Recuperação de informação em documentos XML

Desde 2000, quando aconteceu o primeiro workshop sobre XML e Recuperação de Informação, este tema tem estado presente nas conferências ACM SIGIR, mostrando o interesse da comunidade de Recuperação de Informação em explorar melhor as informações semi-estruturadas, dando a estas um enfoque de RI, em contrapartida ao enfoque de Banco de Dados que sempre receberam. Esta abordagem é importante para estender o papel do XML além do objetivo de troca de dados na Web , tornando-o interessante para troca de informação, dentro de toda a flexibilidade e liberdade que caracterizam a Web.

Em 2002, na Finlândia, durante o 2° workshop sobre o tema, debates sobre o sentido da recuperação de documentos XML, levaram a conclusão de que é importante trata-la desvinculada de operações de Banco de Dados.

Tendo estudado a Web Semântica anteriormente, e concluído ser uma proposta ambiciosa demais para a diversidade da Web e pouco disseminada até o momento, optamos por recuar um passo atrás e estudar de que forma as informações semi-estruturadas, e mais especificamente o XML, podem contribuir para melhorar a precisão e revocação das máquinas de busca. Fomos motivados pelo entendimento de que as *tags* ajudam a definir o significado dos termos que delimitam, podendo, conseqüentemente, contribuir para a precisão dos resultados de uma pesquisa.

Com este objetivo, os principais artigos encontrados sobre o assunto foram estudados e comparados, permitindo algumas conclusões que servirão de base para a escolha de futuros caminhos.

Apresentaremos um breve apanhado de cada um dos artigos abordando suas características mais marcantes, os principais avanços e limitações, seguido de um quadro comparativo. E fechando o presente trabalho apresentaremos nossas conclusões e uma proposta para trabalhos futuros.

Artigo [1] - Searching Text-rich XML Documents with Relevance Ranking

Assumindo o compromisso de utilizar o máximo possível as técnicas convencionais de Recuperação de Informação, seu principal objetivo é permitir a recuperação de documentos XML ordenados por relevância, em contraposição às diversas linguagem que tratam apenas as pesquisas exatas. As consultas serão genéricas, e os resultados não serão exatos, permitindo a ordenação por relevância, calculada a partir do grau de similaridade entre a consulta e o documento.

O Artigo considera intratável a recuperação de documentos XML com estruturas arbitrárias e impõe restrições aos tipos de sub-estruturas dos documentos que poderão ser especificadas na consulta. É definido o conceito de "campos de pesquisa " mapeados em sub-estruturas dos documentos.A partir dessa decisão de projeto, somente sub-estruturas especificas, chamadas de *tag-path*, poderão ser apontadas por uma consulta.

Os trechos em negrito na figura 1.0 correspondem a *tag-paths*.

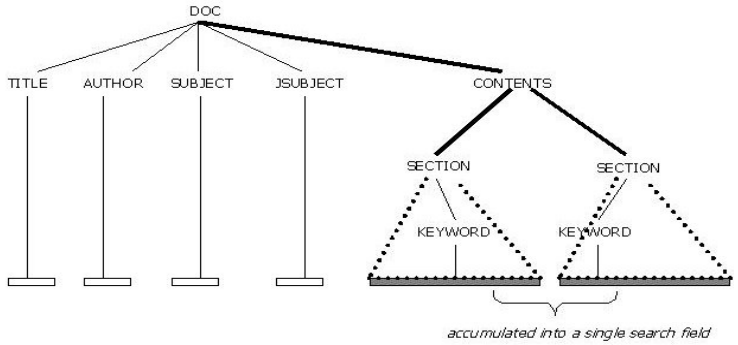

Fig 1.0

Os princípios básicos do projeto são:

- O sistema indexa e pesquisa documentos XML bem-formados, sem considerar as DTDs;
- Um número finito de sub-estruturas pesquisáveis serão definidas previamente à indexação . A associação entre campos de pesquisas e *tag-paths* será definida em um arquivo chamado Format File;
- O indexador verifica o Format File e cria um arquivo para cada "campo de pesquisa";
- Um serviço de aplicação deverá formular uma consulta aceitável pela máquina de busca a partir da informação requisitada pelo usuário. Para isso, este serviço deverá conhecer os "campos de pesquisa" e sua associação semântica com as sub-estruturas dos documentos XML.

