LEONARDO ERWIN WOJCIKIEWICZ

FERRAMENTA DE BUSCA NA WEB COM FEEDBACK EXPLÍCITO

FLORIANÓPOLIS

Novembro de 2012

LEONARDO ERWIN WOJCIKIEWICZ

FERRAMENTA DE BUSCA NA WEB COM FEEDBACK EXPLÍCITO

Trabalho de conclusão de curso submetido à banca examinadora do curso de Sistemas de Informação da Universidade Federal de Santa Catarina para obtenção do título de bacharel em sistemas de informação.

Orientadora:

Prof[a]. Dr[a]. Carina Friedrich Dorneles

UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TÉCNOLOGICO

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

SISTEMAS DE INFORMAÇÃO

FLORIANÓPOLIS

Novembro de 2012

LEONARDO ERWIN WOJCIKIEWICZ

FERRAMENTA DE BUSCA NA WEB COM FEEDBACK EXPLÍCITO

Trabalho de conclusão de curso apresentado como requisito para obtenção do título de Bacharel em Sistemas de Informação.

Orientadora:

_____

Prof[a]. Dr[a]. Carina Friedrich Dorneles

Banca examinadora:

_____

Prof. Dr. Ronaldo dos Santos Mello

_____

MSc. Rodrigo Gonçalves

FLORIANÓPOLIS

Novembro de 2012

Sumário

1. Introdução.....09

2. Fundamentação teórica.....12

2.1 Recuperação da informação.....12

2.2 Sistemas de recuperação de informação.....12

2.3 Busca por similaridade.....13

2.4 *Feedback* de relevância.....13

2.5 Método de Rocchio.....14

3. Trabalhos relacionados.....17

3.1 Comparação de alternativas para incorporação do *feedback*.....17

3.2 Aprendendo a partir de *feedback* implícito.....19

3.3 Usando cliques como *feedback* implícito.....21

3.4 Comparando técnicas de *feedback* explícito e implícito para recuperação na *Web*.....23

3.5 Tabela comparativa.....24

4. Ferramenta.....26

4.1 Visão geral.....26

4.2 Arquitetura.....27

4.3 Implementação.....29

4.4 Funcionamento da ferramenta.....31

5. Experimentos.....33

6. Conclusão e trabalhos futuros.....37

7. Referencias bibliográficas.....39

Lista de Figuras

Figura 1 - Método Rocchio e variações.....15

Figura 2 – Visão geral.....27

Figura 3 – Arquitetura do XFind.....28

Figura 4 – Diagrama de classes.....30

Figura 5 – Campo de busca.....31

Figura 6 – Interface de resultados.....31

Figura 7 - Pagina de resultados com a consulta reescrita.....32

Figura 8 - Gráfico com pesquisa com nome de filmes.....34

Figura 9 - Gráfico com pesquisa com nome de ruas.....34

Figura 10 - Gráfico com pesquisa com nome de instituições.....35

Lista de Tabelas

Tabela 1 - Exemplo da métrica q-grams.....13

Tabela 2 - Comparação entre trabalhos relacionados.....24

Tabela 3 - Configuração de testes.....33

Tabela 4 - Resultado dos testes.....35

Abreviaturas

XML - *Extensible Markup Language*

API - *Application Programming Interface*

SRI - Sistema de recuperação de informação

Resumo

Com o crescente número de páginas e dados na *Web*, a dificuldade de localizar determinados documentos aumentou. Os sistemas de busca mais utilizados atualmente fazem as buscas através de palavras-chave digitadas pelos usuários. No topo dos resultados aparecem as páginas que mais foram acessadas pelos usuários, contendo os termos com maior similaridade comparada com as palavras chaves utilizadas pelo usuário como consulta. Porém, nem sempre os primeiros resultados são aqueles que são desejados pelo usuário. Para tornar o resultado mais preciso, é possível utilizar o *feedback* do usuário, que pode ser implícito ou explícito. A forma de *feedback* implícito não consegue prever todas as vezes o que realmente o usuário deseja, pois ele se baseia em ações que não demonstram explicitamente a escrita da consulta. Uma das formas de fazer isto é quando o usuário precisa trocar as palavras chaves até que os resultados reflitam o que ele deseja. A ideia deste trabalho é utilizar o *feedback* explícito, para que o usuário não precise alterar palavras-chave para refazer sua busca. Ele precisa escolher quais documentos são os mais relevantes e assim o sistema se encarregará de refazer a busca com base nos termos encontrados nestes documentos.

Palavras-chave: Ferramenta de busca, recuperação de informação, *feedback* explícito, xml.

1. Introdução

Com a popularização da *Web*, o número de páginas na Internet cresceu consideravelmente. Desde meados dos anos 90, quando surgiram os primeiros mecanismos de busca, há a preocupação de otimizar os resultados da busca destas ferramentas.

As ferramentas de buscas mais utilizadas, como o *Google*, *Yahoo* e *Bing*, utilizam *webcrawlers,* que são robôs que percorrem toda a web para localizar páginas e depois indexá-las. Quando um usuário faz alguma consulta nestas ferramentas o motor de busca exibe o *link* das páginas ou arquivos que tem alguma relação com a palavra-chave utilizada pelo usuário.

A ordenação destes resultados depende de qual motor de busca é utilizado e pode ser feita de várias formas tais como: localização e quantidade de vezes que o termo procurado aparece em uma página, proximidade entre os termos quando se encontram na mesma página, a densidade do documento (número de vezes que um termo aparece dependendo do tamanho do documento), popularidade do link, links promovidos entre outros menos utilizados (CENDON, 2001).

Além de todos estes critérios de ordenação, os resultados podem ser melhorados quando se utiliza o *feedback* do usuário. Há duas formas de se obter o *feedback* do usuário: explicitamente, quando o usuário escolhe quais documentos são relevantes para sua pesquisa; e implicitamente, quando a ferramenta de busca utiliza dados de utilização dos usuários para indicar quais documentos são relevantes (JUNG, HERLOCKER, WEBSTER 2006).

As duas formas possuem suas vantagens e desvantagens. Com o *feedback* explicito, o grau de precisão é maior que a forma implícita porém, é necessário que o usuário forneça à ferramenta quais os documentos são importantes para sua pesquisa. O *feedback* implícito, apesar de ser menos preciso, consegue fazer toda ordenação sem nenhuma interferência do usuário.

O objetivo principal deste trabalho é o desenvolvimento de uma ferramenta *Web* onde o resultado da busca possa ser melhorado a partir do *feedback* explicito do usuário. A consulta do usuário deve ser reformulada de acordo com os documentos escolhidos como relevantes.

Para limitar o escopo da aplicação, apenas arquivos XML serão indexados. O XML é uma recomendação da W3C (*World Wide Web Consortium*) para linguagens de marcação e por este motivo possibilita a interoperabilidade entre diferentes linguagens.

