COPPE/UFRJ

SEEKLEX – RECUPERAÇÃO DE INFORMAÇÃO EM SUBDOCUMENTOS HIERARQUIZADOS

Renato do Amaral Crivano Machado

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Zimbrao da Silva

Rio de Janeiro
Outubro de 2010

SEEKLEX – RECUPERAÇÃO DE INFORMAÇÃO
EM SUBDOCUMENTOS HIERARQUIZADOS

Renato do Amaral Crivano Machado

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof. Geraldo Zimbrão da Silva, D.Sc.

_____

Prof. Geraldo Bonorino Xexéo, D.Sc.

_____

Prof. Sean Wolfgand Matsui Siqueira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
OUTUBRO DE 2010

Machado, Renato do Amaral Crivano

Seeklex – Recuperação de Informações em Subdocumentos Hierarquizados/ Renato do Amaral Crivano Machado. – Rio de Janeiro: UFRJ/COPPE, 2010.

XXII, 78 p.: il.; 29,7 cm.

Orientador: Geraldo Zimbrão da Silva

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2010.

Referencias Bibliográficas: p. 71-74.

1. Busca e recuperação de informações. 2. Recuperação em documentos estruturados. I. Silva, Geraldo Zimbrão da. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Titulo.

DEDICATÓRIA

A meu avô, Annibal Ferreira do Amaral Filho, que sempre estará em meu coração.

Agradecimentos

Primeiramente, gostaria de agradecer ao Prof. Geraldo Zimbrão por me convencer da oportunidade de cursar o mestrado na COPPE/UFRJ, mesmo estando há vários anos afastado do meio acadêmico. Também por ter sido meu orientador na dissertação e por ter aceitado prontamente um tema que não está diretamente ligado às suas principais linhas de pesquisa.

Agradeço especialmente ao Prof. Geraldo Xexéo, com quem cursei a disciplina de Busca e Recuperação de Informações. No decorrer dessas aulas escrevi – juntamente com o colega Christian Diego – um artigo que serviu de ponto de partida para a presente dissertação.

Agradeço, também, aos professores Jano Moreira de Souza, Jonice Oliveira, Sérgio Exel e Luis Alfredo Vidal, com quem tive o prazer de cursar excelentes disciplinas. A Carlos Mello, doutorando e colega, que muito me auxiliou no desenrolar do curso.

Agradeço a João Alberto de Oliveira Lima por suas sugestões e por disponibilizar artefatos do projeto LexML que viabilizaram este trabalho.

Obrigado aos meus pais, por sempre incentivarem meus estudos além de tantas outras coisas. À minha irmã Cristiana, pela cuidadosa revisão destes escritos. E aos meus filhos, por serem compreensivos nos momentos que tiveram de abrir mão de minha companhia para que eu pudesse me dedicar aos estudos.

Por fim, quero agradecer a Mônica Samico que, com seu amor e sua alegria, me ajudou na etapa mais difícil que enfrentei no mestrado: a conclusão desta dissertação.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SEEKLEX – RECUPERAÇÃO DE INFORMAÇÃO EM SUBDOCUMENTOS HIERARQUIZADOS

Renato do Amaral Crivano Machado

Outubro / 2010

Orientador: Geraldo Zimbrão da Silva

Programa: Engenharia de Sistemas e Computação

Nesta dissertação apresentamos um apanhado de estratégias de busca e recuperação de informações desenvolvidas especificamente para pesquisar no ordenamento jurídico, e que se valem da estruturação hierárquica das leis para obter resultados melhores do que os atingidos pelos métodos convencionais.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SEEKLEX – INFORMATION RETRIEVAL
ON HIERARCHICAL SUBDOCUMENTS

Renato do Amaral Crivano Machado

October / 2010

Advisor: Geraldo Zimbrão da Silva

Department: Computer and Systems Engineering

In this dissertation, we present diverse strategies of information retrieval developed specifically to search the legal literature, and that use it's hierarchical structure to obtain better results than those achieved by the traditional methods.

Índice

Índice de Figuras

Índice de Quadros

Capítulo 1 – Introdução

1.1 – Motivação

Técnicas de arquivamento, organização de índices e recuperação de documentos existem desde a criação das primeiras bibliotecas. Os computadores, dotados de enorme capacidade de processamento e armazenamento, se tornaram ferramentas fundamentais nesse contexto. Com o advento da Internet, o número de documentos a serem organizados se tornou muito maior; consequentemente, mais importância foi dada aos algoritmos de busca e recuperação de informações, que – hoje em dia – representam uma importante área de estudos da computação, atraindo pesquisadores do mundo todo.

Encontrar a informação relevante rapidamente e apresentar de maneira adequada os resultados requer o uso de técnicas e ferramentas apropriadas. A área da computação que trata dessas questões é chamada de Busca e Recuperação de Informações.