A implementação da máquina de busca possui a arquitetura mostrada na figura 2.0 , cujos módulos funcionais detalhamos a seguir:

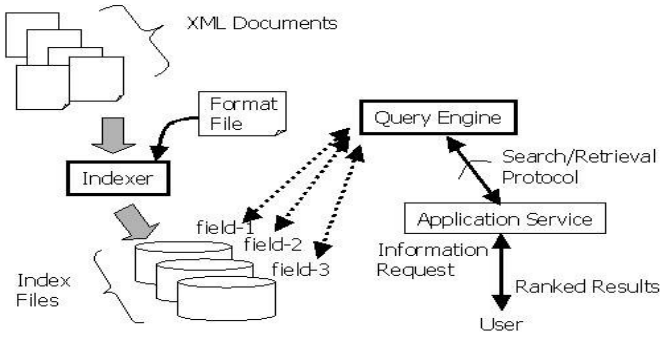

Fig 2.0

Format File - Este arquivo destina-se a definir os índices que serão criados para um conjunto de documentos da coleção. Em sua estrutura ele traz o nome e o formato dos arquivos de índices, as associações entre *tag-paths* e "campos de pesquisas" e a localização de processadores específicos para o idioma dos documentos.

Indexer – É o módulo responsável pela interpretação do Format File e posterior criação dos índices nele definidos. Os termos contidos em cada campo de um documentos receberão pesos utilizando a fórmula tradicional de TFxIDF. Neste módulo ocorrerá a leitura, extração dos termos e o cálculo de seus pesos .

Query Engine - A máquina de pesquisa foi implementada acrescentando-se extensões a uma máquina de busca de texto já existente. Ela permite a busca por termos ou *tags*.

Aplication Service – É o módulo responsável pela interpretação da estrutura da consulta proposta pelo usuário, para identificar o "campo de pesquisa" e o respectivo arquivo de índice.

Não há esclarecimentos sobre como o Format File será criado, nem como consultas sem indicação de sub-estrutura serão tratadas. Os resultados apresentados referem-se apenas ao tempo de processamento necessário à geração dos índices e ao tamanho dos mesmos, mas nenhuma análise sobre a precisão dos resultados é apresentada.

A grande limitação desta proposta é a pré-definição da estrutura dos índices através dos Format Files, diminuindo a flexibilidade das pesquisas, mas representa a primeira tentativa de explorar a estrutura XML em benefício da recuperação de informação.

Artigo [2] – Generating Vector Spaces On-the-fly for Flexible XML Retrieval

O foco dessa proposta é criar mecanismos que permitam ao usuário de uma máquina de busca compor consultas que explorem a estrutura dos documentos XML, obtendo resultados ordenados por relevância de acordo com uma granularidade desejada. Em busca deste objetivo, rompe as barreiras da máquina de busca tradicional que enxergam os documentos de forma plana e também as limitações das linguagens de consulta de XML que efetuam basicamente pesquisas exatas.

A ordenação por relevância leva em conta a estrutura do documento e as restrições que a consulta impõe sobre esta estrutura. Mais especificamente, os conteúdos em diferentes níveis da árvore que representam o documento terão importância diferente para consultas diferentes. Os autores utilizam a idéia de Fuhr [4] que introduz pesos chamados de *augmentation weights* atribuídos a estatísticas como TFxIDF de cada termo, conforme a sua posição na árvore. Os elementos presentes na estrutura de cada documento são agrupados em nós de indexação que implementam as listas invertidas (vide fig 3.0)

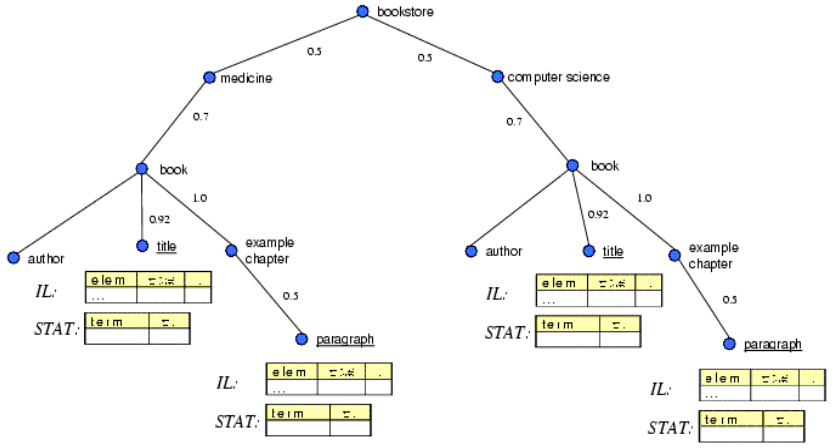

Fig 3.0

As consultas são agrupadas em três tipos diferentes, conforme o escopo da árvore que pretendem abranger:

Single_category – consultas que buscam termos em apenas uma sub-árvore. Por exemplo consultas interessada apenas em livros de medicina, na arvore mostrada na fig 3.