Os arquivos são lidos pela ferramenta e os valores dos elementos são indexados para comparação com a consulta do usuário. A ordenação inicial é feita com base na similaridade entre a consulta do usuário e os termos do documento. Quanto maior a similaridade melhor é a posição do documento na lista dos resultados. A similaridade entre os termos e as palavras-chave é medida através da métrica q-grams. O principal motivo da escolha desta métrica é que a variação do grau de similaridade causada pela disposição dos termos é menor do que em outras métricas. Cada comparação tem uma pontuação que varia de zero, onde não há nenhuma semelhança, e um, quando os termos comparados são iguais. Portanto, quanto mais próximo de um, maior é a semelhança entre os termos (DORNELES, GALANTE, 2008).

O trabalho está estruturado da seguinte forma: no Capítulo 2 é tratada a fundamentação teórica utilizada para o desenvolvimento da ferramenta; no capítulo 3 são apresentados os trabalhos relacionados, que trata sobre aplicações na mesma área de pesquisa; no Capítulo 4 é descrita a ferramenta, que trata sobre a implementação e execução da ferramenta criada; no capítulo 5 são demonstrados os resultados dos experimentos realizados e no Capítulo 6, são apresentadas as conclusões e os trabalhos futuros.

2. Fundamentação teórica

Neste capítulo são abordados conceitos que são agregados ao desenvolvimento e execução da ferramenta.

2.1 Recuperação da informação

A recuperação da informação lida com a representação, armazenamento, organização e acesso aos itens da informação [YATES e NETO 1999]. Tanto a representação quanto organização destes itens deveria facilitar o acesso do usuário à informação de seu interesse. Porém, o usuário não consegue buscar diretamente a informação necessária. Ele precisa traduzir a necessidade da informação em uma consulta para ser processada por uma ferramenta de busca.

2.2 Sistemas de recuperação de informação

Sistemas de recuperação de informação são utilizados para recuperar dados, geralmente na forma de textos. Quando um usuário especifica sua consulta, o sistema processa a recuperação de acordo com os documentos indexados, a estrutura que estes documentos estão inseridos e a própria consulta do usuário e então, uma lista com os documentos recuperados é criada. A ordenação da lista é baseada em um número que é obtido comparando a consulta com os documentos indexados (CARDOSO, 2009).

Quanto maior o número, maior é a similaridade e mais bem ranqueado este documento será.

2.3 Busca por similaridade

Nem todos os documentos que são relevantes para o usuário são recuperados em uma consulta quando se utiliza uma busca por termos semelhantes. A partir disto, surge a necessidade de se fazer a busca através da similaridade. Isto ocorre porque diferentes palavras podem possuir o mesmo significado ou até mesmo pequenos erros gramaticais no documento indexado ou na consulta do usuário podem ocorrer. Utilizando a busca por similaridade, estes documentos podem ser recuperados.

Uma das métricas utilizadas para buscas em textos é o q-grams (UKKONEN, 1992). Ele separa uma palavra em grãos de tamanho q e posteriormente faz uma procura destes grãos no texto.

Abaixo, a tabela um exemplifica o funcionamento da métrica q-grams. O asterístico (*) representa um espaço em branco.

*un	uni	niv	ive	ver	ers	rsi	sid	ida	dad	ade	de*

Tabela 1. Exemplo do q-grams utilizando q=3 e a palavra universidade.

2.4 *Feedback* de relevância

F*eedback* de relevância é um processo de recuperação de informação onde o usuário do motor de busca indica quais documentos são relevantes ou

parecidos aquele que ela está procurando [JUNG, HERLOCKER, WEBSTER 2006]. O efeito esperado é que a nova consulta se aproxime dos documentos relevantes e se afaste dos documentos irrelevantes.

O *feedback* de relevância se divide em dois tipos: *feedback* implícito, onde o sistema coleta informações sem interferência do usuário, e *feedback* explícito, onde o usuário repassa ao sistema quais são os documentos relevantes.

O conceito do *feedback* implícito é baseado na ideia que os usuários continuamente fazem julgamentos de valor enquanto procuram por informação. Pesquisadores propuseram varias formas de *feedback* implícito, tais como cliques para selecionar documentos, *scroll* na página, adição aos favoritos, impressão de página e o tempo gasto em uma página [JUNG, HERLOCKER, WEBSTER 2006].

Com o *feedback* explícito, o usuário indica quais documentos são relevantes e quais não são relevantes, porém é difícil conseguir uma quantidade suficiente de documentos e os usuários nem sempre estão dispostos a fazer um trabalho adicional para prover o feedback [JUNG, HERLOCKER, WEBSTER 2006].

2.7 Método de Rocchio

O método de Rocchio (ROCCHIO, 1971) é uma técnica para expansão de consulta. Ele indica quais termos dos documentos relevantes e irrelevantes escolhidos pelo usuário devem ser adicionados ou removidos à nova consulta.

O número de termos a ser considerado na consulta depende dos valores de α, β, γ.

O método de Rocchio deu origem a mais duas fórmulas semelhantes para expandir a consulta, chamadas de IDE e IDE_DEC_HI [YATES e NETO 1999].

$$Standard\_Rochio: \quad \vec{q}_m = \alpha \vec{q} + \frac{\beta}{|D_r|} \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

$$Ide\_Regular: \quad \vec{q}_m = \alpha \vec{q} + \beta \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

$$Ide\_Dec\_Hi: \quad \vec{q}_m = \alpha \vec{q} + \beta \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \; max_{non\text{-}relevant}(\vec{d}_j)$$

Figura 1. Método de Rocchio e suas variações

Onde α, β, γ são os pesos que cada termo deve receber, q é a consulta inicial, D_r é o conjunto de documentos relevantes, D_n é o conjunto de documentos não relevantes e $max_{non\text{-}relevant}$ refere-se ao documento não relevante mais bem ranqueado.

O método padrão faz uma média de acordo com o peso para cada termo em cada tipo de documento relevante ou irrelevante. Se a média de um termo for maior para o documento relevante, este termo será incluso na nova consulta. A consulta inicial sempre é adicionada à nova consulta.

O método IDE soma a quantidade de vezes que cada termo foi incluído em um documento relevante e em um documento irrelevante. Se a soma do peso de um determinado termo é maior para documentos relevantes do que a soma do peso deste mesmo termo para documentos irrelevantes, ele é

adicionado à consulta. Os termos da consulta inicial são adicionados à lista dos documentos relevantes.

O método MAX incluí a consulta inicial e o termos marcados como relevantes menos os termos do documento selecionado como irrelevante com maior similaridade.

Para exemplificar o uso do método rocchio, vamos supor que uma busca pela palavra universidade e o valor de α, β e γ sejam um. Os seguintes documentos foram recuperados.