O sentido de recuperação de informação é muito amplo. No contexto dessa dissertação, podemos adotar a definição:

> "Recuperação de Informação (RI) é encontrar material (geralmente documentos) de uma natureza desestruturada (geralmente texto) que satisfaça uma necessidade de informação a partir de grandes coleções (geralmente em servidores locais ou na Internet)." - Manning, Raghavan & Schütze (2008, p. 1)

Empresas como Yahoo! e Google, que surgiram a partir da necessidade de organizar as informações disponíveis na Internet, hoje despontam como algumas das mais ricas companhias do mundo.

Devido à topologia da Internet, a maior parte dos métodos de RI considera que a unidade mínima de recuperação de informação é a página web, ou, mais precisamente, um documento. No entanto, existem determinados contextos onde precisamos buscar com uma granularidade mais fina, como é o caso das leis.

No ordenamento jurídico – as leis – observamos que os métodos tradicionais de busca e recuperação de informações não produzem bons resultados, fazendo-se então necessária uma adaptação desses métodos.

Determinados documentos são muito extensos, e para realizar buscas coerentes é necessário dividi-los em partes menores. No caso das leis, podemos aproveitar a sua estrutura fortemente hierárquica, e segmentar cada lei em: título, capítulo, seção,

subseção, artigo, parágrafo, inciso e alíneas.

Desta forma, a lei, que antes era vista como um documento único, passa a ser representada por uma série de subdocumentos (artigos, parágrafos, incisos, etc.) organizados de forma hierárquica, ou seja, em um determinado capítulo existem seções e cada seção tem artigos, que por sua vez possuem parágrafos, incisos, alíneas, etc.

Na presente dissertação, procuramos estudar as peculiaridades dos subdocumentos hierarquizados e apresentar detalhadamente a evolução de um sistema de busca e recuperação de informações específico para esse contexto. Desta forma, pretendemos contribuir para que os escritórios de advocacia e os tribunais tenham à disposição recursos tecnológicos sofisticados, que atualmente são mais difundidos somente em outras áreas de negócio (JENKINS J., 2008).

Outro aspecto a que precisamos estar atentos quando realizamos buscas no ordenamento jurídico é a necessidade de recuperar todas as ocorrências relevantes, pois todas podem ser importantes, por exemplo, para um juiz que precisa proferir uma sentença.

Embora muitos trabalhos na área de busca e recuperação de informações tenham sido desenvolvidos, relativamente poucos tratam de pesquisas em subdocumentos. E, mais especificamente, pesquisas que utilizem principalmente a estrutura hierárquica dos subdocumentos são ainda mais raras.

1.2 – Objetivo

O objetivo desta dissertação é desenvolver métodos específicos para realizar busca e recuperação de informações em subdocumentos hierarquizados. Para isso, propomos duas estratégias:

1. método de busca composto de diversos algoritmos que visam a encontrar e calcular a relevância dos subdocumentos que satisfaçam uma determinada consulta;
2. técnica de visualização de resultados hierárquicos que favorece a compreensão, pois contextualiza, reorganiza e reduz a quantidade de texto apresentada.

Uma aplicação web denominada Seeklex foi desenvolvida a partir das estratégias propostas, e foi disponibilizada na Internet no endereço: http://www.seeklex.com.br.

Por fim, o desempenho do Seeklex foi avaliado e comparado com as técnicas consagradas de busca e recuperação de informações.

1.3 – Estrutura

Esta dissertação está organizada em 5 capítulos, sendo o primeiro esta introdução.

No segundo capítulo apresentamos uma revisão da literatura de busca e recuperação de documentos, com um foco especial em artigos voltados para subdocumentos, documentos estruturados e hierarquizados. Também citamos algumas aplicações de RI e inteligência artificial voltadas para as leis e atividade jurisdicional.

No capítulo 3, são expostos os algoritmos e estratégias que desenvolvemos especialmente para resolver as questões relativas à recuperação de informações em subdocumentos hierarquizados.

O capítulo 4 consiste em uma descrição detalhada da implementação do Seeklex e de sua avaliação.

O último capítulo desta dissertação apresenta as conclusões, considerações finais e propostas para trabalhos futuros.

Capítulo 2 – Revisão da literatura

2.1 – Introdução

O conceito de subdocumento por vezes se confunde com o de documentos estruturados, pois ao extrairmos partes de um documento, estamos naturalmente criando subdocumentos que possuem uma relação estrutural, quer seja a relação de sequência ou a de hierarquia.

Liu et al., em um artigo publicado em 2008, fazem uma revisão da área de recuperação de documentos estruturados, na qual afirmam que a pesquisa tem mostrado que as necessidades de informações (no caso para engenheiros) normalmente não são atingidas ao retornarmos documentos completos, mas sim apresentando as partes mais relevantes.

