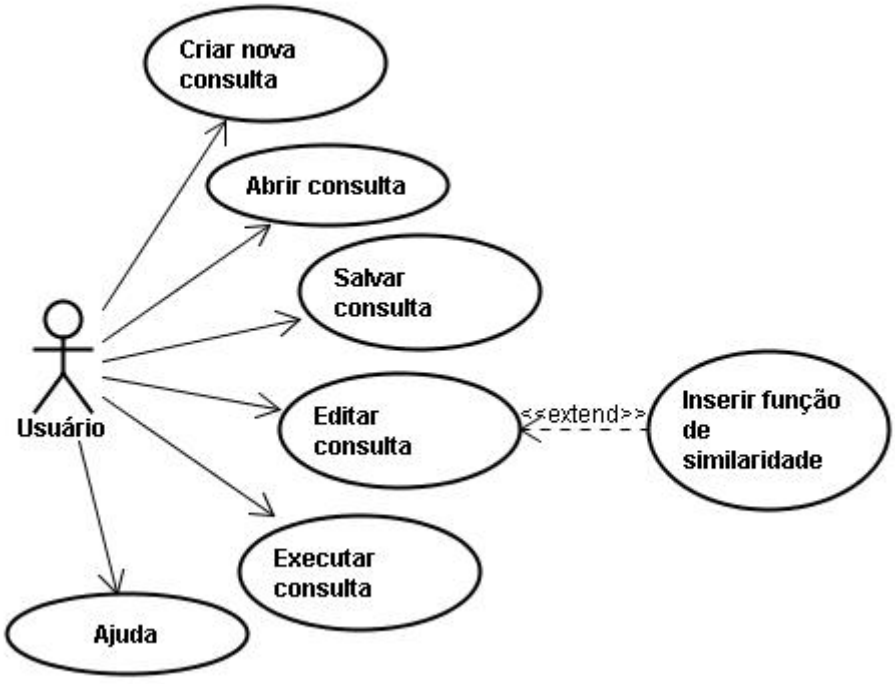

Figura A.1: Diagrama de Casos de Uso

A.2 Diagrama de Classes

A Figura A.2 apresenta o Diagrama de Classes da ferramenta desenvolvida. As classes TelaPrincipal, TelaConsultas, Constantes, Similaridade, FiltroXQuery e Main são próprias da ferramenta, sendo as demais classes pertencentes a bibiliotecas externas.