- Universidade Federal Santa Catarina
- Universidade Estadual Santa Catarina
- Universidade Federal Santa Maria

O usuário marca os dois primeiros resultados como relevante e o último como irrelevante e então obtém a nova consulta.

Utilizando o método padrão de Rocchio a nova consulta será Universidade Estadual Catarina, pois a média dos pesos destes termos que foram selecionados como relevante é maior do que aqueles que foram selecionados como irrelevante.

Utilizando o método IDE a nova consulta será Universidade Estadual Santa Catarina, pois a soma dos pesos destes termos que foram selecionados como relevante é maior do que aqueles que foram selecionados como irrelevante.

Utilizando o método MAX a nova consulta será Estadual Catarina porque estes dois termos foram selecionados como relevantes enquanto os outros estavam contidos no documento irrelevante com maior similaridade comparado a consulta inicial.

3. Trabalhos relacionados

Neste capítulo são descritas alternativas de como coletar e utilizar o *feedback* do usuário.

3.1 Comparação de alternativas para incorporação do *feedback*.

O objetivo deste trabalho (AGICHTEIN, BRILL, DUMAIS, 2006) é mostrar que dados referentes ao comportamento do usuário podem melhorar a ordenação dos melhores resultados em uma busca. São examinadas alternativas para incorporação do *feedback* no processo de ordenação e exploradas as contribuições do *feedback* do usuário comparado a outros métodos de busca comuns.

A principal motivação é entender como o *feedback* implícito pode ser utilizado em longa escala em um ambiente operacional para melhorar a recuperação. O trabalho considerada duas abordagens para ordenação com *feedback* implícito: Tratamento do *feedback* implícito como uma evidência independente para os resultados do ranking, cujo objetivo é reordenar os resultados obtidos de acordo com os cliques do usuários e outras interações suas em sessões anteriores; e integração do *feedback* implícito integrado ao algoritmo de busca, onde os motores de busca modernos ordenam seus resultados com base em um grande número de recursos tais como o conteúdo da página e a qualidade da página.

O objetivo geral do experimento é interpretar corretamente o *feedback*

do usuário obtido traçando as interações do usuário com o motor de busca. A ideia é representar as ações do usuário para cada resultado da busca como um vetor de recursos, e então treinar o algoritmo de ordenação neste vetor para obter valores indicativos de resultados de busca relevantes. O modelo de comportamento de busca na Web observado é a combinação de um componente em "*background*", e um componente de "relevância".

Para aprender a interpretar o comportamento do usuário observado, são relacionadas as ações do usuário com os seus julgamentos para um conjunto de consultas de treino. Encontram-se todas as instâncias das buscas nos *logs* de sessão onde essas consultas foram submetidas ao motor de busca, e agregam-se as características de comportamento do usuário para todas as sessões de busca envolvendo essas consultas.

Vários algoritmos são utilizados para quantificar a efetividade do *feedback* implícito. Eles comparam a eficácia do comportamento implícito do usuário com base no conteúdo correspondente, características de qualidade das páginas estáticas e a combinações de todos os recursos.

O trabalho concluí que o *feedback* implícito é melhor utilizado quando o resultado original de uma consulta é ruim. Uma direção promissora para futuros trabalhos é a aplicação de pesquisas recentes para prever automaticamente a dificuldade da consulta, e apenas incorporar o *feedback* implícito para as consultas mais difíceis. Outro rumo de pesquisa é saber como extender os métodos de *feedback* para consultas inéditas, o que deve melhorar ainda mais a experiência de pesquisa na web dos usuários.

3.2 Aprendendo a partir de *feedback* implícito

O trabalho de (JOACHIMS, RADLINSKI, 2007) fala sobre motores de busca e *feedback* implícito. Os autores afirmam que os logs dos motores de busca contêm informações em que técnicas de *machine-learning* podem ser usadas para melhorar a qualidade dos resultados de uma busca.

Cada vez que um usuário faz uma busca ou clica em algum resultado, esta informação é repassada ao motor de busca. Os grandes motores de busca da internet hoje gravam cliques e consultas utilizadas em suas buscas. Porém, enquanto o *feedback* implícito parece ser intuitivo de ser utilizar, ainda não está claro como estas informações podem ser colocadas de forma operacional.

Infelizmente, o clique do usuário em um resultado não indica necessariamente que este é o que ele estava procurando. Muitas vezes, o usuário olha apenas os três primeiros resultados de uma busca, deixando um resultado relevante despercebido. Com isso conclui-se que os usuários avaliam apenas poucos resultados antes do clique, ou seja, apenas os primeiros *links*. Quando os dois primeiros resultados de uma busca são invertidos, aquele que era o segundo resultado do *ranking* passa a receber muito mais cliques do que o primeiro.

Mais informativo do que saber em qual *link* o usuário clicou é saber em qual ele não clicou. Isto é chamado de preferencia relativa. Isto opõe a tendência do usuário de clicar apenas nos links melhores ranqueados onde a ideia de que não clicar em um resultado no topo do ranking seria um erro. A ideia é avaliar porque um usuário escolheu um resultado em detrimento de outro.

É possível realizar experimentos de forma a eliminar ruídos para melhorar os resultados. Para exemplificar isto, considera-se o problema de qual motor de busca o usuário prefere: Google ou Yahoo. Qualquer busca que o usuário fizer retornará resultados de ambos os sistemas em uma única página onde o usuário não saiba qual é a fonte deles. Um usuário sem preferência por nenhum motor de busca tem a mesma chance de escolher um resultado de ambas as páginas. Se ele escolhe sempre um de um mesmo sistema, concluímos que ele prefere o resultado deste mesmo motor de busca.

Experimentos também podem ser feitos para melhorar o valor do *feedback*. Como o usuário sempre escolhe os resultados do topo, resultados mal ranqueados não oferecem nenhum *feedback*. Colocar estes resultados no topo do *ranking* pode não ser ideal no início, mas pode acelerar o aprendizado do motor, pois como o usuário não irá clicar neles, o aprendizado ocorre mais rapidamente.

Além dos cliques em resultados, motores de busca também podem obter *feedback* a partir do tempo de leitura antes de um clique em algum resultado. Se um resultado é escolhido rapidamente após outro já ter sido escolhido significa que o primeiro não tinha relevância com o que o usuário procurava. Outro interessante fator é o abandono da busca, que é quando o usuário resolve não clicar em nenhum dos resultados exibidos. Por fim, temos a chamada “*query chain*”, ou cadeia de consultas. Isso ocorre quando usuários alteram os termos de uma consulta. Isto é útil para saber como o usuário formula suas necessidades.

3.3 Usando cliques como *feedback* implícito

Os dados dos cliques podem ser considerados uma forma de *feedback* de relevância, segundo (JUNG, HERLOCKER, WEBSTER, 2006). A ideia deste trabalho é descobrir como os dados de pesquisa de um determinado usuário podem auxiliar outros usuários do sistema.