Para resolver isso, a tecnologia de recuperação de documentos estruturados se estabeleceu nos últimos anos como um campo ativo de pesquisa e desenvolvimento que se distingue da recuperação de informações tradicional ao utilizar informações tanto da estrutura quanto do conteúdo, permitindo, desta forma, que os usuários recuperem as partes mais relevantes dos documentos. E mais do que isso, a representação adequada da estrutura dos documentos permite que, na visualização do resultados, as partes sejam apresentadas de forma agregada. Ou seja, a utilização da informação estrutural possibilita dois ganhos: maior precisão e maior funcionalidade.

É bem sabido que a busca de informações é um processo vago e impreciso, especialmente em documentos estruturados, pois normalmente são longos e complexos. Uma boa apresentação da informação, com interfaces com o usuário bem desenhadas, é considerada particularmente útil, nesse caso.

A Figura 2.1 representa todos os componentes envolvidos em um processo de consulta a um sistema de recuperação de informações baseado em documentos estruturados. Nela, podem ser observados os momentos de interação entre o usuário e o sistema, tanto pela ótica do usuário quanto pelo ponto de vista do responsável pela implementação.

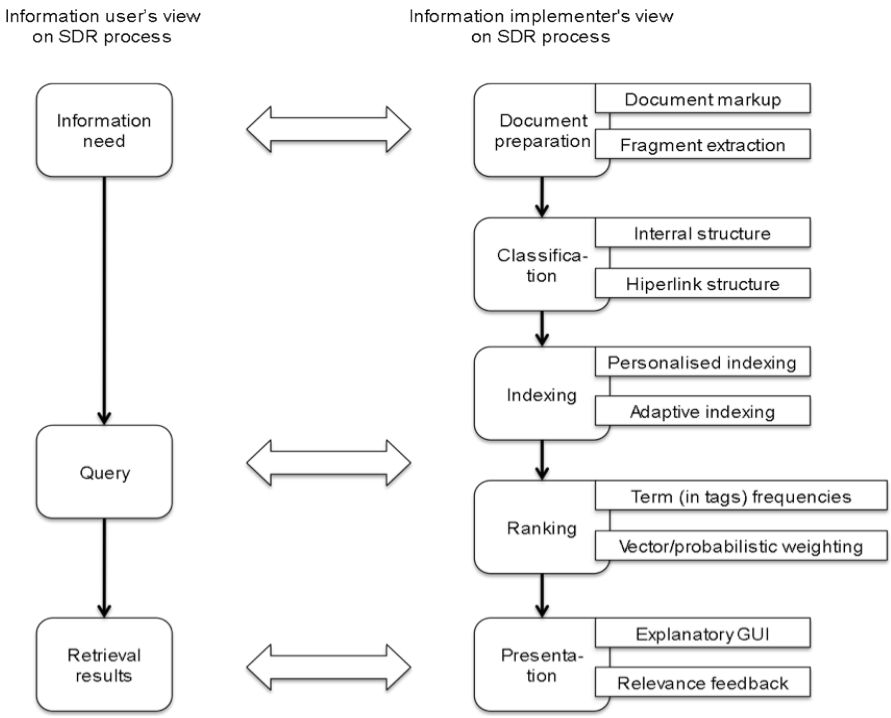

Figura 2.1 – Recuperação de documentos estruturados pelas perspectivas do usuário e do implementador. (Liu et al., 2008)

Ainda de acordo com Liu et al., atualmente os principais interesses da comunidade científica estão voltados para as etapas de classificação dos documentos e de apresentação dos resultados. Comparativamente, o interesse na indexação, busca e avaliação da relevância tem se mostrado menor.

2.2 – Recuperação de Informação

RI é uma área de pesquisa consagrada e possui um sistema teórico relativamente maduro. (ZHANG, 2008, p. 21)

Antes de apresentarmos com mais profundidade as especificidades dos documentos estruturados, é importante que façamos uma revisão dos principais conceitos da área de RI que serão amplamente referenciados nessa dissertação.

2.2.1 – Modelo de Espaço Vetorial

No modelo de espaço vetorial, que foi inicialmente introduzido por Salton (1988), um documento é definido pelo peso de *n* atributos independentes. Na maioria dos casos, esses atributos são denominados "termos" e são obtidos a partir das palavras extraídas do título, resumo ou do texto completo do documento.

$$d_i = (a_{i1}, a_{i2}, ..., a_{ij}, ..., a_{in})$$

Uma matriz de atributos de documentos consiste em um grupo de vetores de documentos, na qual as linhas representam os documentos e as colunas, os atributos:

$$D = \begin{bmatrix} a_{i1} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & a_{ij} & ... \\ a_{m1} & a_{m2} & ... & a_{mn} \end{bmatrix}$$

No modelo vetorial, uma consulta é representada de forma semelhante a um documento, o que possibilita diversas operações matemáticas, como por exemplo o cálculo de similaridade.

$$q = (q_1, q_2, ..., q_j, ..., q_n)$$

Olhando para cada um dos documentos na matriz, observamos que o número de atributos cujo peso é maior que zero é muito pequeno, consequentemente, trata-se de uma matriz esparsa, na qual a maioria dos elementos é zero.