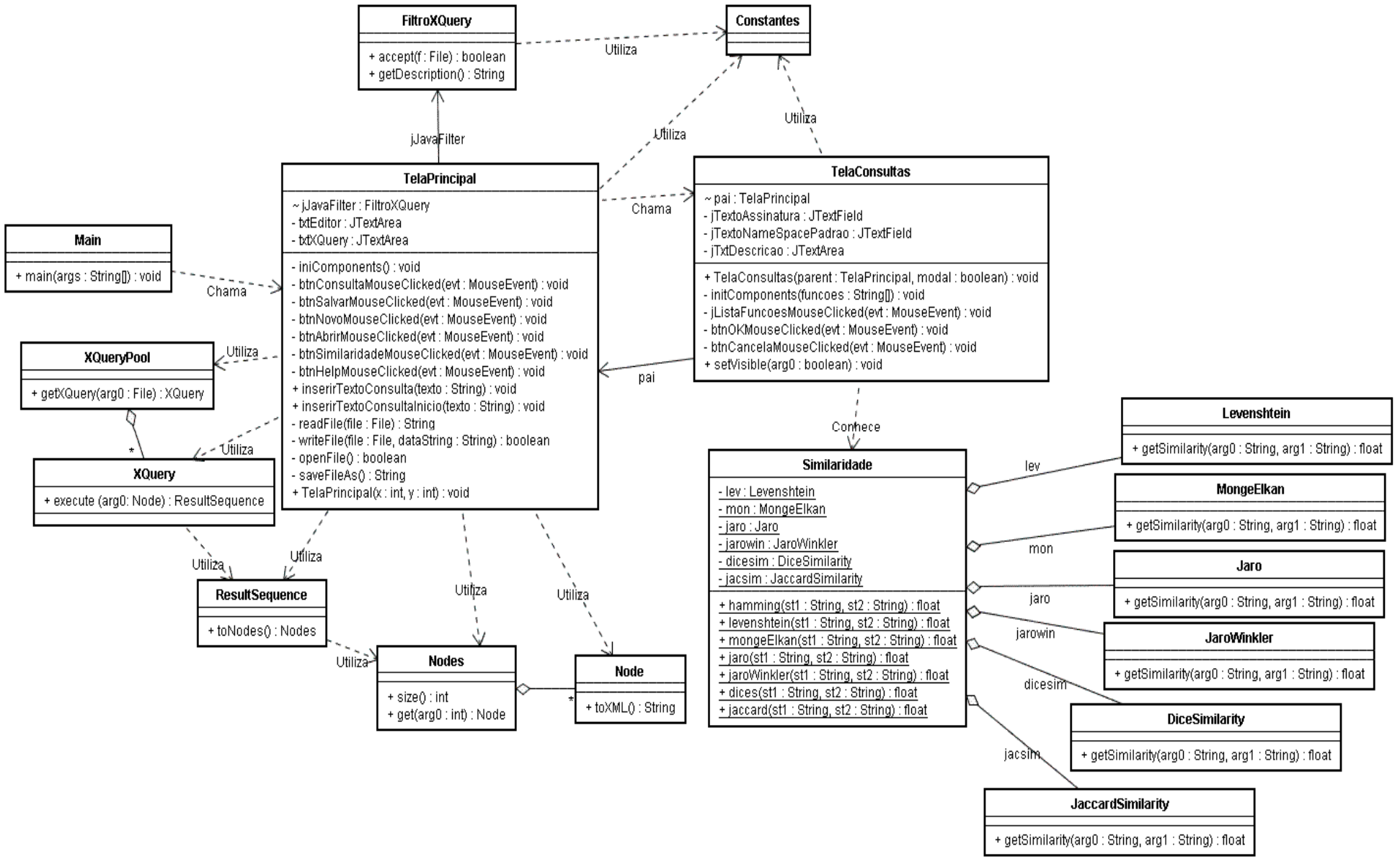

Figura A.2: Diagrama de classes

A classe Main serve para inicializar o programa. Ela instancia a TelaPrincipal e passa o controle para a mesma. A classe TelaPrincipal implementa a janela principal do programa, e apresenta também diversos métodos com a finalidade de manipular arquivos de consulta XQuery e tratar interações do usuário. Também é a partir da TelaPrincipal que é chamada a classe TelaConsultas, a qual é a responsável pela inserção das funções de similaridade no texto da consulta.

A classe FiltroXQuery implementa um filtro que determina quais arquivos serão exibidos na funcionalidade de abrir consultas com base em suas extensões. A classe Constantes possui uma série de constantes que são usadas nesse programa. Assim, pode-se alterar uma constante do programa diretamente nessa classe, facilitando a sua localização.

A classe Similaridade pode ser considerada uma das mais importantes dessa ferramenta. Ela é a responsável pela implementação do *namespace* de funções de similaridade. Nessa classe são oferecidas funções de similaridade estáticas que permitem a execução, pela classe XQuery, dessas funções de similaridade. A maior parte das funções de similaridade utilizadas nessa classe foram implementadas pela bibilioteca *Similarity Metrics* (CHAPMAN, s.d.). A biblioteca NUX (BERKELEY LAB, s.d.) foi usada para a execução das consultas XQuery.

A.3 Casos de uso

Essa seção descreve os três casos de uso principais: Editar consulta, Inserir função de similaridade e Executar consulta.

A.3.1 Editar consulta

O caso de uso inicia quando o usuário coloca em foco a caixa de texto de consulta na tela principal.

Cenário Principal

1. O usuário efetua suas modificações digitando com o teclado
2. O sistema apresenta a consulta modificada ao usuário na caixa de texto de consulta na tela principal
3. O caso de uso termina

Cenário Alternativo 1 – Inserir função de similaridade

Alternativamente ao passo 1 do cenário principal

1. O usuário clica no botão inserir função de similaridade
2. O sistema executa o caso de uso “Inserir função de similaridade”
3. O caso de uso termina

A.3.2 Inserir função de similaridade

O caso de uso inicia quando o usuário clica no botão inserir função de similaridade

Cenário Principal

1. O sistema apresenta uma caixa de diálogo com funções de similaridade
2. O usuário escolhe uma função de similaridade na lista de funções
3. O sistema apresenta, em caixas de texto distintas, a assinatura, o *namespace* padrão e a descrição da função selecionada
4. O usuário clica no botão OK
5. O sistema insere a função de similaridade na caixa de texto de consulta na tela principal na posição do cursor

6. O sistema esconde a caixa de diálogo com funções de similaridade
7. O caso de uso termina

Cenário Alternativo 1 – Editar função de similaridade

Alternativamente ao passo 4 do cenário principal

1. O usuário edita a assinatura da função de similaridade escolhida
2. Retorna ao passo 4 do cenário principal

Cenário Alternativo 2 – Inserir *namespace* padrão

Alternativamente ao passo 4 do cenário principal

1. O usuário marca a opção de inserir *namespace* padrão
2. O usuário clica no botão OK
3. O sistema insere a função de similaridade na caixa de texto de consulta na tela principal na posição do cursor
4. O sistema insere o *namespace* padrão na primeira linha da caixa de texto de consulta na tela principal
5. Retorna ao passo 6 do cenário principal