A obtenção dos dados dos cliques são uma forma interessante de *feedback* implícito por várias razões: são fáceis de coletar e são mais valiosas comparadas a outras formas de *feedback* implícito. No trabalho, são estudadas três questões: como os cliques podem melhorar a precisão do motor de busca; o que mais pode ser coletados através dos cliques e; qual o valor dos dados do clique para as mudanças do *feedback* de relevância quando o objetivo é procurar um documento para o usuário que satisfaça completamente suas necessidades (documentos estritamente relevantes).

Três áreas de pesquisas nesta área são interessantes: comportamento implícito do usuário na recuperação como fator de relevância (grande quantidade de informações que podem ser adquiridas); dados dos cliques do usuário como uma evidência para julgar o sucesso da busca (assume-se que documentos clicados são mais relevantes que os que não foram); e sistemas de filtros colaborativos (comunidade trabalha para separar informações interessantes e não interessantes).

O SERF (*System for Electronic Recommendation Filtering*), que é uma ferramenta de busca da Universidade do Estado de Oregon (OSU), é um protótipo de um sistema de busca de documentos que aplica técnica de filtro

colaborativo, em que suas capacidades de busca são melhoradas ao observar as sessões de cada usuário, consultas e consequente navegação.

Estudos anteriores examinaram como os cliques do usuário no resultado da busca poderiam ser utilizados para passar o *feedback* para o motor de busca. Utilizar os cliques em toda a pesquisa irá aumentar o número de dados do *feedback*, e com este aumento de dados, mais documentos irrelevantes deveriam ser marcados como relevantes, porém, proporcionalmente foi o número de documentos relevantes que cresceu.

Utilizando o conceito de *learning machine*, foi criada a hipótese que os últimos documentos visitados em uma sessão de busca pode ser um subconjunto mais preciso de dados sobre relevância. Os testes comprovaram a hipótese mostrando uma maior precisão quando se utiliza este conceito. O ponto negativo é que menos resultados serão recuperados.

O resultado geral de testes mostrou que a forma mais precisa de *feedback* implícito é quando se utiliza o conjunto com os últimos documentos visitados em uma sessão de busca. Por fim, alguns problemas precisam ser resolvidos para melhorar a qualidade dos resultados. O primeiro deles é que o usuário forneça um *feedback* correto que forneça os melhores resultados. Outro problema é como agregar consultas semelhantes cujas informações que os usuários estão procurando sejam diferentes. Ainda há o problema das consultas cuja janela de tempo é limitada, como por exemplo, quem é o atual campeão de futebol, onde é necessário manter o *feedback* sempre atualizado.

3.4 Comparando técnicas de *feedback* explícito e implícito para recuperação na *Web.*

No trabalho de (WHITE, JOSE, RUTHVEN, 2010), os autores examinaram como o *feedback* implícito poderia substituir a técnica de *feedback* explícito. Para fazer esta comparação, foi desenvolvido um sistema onde era possível comparar as diferentes técnicas.

O sistema consiste de duas interfaces: Uma que utiliza o *feedback* explícito onde cada resultado possuía um *checkbox* para que o usuário marcasse caso o documento era relevante. A interface para o *feedback* implícito assumia que todos os resultados eram de interesse do usuário.

Foram feitos três diferentes testes para comparar as técnicas: Números de páginas de resultados visualizados; número de buscas completadas; e quanto tempo se levava para completar a busca.

A análise do *log* de resultados mostrou que a diferença entre as diferentes técnicas eram insignificantes e que nenhum sistema era mais eficiente que outro.

3.5 Tabela comparativa

A tabela a seguir mostra as diferenças entre os cada um dos trabalhos relacionados e a ferramenta desenvolvida neste trabalho (XFind).

Características	(AGICHTEIN, BRILL, DUMAIS, 2006).	(JOACHIMS, RADLINSKI, 2007).	(JUNG, HERLOCKER, WEBSTER, 2006).	(WHITE, JOSE, RUTHVEN, 2010)	XFind
Tipo de *feedback*	Implícito	Implícito	Implícito	Implícito e explícito	Explícito
Método de coleta	Comportamento do usuário.	Cliques, tempo do usuário em uma página.	Cliques durante a sessão de busca	Cliques e escolha do usuário	Escolha do usuário
Recurso para ordenação de resultados	Dados referentes ao comportamento do usuário na página de resultados	Consulta ao *log* da ferramenta	Dados de pesquisas de outros usuários	Usuário seleciona apenas resultados relevantes	Usuário seleciona resultados relevantes e irrelevantes
Dados considerados	HTML	HTML	HTML	HTML	XML
Interface	Web	Web	Web	Web	Web
Reformulação da pesquisa	Usuário precisa refazer a pesquisa com novos termos	Usuário precisa refazer a pesquisa com novos termos	Usuário precisa refazer a pesquisa com novos termos	Usuário refaz pesquisa com novos termos ou é feita de forma automática	Feita de forma automática
Capacidade de aprendizado de máquina	Sim	Sim	Sim	Sim	Não

Tabela 2. Comparação entre trabalhos relacionados

A seguir, a descrição de cada característica:

Tipo de *feedback*: modelo de *feedback* utilizado para estudo.

Método de coleta: Formas que a aplicação utiliza para capturar o *feedback.*

Recurso para ordenação de resultados: Dados utilizados para otimizar o resultado da busca.

Dados considerados: Tipos de dados que são lidos e processados pela aplicação.

Interface: Ambiente de interação entre a aplicação e o usuário.

Reformulação da pesquisa: Capacidade da ferramenta realizar uma nova consulta para melhorar os resultados.

Capacidade de aprendizado de máquina: Capacidade que a ferramenta tem de aprimorar os resultados com o tempo.

A principal diferença entre a ferramenta desenvolvida e os trabalhos relacionados é que ela é a única a trabalhar exclusivamente com *feedback* explícito onde o usuário pode selecionar documentos relevantes e irrelevantes, além de oferecer três diferentes algoritmos para reescrever a consulta.

Outra vantagem da ferramenta desenvolvida é que o comportamento do usuário não influência em pesquisas futuras. Assim, um clique em um resultado errado não irá alterar o resultado em uma próxima busca.

4. Ferramenta XFind

Este capítulo aborda a ferramenta XFind (X é a consoante inicial de XML e *Find* significa encontrar na língua inglesa), que permite que usuários forneçam *feedback* para obter uma melhor ordenação no resultado da busca em documentos XML.

O capítulo se divide nas seguintes partes: visão geral, onde é introduzido o funcionamento da ferramenta; arquitetura, onde são exibidas as diferentes partes do sistema; conceitos e tecnologias utilizadas pela aplicação; implementação, onde é discutido o desenvolvimento da ferramenta; e execução, que discute o funcionamento da ferramenta.