2.2.2 – Métodos para atribuir peso a termos

É de fundamental importância o cálculo automático dos pesos que são atribuídos aos termos, de modo que, ao se realizar uma pesquisa, possa ser computado o peso global de cada documento resultante.

Existem vários fatores que podem ser levados em consideração no cálculo do peso, por exemplo: a frequência de um termo em um documento, o tamanho do documento, a distribuição dos termos em uma coleção de documentos, a posição do termo relativa ao documento como um todo ou a outros termos pesquisados, etc.

2.2.3 – Stop Words

Algumas palavras que são extremamente comuns podem ser desprezadas quando indexamos um conjunto de documentos. Por exemplo, em português podem ser desprezadas as palavras “a”, “o”, “e”, “ou”, etc., pois essas existem em praticamente todos os documentos e dificilmente alguém terá interesse em realizar uma pesquisa envolvendo esses termos.

Em alguns casos, pode ser interessante desprezar determinadas palavras que são

comuns apenas na coleção de documentos em questão. No caso da busca em leis, poderíamos optar por não indexar palavras como: "artigo", "lei", etc. No entanto, se a coleção de documentos se referisse a outro assunto, por exemplo Tecnologia da Informação, essas palavras não seriam tão corriqueiras e deveriam ser mantidas no índice.

2.2.4 – Stemming

Martin Porter recebeu em 2000 o Prêmio Tony Kent Strix por seu trabalho em algoritmos para simplificação de palavras. A ideia é simples: reduzir as palavras apenas aos seus radicais, de modo que todas as palavras que sejam derivadas de um mesmo radical sejam convertidas em um mesmo termo.

Dessa forma, são eliminadas determinadas diferenças que seriam irrelevantes para as necessidades de informação dos usuários. Por exemplo: plurais, como em "lei" e "leis", ou variações de gênero, como em "menino" e "menina", são desprezadas tanto na criação dos índices, quanto por ocasião das consultas.

2.2.5 – Inverso da frequência nos documentos

Karen Spärck Jones (1972) introduziu o método do inverso da frequência nos documentos, que doravante denominaremos de IDF, pois é o acrônimo de *Inverse Document Frequency,* um termo amplamente utilizado na literatura. Desde sua criação, este método tem sido muito utilizado e obtido excelentes resultados, mesmo quando confrontado com estratégias mais cuidadosamente engendradas. A ideia por trás do IDF consiste em privilegiar os termos que são mais raros na coleção, de modo que, se uma pesquisa envolve dois ou mais termos, os documentos que possuírem os termos mais raros serão beneficiados no cálculo da relevância.

Se considerarmos que N é o número total de documentos em uma coleção e que d_i é o número de documentos que possuem o termo "i", então temos que:

$$IDF_i = Log(\frac{N}{d_i})$$

O cálculo final do peso documento em relação a um termo deve incluir também uma parcela referente ao número de ocorrências do termo no documento em questão. Esta frequência de ocorrências – f_i – é conhecida como *TF* (do inglês, *Term. Frequency*).

A fórmula final para o cálculo da relevância, de acordo com o método é expressa assim:

$$peso_i = f_i \times Log(\frac{N}{d_i})$$

2.2.6 – Avaliação dos resultados de RI

Existem diversas abordagens para avaliar a qualidade dos métodos de RI, mas as mais básicas e fundamentais são as avaliações de precisão e cobertura.

Quando um usuário realiza uma busca em uma base de dados de documentos, o sistema retorna uma lista de documentos relevantes. Nessa operação, na verdade, estão sendo definidos quatro grupos distintos de documentos, a saber: relevantes e recuperados (A), irrelevantes porém recuperados (B), relevantes que não foram recuperados (C) e irrelevantes que não foram recuperados (D). Estes grupos estão representados no Quadro 2.1, abaixo:

	Recuperado	Não recuperado
Relevante	A	C
Não relevante	B	D

Quadro 2.1 – O resultado de uma busca divide os documentos em quatro grupos

Logicamente, um bom resultado deve maximizar A e D e minimizar B e C.