Cenário Alternativo 3 – Cancelar a inserção

Alternativamente aos passos 2 ou 4 do cenário principal

1. O usuário clica no botão "Cancelar"
2. Retorna ao passo 6 do cenário principal

A.3.3 Executar consulta

O caso de uso inicia quando o usuário clica no botão executar consulta

Cenário principal

1. O sistema executa a consulta que está na caixa de texto de consulta na tela principal
2. O sistema apresenta o resultado da consulta na caixa de texto de resultado na tela principal
3. O caso de uso termina

Cenário Alternativo 1 – A consulta é inválida

Alternativamente ao passo 1 do cenário principal

1. O sistema limpa a caixa de texto de resultado na tela principal
2. O caso de uso termina

A.4 Diagramas de Seqüência

A Figura A.3 representa o Diagrama de Seqüência do cenário principal do caso de uso Inserir função de similaridade. A Figura A.4 representa o Diagrama de Seqüência do cenário principal do caso de uso Executar consulta.

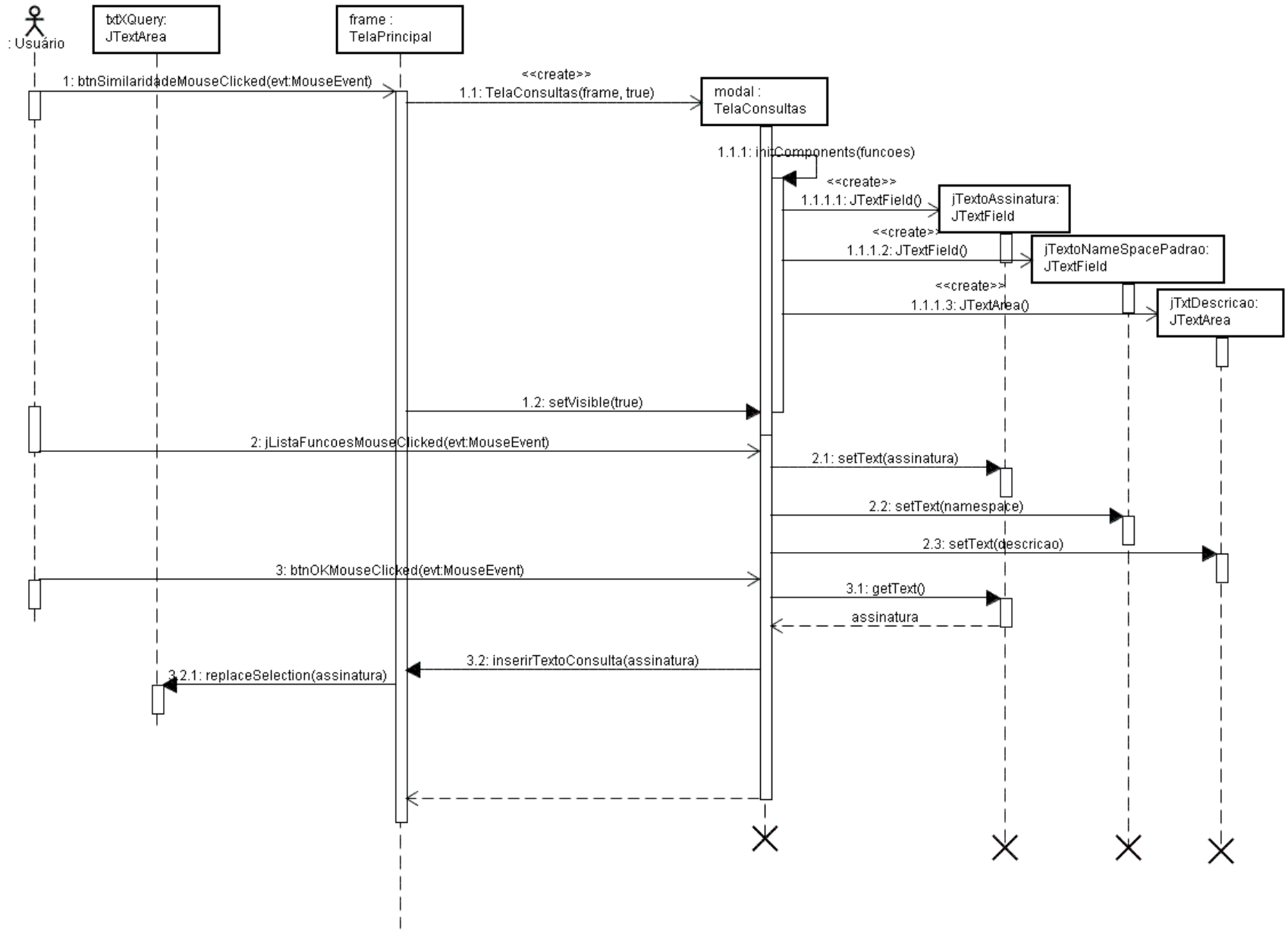

Figura A.3: Diagrama de seqüência do cenário principal do caso de uso Inserir função de similaridade