4.1 Visão geral

A ferramenta trabalha em três estágios diferentes: (I) busca através de similaridade e carga dos documentos; (II) exibição dos resultados em uma estrutura ordenada de acordo com o nível de similaridade; e (III) uma busca a partir dos documentos relevantes escolhidos pelo usuário.

A primeira parte consiste em buscar os documentos mais similares de acordo com as palavras-chave digitadas pelo usuário. Os documentos XML são carregados pela ferramenta e depois são processados para serem lidos e então ordenados de acordo com o grau de similaridade. Após o processo inicial, inicia-se a segunda parte onde os documentos são mostrados em uma interface Web. Para facilitar a pesquisa do usuário, os documentos foram separados em quatro listas de acordo com o grau de similaridade: alta, média,

baixa, muito baixa. Na tela de resultados, é possível o usuário informar ao sistema se o documento é relevante ou não relevante e então, inicia-se a terceira parte. O sistema faz uma nova consulta de acordo com as escolhas do usuário. A nova consulta é resultado do algoritmo que aplica o método de Rocchio.

4.2 Arquitetura

A Figura dois mostra a visão geral da ferramenta desenvolvida, com seus principais módulos.

Figura 2. Visão Geral de funcionamento

O usuário faz sua consulta utilizando palavras-chave que será comparada aos valores de todos os elementos de um arquivo XML. Estes arquivos, que alimentam a aplicação, estão em uma pasta com caminho pré-definido pelo sistema. As comparações entre a palavra-chave e os valores de elementos são feitos através da métrica q-grams, que compara os termos da consulta e os termos dos documentos e atribuí uma pontuação que é o grau de

similaridade. Para cada arquivo, é feito média aritmética de acordo com o valor da similaridade entre os elementos e a palavra-chave, e então o arquivo comparado é adicionado a uma lista de resultados de acordo com o nível de sua similaridade. Por fim, o resultado é exibido para o usuário em quatro grupos divididos pela similaridade: alta, cuja similaridade é maior ou igual que 0.75; boa, cuja similaridade é maior ou igual que 0.50; média, cuja similaridade é maior ou igual a 0.25; e baixa onde a similaridade é entre zero e menor que 0.25.

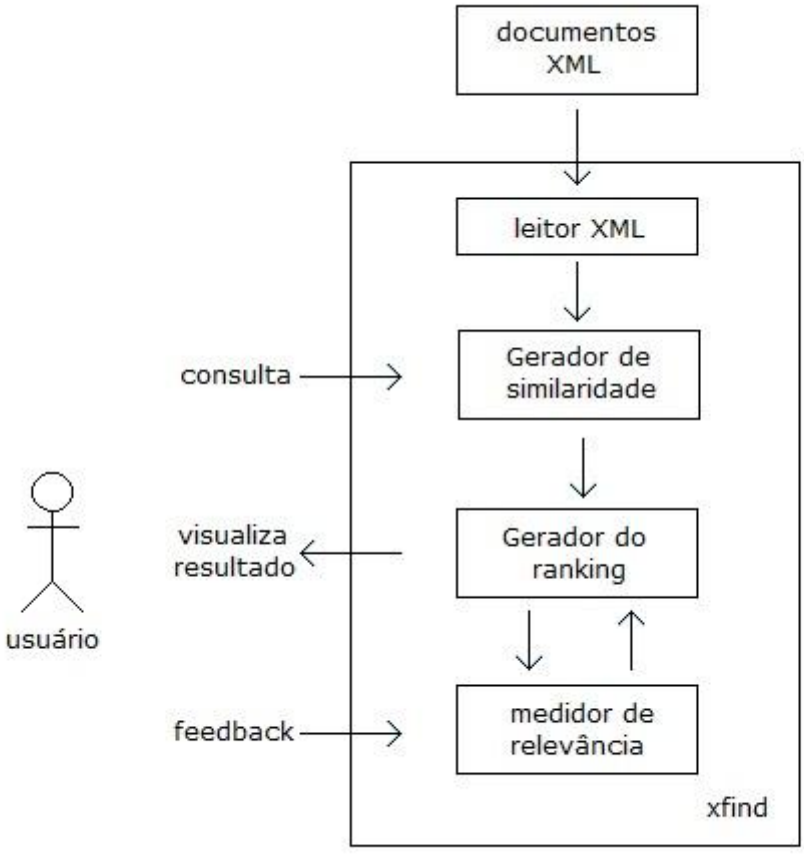

Figura 3. Arquitetura do XFind

A Figura três apresenta a arquitetura do XFind e seus módulos internos. O leitor XML recebe os arquivos XML e processa os elementos e atributos para que possam ser lidos pela aplicação. Isto é feito com o auxílio da biblioteca

JDOM. Ele processa os arquivos no formato XML em uma estrutura de árvore, facilitando a manipulação em arquivos deste tipo. O Gerador de similaridade gera um valor comparando os termos da consulta do usuário com os termos de cada documento processados pelo leitor XML utilizando a métrica de similaridade q-grams, contido na biblioteca SIMMETRICS. O gerador do *Ranking* ordena os resultados de forma decrescente e em quatro grupos de acordo com o nível de similaridade de cada documento e envia os resultados para a interface do usuário. O medidor de relevância recolhe o *feedback* do usuário e calcula a relevância de todos os documentos de acordo com aqueles escolhidos pelo usuário. O gerador do *Ranking* passa a ordenar de acordo com o valor de relevância de cada documento e envia novamente os resultados para o usuário.

4.3 Implementação

Toda a implementação do projeto foi realizada utilizando a IDE Netbeans 7.1 com o código sendo desenvolvido sob as linguagens JAVA 1.7 e JSP[1], utilizando-se das seguintes bibliotecas externas: JDOM[2], para leitura dos arquivos XML[3]; e o SIMMETRICS[4], para utilização da métrica q-grams.

O desenvolvimento da aplicação foi dividido em três partes: busca e carga inicial dos arquivos, exibição dos resultados e implementação da coleta do *feedback* do usuário.