A cobertura, que representa a relação entre o número de documentos recuperados e o número total de documentos relevantes, é calculada a partir da seguinte fórmula:

$$Cobertura = \frac{A}{A \cup C}$$

Já a precisão indica a proporção de documentos incorretamente recuperados, e é expressa pela seguinte equação:

$$Precisão = \frac{A}{A \cup B}$$

Na maioria dos casos, a precisão e a cobertura se comportam de forma inversamente proporcional, ou seja, quando um sistema apresenta boa precisão, a cobertura é pequena e vice-versa. Isso decorre do fato de que, para aumentar a

cobertura, geralmente se busca aumentar a quantidade de documentos em "A", e isso faz com que sejam inseridos documentos irrelevantes, o que reduz a precisão.

2.3 – Recuperação de documentos estruturados

Um modelo de recuperação de informações que combine busca textual de conteúdo com indicação de componentes estruturais dos documentos é chamada de *recuperação de documentos estruturados*. (BAEZA-YATES, R., RIBEIRO-NETO, B., 1999, p. 61-66)

Problemas de busca textual, quando envolvem questões que dependem muito fortemente de estrutura, são bem resolvidos por bancos de dados relacionais. No entanto, existem muitos casos que podem ser tratados de modo mais eficiente se forem modelados como recuperação de documentos estruturados. (MANNING C. D. et al., 2008)

Para recuperar informações de forma estruturada, é necessário que se utilize uma linguagem que permita definir uma busca através da combinação de palavras e de componentes estruturais. No entanto, de acordo com Manning et al. (2008), muitas vezes os usuários não sabem o nome que foi escolhido para designar as diferentes estruturas, ou eles podem nem mesmo saber como compor pesquisas estruturadas.

A seleção de um modelo de recuperação de documentos estruturados não é tarefa simples, pois é necessário encontrar um bom equilíbrio entre a eficiência e a capacidade de atender as necessidades específicas da aplicação em questão. No desenvolvimento do Seeklex, diversas adaptações foram realizadas até que pudéssemos encontrar um modelo capaz de atender plenamente.

Os seguintes termos são utilizados para que se possa compreender os modelos de recuperação de documentos estruturados.

- *Resposta*: é uma posição no texto de uma sequência de palavras que satisfaz a busca submetida pelo usuário.
- *Região*: uma porção contígua do texto.
- *Nó*: um componente estrutural do documento que é conhecido tanto pelo autor do texto quanto pelo usuário interessado em realizar a busca. Capítulo, seção, artigo e parágrafo são exemplos de nós encontrados em normas jurídicas.

Os dois modelos mais básicos de recuperação de documentos estruturados são descritos a seguir.

2.3.1 – Modelo baseado em listas não sobrepostas:

Diversas listas de indexação são construídas de modo a refletir, por exemplo, os níveis hierárquicos do texto. Cada elemento de uma lista representa uma região do texto e essas regiões nunca se sobrepõem em uma mesma lista. Mas pode haver sobreposições em se tratando de listas diferentes. Esse modelo, inicialmente proposto por Burkowski (1992), está ilustrado na Figura 2.2, abaixo.

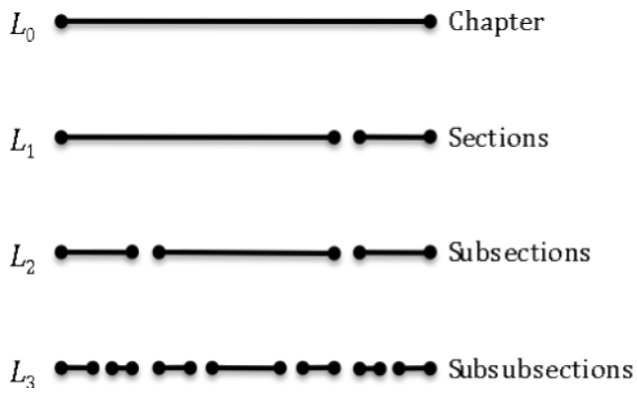

Figura 2.2 – Representação de um documento estruturado através de listas não sobrepostas. (BAEZA-YATES, R., RIBEIRO-NETO, B., 1999, p. 63)

2.3.2 – Modelo baseado em nós próximos

Caso possamos restringir o modelo no sentido de garantir que haverá uma única hierarquia, Baeza-Yates e Navarro (1996, 1997) propuseram utilizar uma estratégia de busca em índices mais otimizada. Um único índice pode mapear todas as ocorrências de cada palavra, enquanto um índice auxiliar identifica os componentes estruturais.

Neste modelo, quando encontramos uma ocorrência, seguimos descendo na hierarquia até encontrar o nó de nível mais baixo onde o termo ocorre. Para procurar a próxima ocorrência, não precisamos recomeçar do topo da hierarquia, podemos apenas verificar se o mesmo nó também possui a próxima ocorrência do termo e assim por diante. Se isso ocorrer, saberemos que todos os componentes estruturais hierarquicamente superiores também são respostas válidas para a consulta. A Figura 2.3, abaixo, representa o índice auxiliar de componentes hierárquicos e o índice único invertido de termos.