Multi-category – consultas que buscam termos em mais de uma sub-árvore. Por exemplo consultas interessadas em livros sobre medicina e biologia.

Nested- category – consultas que buscam termos em toda uma sub-árvore mas cujos elementos possuem diferentes relevâncias para uma mesma consulta. Por exemplo o elemento título e parágrafo no exemplo. (a presença de um termo no título certamente será mais relevante que a presença do mesmo termo num parágrafo.)

Para cada uma das três categorias de consulta os termos terão pesos diferentes, dependendo da estrutura da consulta. Estes pesos deverão ser calculados dinamicamente a partir de um índice básico pré-calculado.

Os índice básicos serão criados para cada elemento da árvore que possuem conteúdo textual , sendo que o artigo não detalha como isso ocorrerá, mas apenas sugere que poderão ser criados a partir de

técnicas padrões de RI como extração de termos e eliminação de *stop words*. Partindo destes índices básicos ,cada tipo de consulta terá uma fórmula especifica obtidas a partir do Modelo Vetorial.

$$RSV(e,q) = \sum_{t \in q} tf(t,e)\ ief_{cat}(t)^2\ tf(t,q)$$ Eq.1.0 – Single-category

$$RSV(e,q) = \sum_{t \in a} tf(t,e)\ ief_{mcat}(t)^2\ tf(t,q)$$ Eq. 2.0 - Multi-category

$$RSV(e,q) = \sum_{se \in SE(e)} \sum_{t \in q} tf(t,se)\ \Big(\prod_{l \in path(e,se)} aw_l\Big)^2 ief_{cat(se)}(t)^2\ tf(t,q)$$
$$= \sum_{se \in SE(e)} \Big(\Big(\prod_{l \in path(e,se)} aw_l\Big)^2 \sum_{t \in q} tf(t,se)\ ief_{cat(se)}(t)^2\ tf(t,q)\Big)$$

Eq. 3.0 – Nested_category

Em todas as equações observa-se que as estatísticas são calculadas para cada elemento na estrutura do documento e não para cada documento, como na fórmula original do Modelo Vetorial. Na equação 2.0 a freqüência do elemento na coleção será a somatória da freqüência do elemento nas diversas categorias abrangidas pela consulta (*ief* $_{mcat}$). Para a consulta aninhada (equação 3.0) a relevância será a soma ponderada da relevância para uma categoria simples, onde os pesos serão os *augmentation weights*

O grande avanço deste artigo é a flexibilidade admitida no processo de indexação, criando índices específicos para cada documento sem restrições em sua estrutura. Em contrapartida o processamento da consulta torna-se mais demorado tendo em vista a maior complexidade das fórmulas especialmente para consultas em multi-categorias e categorias aninhadas.

A intenção de tratar diferentes granularidades de consulta, por outro lado, exigem do usuário um conhecimento da estrutura do documento ou a sua dedução. Não há a possibilidade de uma busca integrada em documentos XML ou não, uma vez que os índices trazem estatísticas a nível de elemento e não de documento. Destina-se portanto, exclusivamente à recuperação de informação em documentos XML.

Artigo [3] – An Extension of the Vector Space Model for Querying XML Documents via XML Fragments

Esta proposta busca criar uma ferramenta de pesquisa mais amigável, onde usuários possam expressar suas consultas na forma de texto livre ou através de consultas mais complexas dependendo do conhecimento que possuem das DTDs. O resultado também será ordenado por relevância colocando no topo do *ranking* documentos cuja estrutura mais se aproxime daquela proposta na consulta.

O *ranking* será gerado a partir de uma extensão do modelo vetorial que utilizará como unidades de indexação não apenas termos, mas pares da forma **(t_i,c_i),** onde os termos serão qualificados pelo contexto onde aparecem. Na equação 4.0, o peso de termos individuais será substituído pelo peso dentro de uma contexto, **wd(t,c)**. Mas além disso o modelo vetorial deixa de ter dimensões ortogonais entre **t** e **c**, passando a considerar diferentes níveis de semelhança entre os contexto da consulta e do documento, representada pelo fator **cr(ci,ck),** conforme equação 5.0.

$$\mathbf{RVS(q,d) = \sum wq(t_i)*wd(t_i)/ |Q|*|D|}$$ Eq. 4.0

$$\mathbf{RVS(q,d) = \sum_{ci} \sum_{ck} wq(t_i,c_i)*wd(t_i,c_k)*cr(c_i,c_k)/ |Q|*|D|}$$ Eq. 5.0