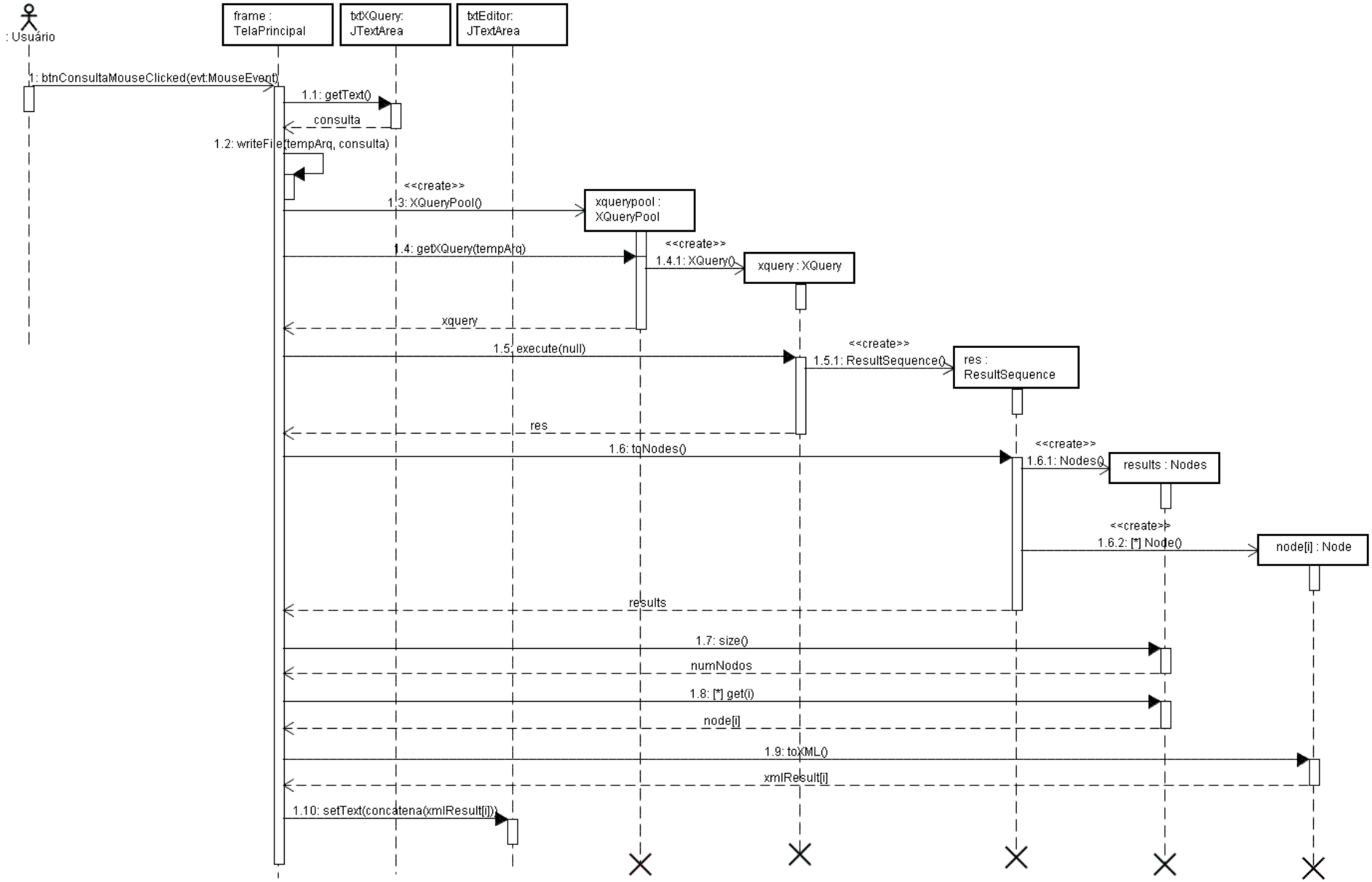

Figura A.4: Diagrama de seqüência do cenário principal caso de uso Executar consulta