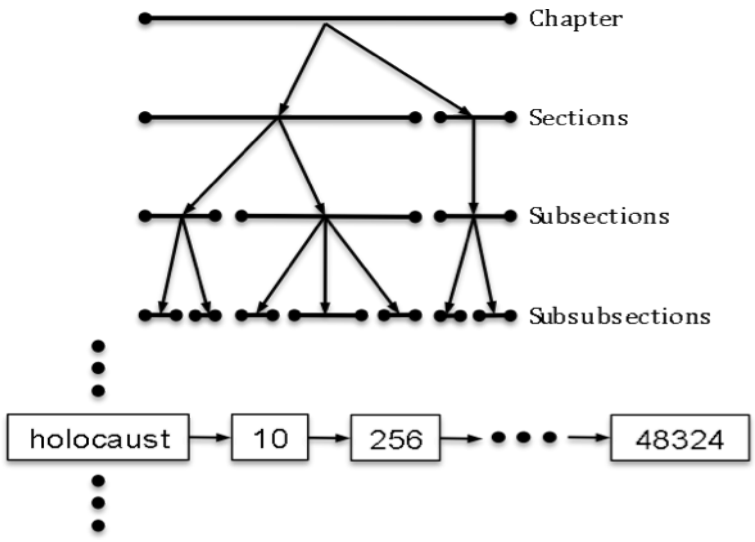

Figura 2.3 – Representação de um índice auxiliar de componentes hierárquicos e de um índice único invertido de termos. (BAEZA-YATES, R., RIBEIRO-NETO, B., 1999, p. 64)

2.3.3 – Modelos de navegação

Geralmente, quando o usuário tem maior clareza sobre o que deseja encontrar, prefere a ferramenta de busca ao processo de navegação.

Quando esse objetivo não está tão claro, o sistema pode propiciar a navegação em algumas modalidades diferentes, como por exemplo: sequencial, hipertextual ou guiada pela estrutura. (BAEZA-YATES, R., RIBEIRO-NETO, B., 1999, p. 66)

No caso das normas jurídicas, a navegação guiada pela estrutura poderia apresentar uma lista de capítulos que, quando clicados, revelariam suas seções e assim por diante até que se alcançasse, no último nível de profundidade, o texto propriamente dito.

2.4 – Recuperação de XML

Os documentos XML são os exemplos mais comuns de documentos estruturados. Portanto, devemos estender nossa revisão da literatura de modo a contemplar, também, os esforços que vêm sendo realizados no sentido de recuperar informações a partir de documentos XML.

Algumas pesquisas não podem ser bem respondidas por sistemas que não considerem critérios de priorização, como por exemplo o TF-IDF. E alguns desafios devem ser enfrentados para que os bons resultados possam ser obtidos na recuperação de informações contidas em documentos XML. O mais complexo deles é que a melhor resposta normalmente não é composta de documentos inteiros, mas sim de partes deles

(MANNING, 2008, p. 147-162). Essas partes são denominadas *elementos* no jargão do XML. Se considerarmos que cada documento XML é composto por uma hierarquia de partes menores, os elementos são os nós dessa árvore.

Um critério comumente utilizado para selecionar a parte mais apropriada de um documento é o Princípio da Recuperação de Documentos Estruturados:

> "**Princípio da Recuperação de Documentos Estruturados**. Um sistema sempre deve recuperar a parte mais específica de um documento que responda à pesquisa." (FUHR, N., GROßJOHANN, K., 2001)

Este princípio foi extraído de um trabalho anterior que ressaltava a importância de restringir o resultado quando os documentos são muito longos:

> "Nós acreditamos que o processo de recuperação, no contexto de grandes quantidades de informação estruturada, deve se focar nas menores unidades (i.e. de mais baixo nível na estrutura lógica) que satisfaçam a consulta." (CHIARAMELLA, Y., MULHEM, P., FOUREL, F., 1996)

Considere a consulta pelo termo `"direito"` aplicada ao fragmento de XML da Figura 2.4, abaixo. A palavra "direito" ocorre nos textos do "Título II" e do "Capítulo I", bem como no conteúdo do primeiro inciso. Todos esses elementos são resultados possíveis. No entanto, neste caso, o "Título II" – o nó de mais alto nível – é a escolha mais adequada. Decidir qual nível da árvore é a melhor resposta para a consulta não é tarefa fácil.

```
<Norma titulo="Constituição Federal de 1988">
      <Titulo rotulo="TÍTULO II" texto="Dos Direitos e Garantias Fundamentais">
            <Capitulo rotulo="CAPÍTULO I"
                  texto="Dos Direitos e Deveres Individuais e Coletivos">
                  <Artigo rotulo="Art. 5°">
                        <Caput>Todos são iguais perante a lei, sem distinção de
                                    qualquer natureza, garantindo-se aos brasileiros e
                                    aos estrangeiros residentes no País a
                                    inviolabilidade do direito à vida, à liberdade, à
                                    igualdade, à segurança e à propriedade, nos termos
                                    seguintes:</Caput>
                        <Inciso rotulo="I - ">homens e mulheres são iguais em
                                    direitos e obrigações, nos termos desta
                                    Constituição;</Inciso>
                  </Artigo>
            </Capitulo>
      </Titulo>
</Norma>
```

Figura 2.4 – Fragmento de XML representando uma norma jurídica.