Durante a indexação os documentos XML serão percorridos e um vetor de pares **(t,c)** será extraído para criar o perfil de cada documento. Armazenando os termos e seus contextos, a lista invertida do

termo **t**, contendo todos os documentos onde **t** aparece, será divida em diversas listas, uma para cada contexto, permitindo assim a recuperação do termo em determinado contexto. A estrutura dos indexadores armazenará **t** e **c** como uma única chave **t#c** e no momento da recuperação o sistema poderá identificar ocorrências precisas de **t** dentro de um contexto **c**. Poderá, também, recuperar todos os contextos onde **t** aparece fazendo uma junção de todas as listas através do sufixo **t#**. As consultas poderão ser formuladas contendo os termos dentro de contextos ou apenas termos, como numa máquina de busca tradicional, ou poderá também conter uma mistura das duas situações, como por ex:

<article><title> XML tutorial <title><article> relating to XPath XQquery.

Gerando o seguinte conjunto **(t,c) ,** para pesquisa:

{(xml,article/title), (tutorial, article/title), (relating, null), (Xpath, null), (XQuery, null)}

O Algoritmo de ordenação levará em conta o peso do termo no contexto, bem como a semelhança entre os contextos.O Cálculo de **wd(t,c)** é trivial e utiliza as estatísticas da chave **t#c** , **t#c'**, etc. Para o cálculo de **cr** foram definidos alguns critérios e para cada um foi criado um fator. Observe o efeito de cada um deles para a consulta **Q=/book/chapter/title**:

LCS(Q,A): Longest Common Subsequence - Maior seqüência comum entre os contextos Q e A

A	**LCS**	POS	GAPS	LD	CR
media/**book/chapter/title/**number	**3**	3	0	2	0.84
media/**chapter/book/title/**number	**2**	3	0	3	0.53
media/**title/chapter/book/**number	**1**	2	0	4	0.29
magazine/volume/article/**title/**number	**1**	4	0	4	0.19

POS(Q,A): Average Optimal Position – Posição média ótima

A	LCS	**POS**	GAPS	LD	CR
/book/chapter/title/subtitle/number	3	**2**	0	2	0.92
media/**book/chapter/title/**number	3	**3**	0	2	0.84
media/catolog/**book/chapter/title**	3	**4**	0	2	0.75

GAPS(Q,A): Número de saltos existentes entre os elementos comuns aos contextos Q e A

A	LCS	POS	**GAPS**	LD	CR
Media/catalog **/book/chapter/title/**subtitle/number	3	4	**0**	4	0.78
catolog/**book/**chapters/**chapter/**section/**title/**numbe r	3	4	**2**	4	0.68

LD(Q,A): Length Difference – diferença de tamanho entre os contextos Q e A

A	LCS	POS	GAPS	**LD**	CR
/book/chapter/title/subtitle/subtitle/number/bulle t	3	2	0	**4**	0.88
book/chapter/title/subtitle	3	2	0	**1**	0.95

A fórmula final de **cr** é dada por: **cr(Q,A) = αLCS(Q,A) + βPOS(Q,A) - ?GAPS(Q,A) – dLD(Q,A)**
Cujos valores poderão variar entre 0 e 1.

Os documentos recuperados serão mostradas de forma semelhante a uma máquina de busca tradicional exceto pelo fato de permitir ao usuário a escolha de elementos dos documento que deseja mostrar no ranking.

O artigo também não apresenta curvas de precisão e revocação, apesar de ter utilizado a coleção INEX [http//qmir.dcs.qmw.ac.uk/inex] para testes.

Constata-se aqui um avanço ao permitir que sejam pesquisados documentos XML ou não. Há um tratamento dinâmico da semelhança entre os diversos contextos. Mas para que a estrutura dos documentos possam contribuir para a ordenação, as consultas deverão informar os elementos desejados, caso contrário todas as listas de um termo, em todos os contextos, serão aglutinadas constituindo-se assim um caso especial de consulta do Modelo Vetorial tradicional. Ou seja, a estrutura do documento não é avaliada para contribuir para a ordenação dos documentos a menos que o usuário tenha uma noção da estrutura dos mesmos e resolva explorar isso na formulação da consulta.