[1] www.oracle.com/technetwork/java/javaee/jsp/
[2] www.jdom.org
[3] www.w3.org/xml
[4] Sourceforge.net/projects/simmetrics

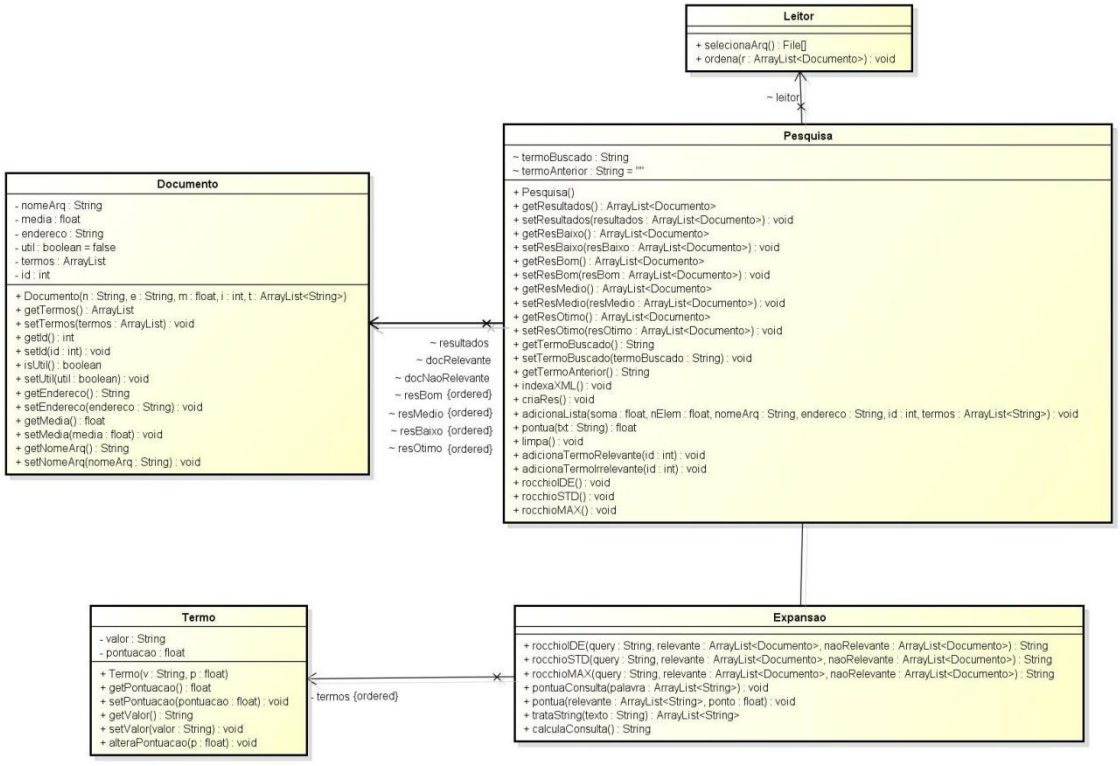

Figura 4. Diagrama de classes

O diagrama de classes da Figura quatro mostra as cinco classes presentes na aplicação. A classe documento é responsável por guardar os dados de cada documento como, por exemplo, o seu grau de similaridade comparada às palavras-chave digitadas pelo usuário. A classe pesquisa é a principal classe do sistema, ela é responsável pela criação e ordenação da lista dos resultados, cálculo da medida de similaridade, e também é responsável por se comunicar com a interface do usuário. A classe Pesquisa processa os dados em *XML* para serem lidos pela aplicação. A classe Expansao é responsável por reescrever a consulta com base no *feedback* do usuário, ela possui o algoritmo Rocchio e suas variações implementadas. Por fim, a classe Termo guarda a pontuação de cada termo que pertence ao grupo de documentos selecionado pelo usuário como relevante ou não relevante.

4.4 Funcionamento da ferramenta

Para o funcionamento da aplicação, o usuário precisa inserir no campo de busca os termos necessários para iniciar a pesquisa, conforme é exemplificado na Figura cinco a seguir.

Figura 5. Campo de busca do usuário

Após o início da busca, o usuário é direcionado para uma página de resultados conforme a Figura seis.

Figura 6. Interface de resultados

A página de resultados possui quatro abas divididas pelo nível de similaridade. Elas facilitam a navegação do usuário para que o mesmo consiga visualizar melhor os documentos que não possuem um alto grau de similaridade, podendo selecionar documentos relevantes mesmo que estes

estejam mal ranqueados. Cada página permite a visualização de até dez documentos ordenados pelo grau de similaridade, porém não há um limite máximo permitido de documentos e páginas. É possível ver o caminho de cada documento no sistema, o grau de similaridade, o nome do arquivo e os termos contidos em cada documento. As imagens com o símbolo de mais “+” e menos “-“ são botões que permitem ao usuário enviar o *feedback* ao sistema. No topo da página também são informadas a consulta atual e a consulta anterior, podendo o usuário comparar a consulta com e sem *feedback*.

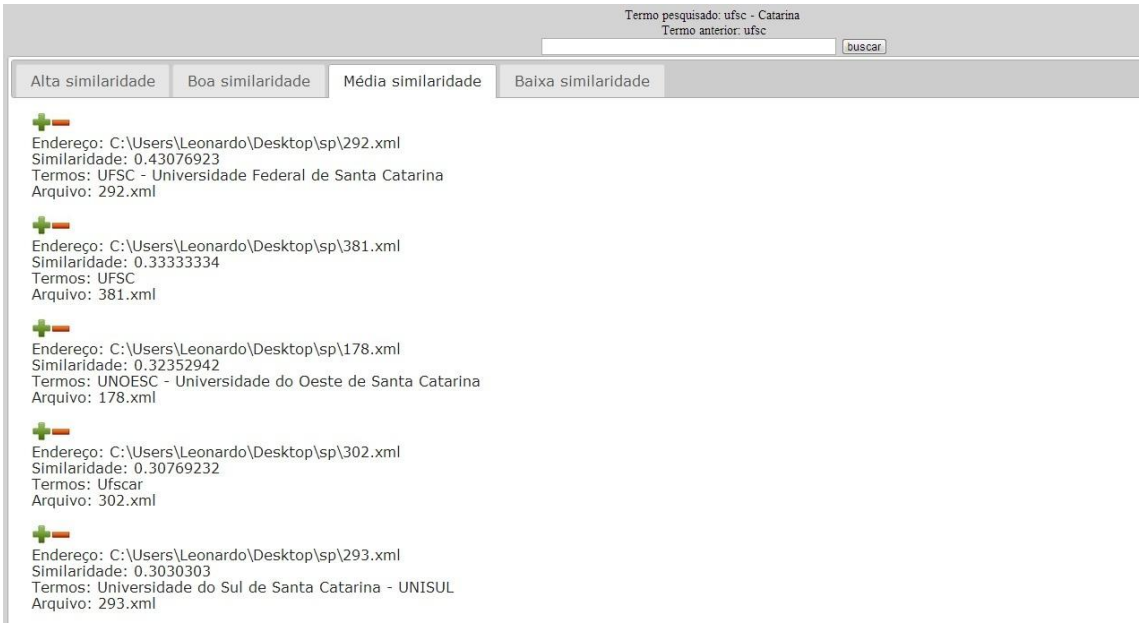

Figura 7. Pagina de resultados com a consulta reescrita

No fim da página, existem três botões para iniciar a nova consulta. O primeiro botão reescreve a consulta com o método Rocchio padrão, já os outros pertencem as suas variações IDE e MAX, respectivamente. Ao clicar em algum dos três botões a consulta é então reescrita com o algoritmo escolhido e a pagina é atualizada com uma nova lista de resultados. O processo pode ser repetido infinitamente.