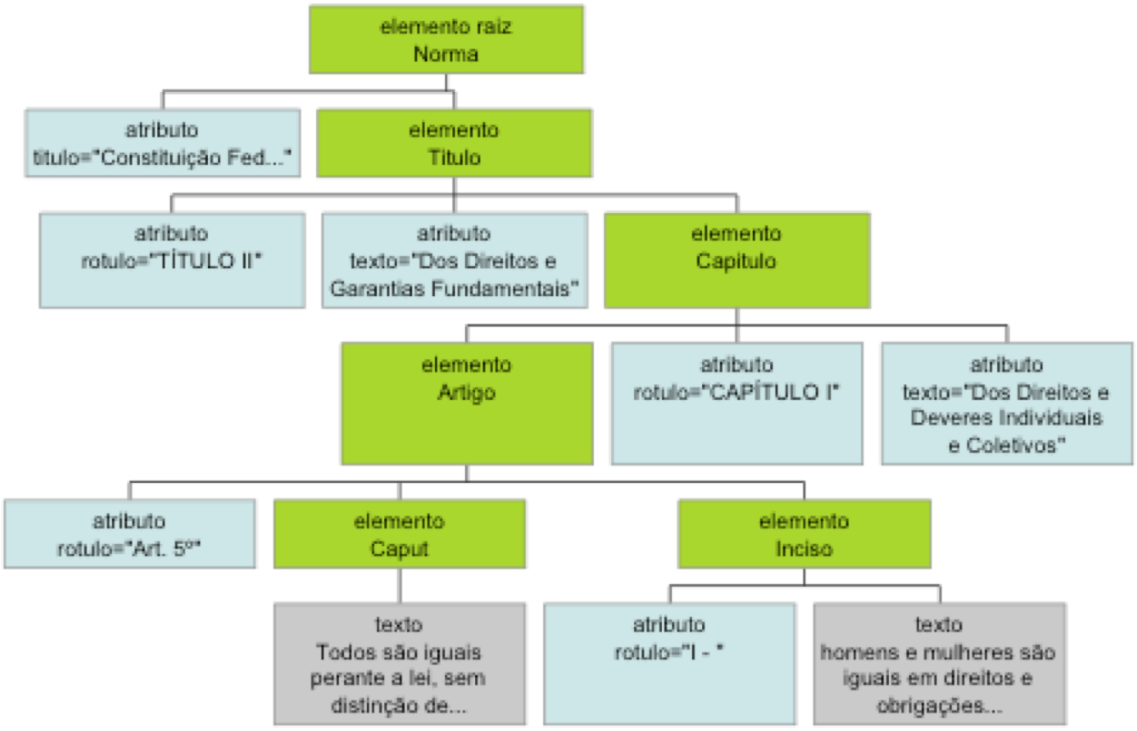

Figura 2.5 – O documento da figura anterior representado na forma de uma árvore de elementos, atributos e textos.

Muitas vezes, os resultados de consultas estruturadas ainda conterão elementos de diferentes níveis dentro de uma mesma linha hierárquica. Sendo assim, surge a necessidade de realizar uma etapa de pós-processamento, onde poderão ser eliminadas algumas redundâncias. A contextualização também é importante para a compreensão dos resultados:

> "Esse contexto pode ser útil na interpretação do parágrafo (e.g., a fonte da informação reportada) mesmo se o parágrafo sozinho satisfizer a pesquisa." (MANNING, 2008, p. 186)

Outra importante estratégia para facilitar a compreensão dos resultados por parte do usuário consiste em aplicar um destaque nos termos que foram utilizados na pesquisa. Dessa forma, o usuário conseguirá varrer um texto de maior tamanho, como por exemplo um artigo, em um tempo pouco superior ao que antes era necessário para varrer um parágrafo. (MANNING, 2008, p. 186)

Além da questão de quais partes dos documentos devem ser apresentadas ao usuário, também existe a necessidade de definir quais partes serão indexadas e de qual maneira.

Outro grande desafio na recuperação de XML é a eventual necessidade de

calcular diferentes valores de IDF, em função dos diferentes contextos onde os textos podem estar inseridos.