Conclusão

Concluindo o trabalho apresentamos abaixo um quadro comparativo das três propostas, enfatizando aspectos de funcionalidade de cada um:

Item	**Artigo [1]**	**Artigo[2]**	**Artigo[3]**
Usuário deve conhecer a estrutura dos documentos	Não	Não	Não
Há restrição na estrutura dos Documentos	Sim	Não	Não
Pesquisa documentos não XML	Não	Não	Sim
Ordenação por relevância	Sim	Sim	Sim
Explora a estrutura do documento independente da consulta	Sim	Não	Não

Observamos que todas ordenam os documentos por relevância, ou seja, propiciam a busca aproximada, enquadrando-se como uma alternativa para as linguagens que trabalham apenas com busca exata.

A primeira proposta restringe a estrutura dos documentos da coleção àquela prevista no Format File, já as outras duas alternativas trabalham amplamente com qualquer estrutura de documento da coleção.

A proposta 3 permitem a busca em documentos não XML podendo, portanto, ser implementadas em máquinas de busca de uso geral.

Nenhuma das alternativas exige que o usuário conheça a estrutura do documento. A primeira identifica os campos de pesquisa a partir da requisição do usuário, no módulo Serviço de Aplicação. Nas demais alternativas, caso o usuário não insira elementos da estrutura dos documentos na formulação da consulta a pesquisa recaíra no Modelo Vetorial tradicional.

Trabalhos futuros

Estudando as três alternativas e observando os aspectos de funcionalidade que seriam desejáveis em uma máquina de busca para documentos XML, consideramos que uma ferramenta com este propósito deveria explorar a estrutura dos documentos independente de estar explícito na formulação da consulta , ou seja, explorar esta estrutura como uma fonte adicional de evidência, sem exigir do usuário qualquer noção sobre a mesma.

Analisando o problema , dividimos o universo de busca em três componentes básicos:
O termo (**t**), o elemento (**e**) e o documento (**d**). A partir daí podemos seguir três linhas básica:

- Combinamos termos e elementos, e calculamos TF e IDF sobre esta combinação como-se os termos do Modelo Vetorial fossem na verdade um termo composto, t#c;
- Consideramos o elemento como uma subdivisão do documento e calculamos as estatísticas do Modelo Vetorial sobre o elemento, ao invés de trata-la sobre o documento;
- Combinarmos as duas opções anteriores, e trabalharmos com todas as estatísticas envolvendo **t** , **e** e **d.**

No primeiro caso o *ranking* de um documento será dado pela somatória das contribuições de cada contexto onde os termos da pesquisa aparecem.

No segundo caso o *ranking* de um documento será dado pela somatória dos *rankings* de cada elemento. Documentos com elementos raros na coleção e documentos com elementos onde o termo aparece com maior freqüência terão maior *ranking*. Mas documentos com maior número de elementos contendo os termos da pesquisa também serão favorecidos.

As propostas anteriores assemelham-se ao terceiro e segundo artigo respectivamente, exceto pelo fato de trabalhar sem a especificação de contexto na consulta.

No terceiro caso *o ranking* de um documento será dado pela somatória da contribuição de cada termo, que levará em conta a freqüência de cada termo nos elementos, de cada elemento nos documentos, e também da freqüência dos elementos na coleção e dos documentos que contem os elementos na coleção. Casos especiais onde termos e elementos só aparecem juntos deverão ser favorecidos no *ranking*, caracterizando um acréscimo de semântica ao termo, devido a um contexto bem definido.

Acreditamos que uma formulação correta das estatísticas envolvendo **t**, **e** e **d** poderão levar a resultados interessantes, sem a necessidade de especificação de contexto pelo usuário, como acontece nas três propostas apresentadas.

Referências

[1] Y. Hayashi, J. Tomita e G. Kikul, "Searching Text-rich XML Documents with Relevance Ranking", ACM SIGIR 2000 Workshop on XML and Information Retrieval, Atenas, Grécia , 28 de Julho de 2000 .

[2]T. Grabs e H. J. Schek, "Generating Vector Spaces On-the-fly for Flexible XML Retrieval", ACM SIGIR 2002 Workshop on XML and Information Retrieval, Tampere, Finlândia, Agosto 2002.

[3] D. Carmel, N. Efraty, G. Landau, Y. Maarek e Y. Mass, "An Extension of the Vector Space Model for Querying XML Documents via XML Fragments", ACM SIGIR 2002 Whorkshop on XML and Information Retrieval, Tampere, Finlândia, Agosto de 2002.

[4] N. Fuhr e K. Grojohann. "XIRQL: A Query Language for Information Retrieval in XML documents", ACM SIGIR Conference on Research and Development in Information Retrieval, páginas 172-180 ACM Press , 2001.