5. Experimentos

Os experimentos foram feitos utilizando-se documentos pequenos, que possuíssem poucos termos. Para isso, foram feitos testes em três grupos distintos de arquivos: nomes de ruas e nomes de filmes, onde havia poucas semelhanças entre os arquivos; e nome de instituições de ensino, onde os termos contidos em cada documento eram mais semelhantes entre os diferentes arquivos.

Foram utilizadas duas métricas para avaliação dos testes: revocação e precisão. A revocação é a quantidade de documentos relevantes que foram recuperados comparados com a quantidade total de arquivos relevantes. Já a precisão é a quantidade de arquivos relevantes recuperados entre todos os arquivos recuperados.

Quando o valor da revocação chegar a um, significa que todos os arquivos relevantes foram recuperados, e quanto maior o valor da precisão, mais arquivos relevantes estão no topo do *ranking*.

Tipo	Número de documentos	Número de consultas
Nome de ruas	142	6
Nome de filmes	132	6
Instituições	252	6

Tabela 3. Configuração de teste

Para cada grupo de documento, foram executadas seis diferentes consultas e para cada consulta foram selecionados os mesmos documentos relevantes e irrelevantes. Cada consulta foi executada com os três algoritmos

com o objetivo de saber qual era o melhor, e por isto, o valor de ajuste de α, β e γ ficou fixado em um. A seguir estão os gráficos com as médias de revocação e precisão de todos os três grupos de dados (Figuras 8 a 10). O algoritmo IDE_DEC_HI está sinalizado como MAX.

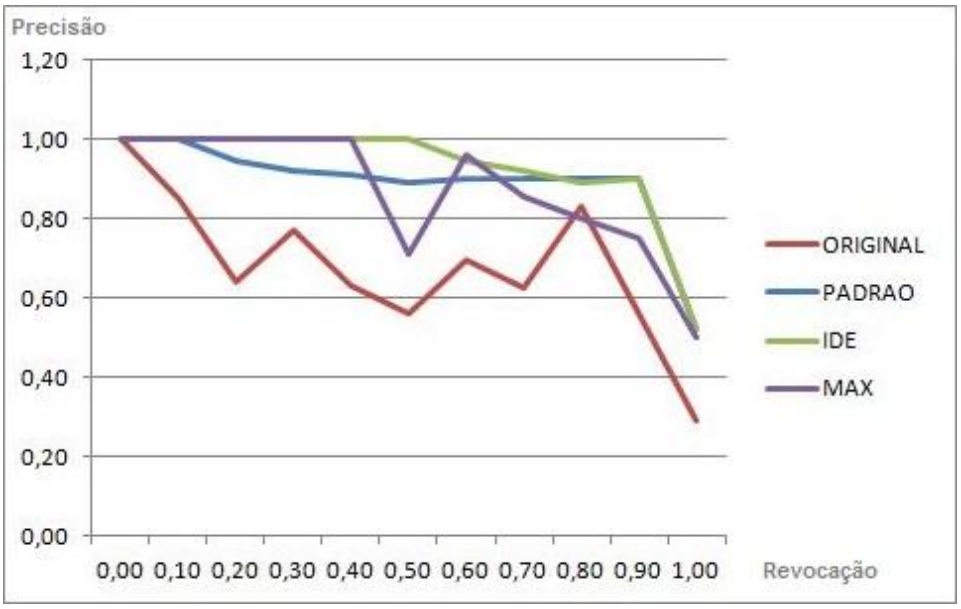

Figura 8. Gráfico com pesquisa com nome de filmes

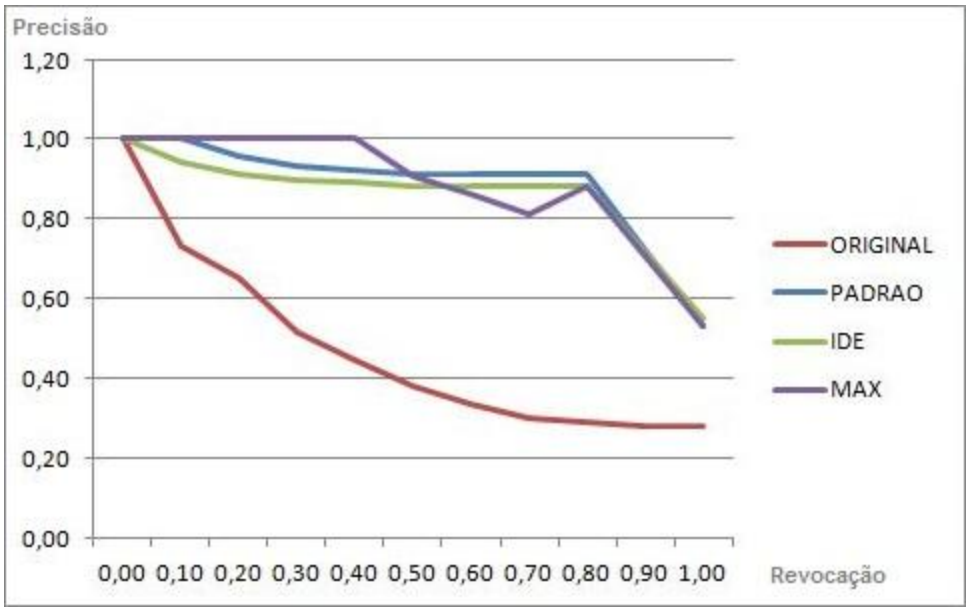

Figura 9. Gráfico com pesquisa com nome de ruas

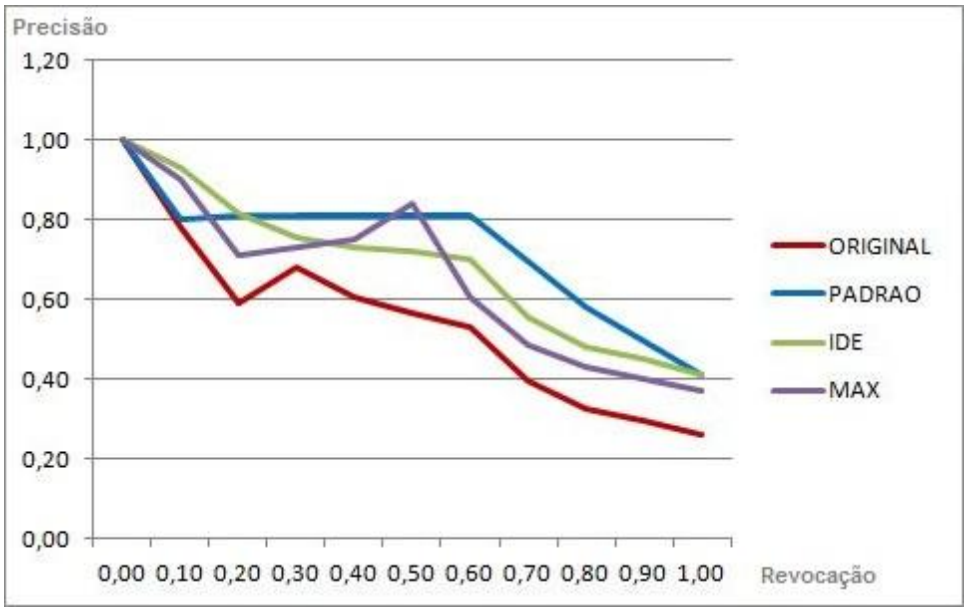

Figura 10. Gráfico com pesquisa com nomes de instituições

Nos dois primeiros gráficos, nome de filmes e de ruas, o algoritmo IDE obteve uma pequena vantagem sobre os demais algoritmos. Já no terceiro gráfico, o algoritmo padrão de Rocchio é o que obteve o melhor resultado. O algoritmo MAX foi o pior algoritmo em todas as situações. Comparando com o original, todos os algoritmos obtiveram resultados superiores.

A seguir, uma tabela com o resultado médio da precisão para todos os grupos e algoritmos.

Tipo	Original		Rocchio Padrão		IDE		MAX (IDE_DEC_HI)	
	Precisão	Revocação	Precisão	Revocação	Precisão	Revocação	Precisão	Revocação
Nome de ruas	0,39	0,59	0,95	0,59	0,96	0,59	0,95	0,59
Nome de filmes	0,56	0,59	0,92	0,59	0,95	0,59	0,91	0,59
Instituições	0,38	0,57	0,69	0,57	0,65	0,57	0,64	0,57
Média	0,44	0,58	0,85	0,58	0,85	0,58	0,83	0,58

Tabela 4. Média de precisão e revocação para cada tipo de teste.

De modo geral, os três algoritmos obtiveram resultados semelhantes. Os testes com nome de ruas e nome de filmes mostraram resultados muito parecidos devido à diferença dos termos entre cada documento. Com o grau de precisão sempre acima de 90%, os algoritmos funcionam bem quando os documentos são menos semelhantes entre si.

Com os testes no grupo de instituições, a precisão caiu aproximadamente 30% para cada algoritmo. Um dos motivos pode ser a maior semelhança o nome das instituições, como por exemplo, Universidade federal de Santa Catarina e Universidade Federal de Santa Maria. Em uma consulta, os documentos escolhidos podem fazer com que o grau de similaridade fique alto para ambas as instituições. Outro motivo é que uma instituição pode ser chamada de diferentes formas, por exemplo, UFSC e Universidade Federal de Santa Catarina. O grau de similaridade entre estes dois nomes é baixo o que pode fazer um deles ficar no topo do ranking e o outro em uma posição sem destaque.

6. Conclusão

Atualmente, grande parte sistemas de recuperação de informação, como o *Google* e o *Bing*, utilizam a técnica de *feedback* implícito para aumentar a precisão de suas consultas. Quando o usuário não obtém resultados satisfatórios, ele é obrigado a reescrever sua consulta para que um melhor resultado seja obtido.

A proposta do trabalho é o desenvolvimento de uma ferramenta de busca para arquivos XML onde, a partir do resultado de uma pesquisa inicial, o usuário indique quais documentos são relevantes e quais documentos não são relevantes e então a própria ferramenta reescreva a consulta. A ordenação do *ranking* com os resultados é feita através da comparação de similaridade da consulta com os termos de cada documento. A métrica de similaridade escolhida foi o q-grams*, pois* permite que os termos quando comparados possuam alto grau de similaridade mesmo quando não estão na mesma sequencia seja na consulta ou no documento. O algoritmo implementado para reescrever a busca foi o método de Rocchio, e além dele foram implementadas duas variações para comparação.

Por fim, foram feitos alguns experimentos para verificar quais dos algoritmos implementados conseguia obter a melhor precisão. Os arquivos de testes eram documentos com poucos termos e foram divididos em três grupos: nome de ruas, instituições de ensinos e nome de filmes. Cada grupo foi testado com a mesma consulta para cada algoritmo.

Durante o trabalho, algumas necessidades e ideias foram levantadas e incluídas dentre os trabalhos futuros, que estão listados a seguir.

1. Possibilidade de processar arquivos que sejam diferentes de XML, como por exemplo, HTML.

2. Implementação da métrica TF-IDF (*term frequency–inverse document frequency*) (ROBERTSON, 1972) para que arquivos com grande volume de termos possam ser buscados com maior eficiência.

3. Tornar os quatro grupos de resultados na interface dinâmicos com base na similaridade dos documentos recuperados. Com valores fixos, é possível que um ou vários grupos fiquem vazios. Assim os grupos agora seriam divididos igualitariamente dos mais semelhantes ao menos semelhantes.

4. Acrescentar suporte a bancos de dados XML, assim a busca não ocorreria apenas em diretórios de arquivos.

5. Com o objetivo de melhorar o resultado das buscas, implementar um treinamento inicial para um futuro suporte a *machine learning* e inclusão de semântica na pesquisa utilizando elementos do XML.

7. Referências

Agichtein, Eugene; Brill, Eric; Dumais, Susan. Improving Web Search Ranking by Incorporating User Behavior Information. ACM, 2006. Artigo.

BAEZA-YATES, Ricardo B.; NETO, Berthier R. Modern Information Retrieval. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

Bing, L. Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data. Second Edition, July 2011. Springer

Cardoso, Olinda Nogueira Paes. Recuperação de informação. Universidade Federal de Lavras, 2009. Artigo.

Cendon, Beatriz Valadares. Ferramentas de busca na *Web.* Ci. Inf. [online]. 2001, vol.30, n.1, pp. 39-49. ISSN 0100-1965

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

Joachims, Thorsten; Radlinski, Filip. Search Engines that Learn from Implicit Feedback. Cornell University, 2007. Artigo.

Jung, Seikyung; Herlocker, Jonathan L; Webster, Janet. Click data as implicit relevance feedback in web search. Elsevier, 2006. Artigo.

Kondrak, Grzegorz. N-gram similarity and distance. University of Alberta, 2005. Artigo.

Robertson, Stephen. Term specificity. Journal of Documentation, Vol. 28, pp. 164–165. 1972.

Rocchio, J. J. The SMART Retrieval System - Experiments in Automatic Document Processing. Prentice Hall, 1971.

Ukkonen, E. Approximate string-matching with q-grams and maximal matches. Theoretical Computer Science. University of Helsinki, 1992. Artigo

White, R.W. and Jose, J.M. and Ruthven, I. (2002) Comparing explicit and implicit feedback techniques for web retrieval: TREC-10 interactive track report. In: Proceedings of the Tenth Text Retrieval Conference (TREC-10), 2001, Maryland, USA.