2.4.1 – Avaliação da recuperação de XML

A *INEX* (**IN***itiative for the* **E***valuation of* **X***ML retrieval*) (SHLOMO G. et al., 2009) tem sido o principal programa destinado à avaliação de recuperação de XML. Esse esforço colaborativo já produziu diversas coleções de testes e também critérios de julgamento.

Em 2002, a INEX definiu que são necessárias duas dimensões ortogonais para calcular a relevância dos elementos. A primeira, no Quadro 2.2, abaixo, foi designada cobertura do componente e representa a grau de precisão na seleção da granulosidade: o elemento não deve ser nem muito abrangente nem demasiadamente específico.

Valor	Símbolo	Descrição
Cobertura Exata	E	A informação desejada é o principal tópico do componente, e ele é uma unidade de informação auto-contida.
Muito Pequeno	S	A informação desejada é o principal tópico do componente, mas ele não tem sentido por si só.
Muito Grande	L	A informação desejada está presente no componente, mas não é seu principal tópico
Sem Cobertura	N	A informação desejada não é um tópico do componente

Quadro 2.2 – INEX 2002 - Dimensão de cobertura de componente.

A segunda dimensão ortogonal definida pela INEX 2002 foi denominada relevância do tópico e também foi classificada em quatro diferentes níveis:

Valor	Símbolo
Altamente Relevante	3
Bastante Relevante	2
Marginalmente Relevante	1

Não Relevante	0

Quadro 2.3 – INEX 2002 - Dimensão de relevância do tópico.

As combinações entre coberturas de componentes e relevâncias de tópicos foram quantificadas da seguinte maneira:

$$Q(rel,cob)=\begin{cases} 1.00 & se \quad (rel,cob) = 3E \\ 0,75 & se \quad (rel,cob) \in \{2E,3L\} \\ 0,50 & se \quad (rel,cob) \in \{1E,2L,2S\} \\ 0,25 & se \quad (rel,cob) \in \{1S,1L\} \\ 0,00 & se \quad (rel,cob) = 0N \end{cases}$$

Na recuperação de XML, a efetividade geralmente é mais baixa do que na recuperação de documentos não estruturados, isso porque, além de encontrar o documento, também é necessário identificar qual a parte mais relevante para a consulta em questão. (MANNING, 2008, p. 195)

Ao se impor às consultas restrições estruturais adicionais, os resultados encontrados tendem a ser mais relevantes e precisos, no entanto, os elementos que não passarem pelo filtro estrutural serão omitidos, reduzindo, consequentemente, a completude do resultado. (MANNING, 2008, p. 196)

2.4.2 – Recuperação centrada em texto ou em dados

As consultas podem ser divididas em dois grupos principais: as que são fortemente textuais e as seleções de dados (FUHR, N., GROßJOHANN, K. 2001):

- **Textuais**: são caracterizadas por (i) longos campos de texto (e.g., seções de um documento), (ii) correspondência inexata entre a consulta e os resultados, e (iii) resultados classificados por relevância. Bancos de dados relacionais não são adequados para lidar com esse tipo de consulta.
- **Dados**: são caracterizadas por (i) lidar basicamente com dados numéricos ou outros tipos de atributos não textuais, e (ii) na maioria dos casos, a seleção de resultados é realizada através de uma adequação exata às restrições impostas pela consulta.

Existem poderosas linguagens de consulta para XML que são capazes de lidar com restrições de atributos, junções e ordenação. Dentre elas, a que mais se destaca é XQuery, uma linguagem padronizada pelo W3C e que foi projetada para ser aplicada

em todas as áreas onde o XML é utilizado.

Recentemente, muitos esforços vêm sendo envidados no desenvolvimento de sistemas de recuperação de informações com classificação por relevância baseados em XQuery. No entanto, por se tratar de uma linguagem complexa, é um grande desafio obter níveis de desempenho comparáveis aos atualmente encontrados nas soluções existentes para recuperação de informações não estruturadas. (MANNING, 2008, p. 197)

2.4.3 – Diferentes abordagens para a recuperação de XML

Os participantes da INEX desempenham e avaliam regularmente quatro diferentes tarefas relacionadas à recuperação de informações em documentos XML. Estas tarefas simbolizam as necessidades de informações presentes em diferentes cenários que refletem suposições a respeito de como os resultados devem ser apresentados e de quais são os objetivos específicos dos usuários. São elas: (WOODLEY, 2007)

- **Recuperação Completa**: a principal estratégia de recuperação em XML consiste na capacidade de estimar a relevância de elementos ou passagens potencialmente recuperáveis de uma coleção. Portanto, a Recuperação Completa simplesmente avalia o quão bem os sistemas conseguem retornar elementos ou passagens ordenados pela relevância em relação à consulta. Como os resultados geralmente seriam ainda pós-processados, não são realizadas suposições a respeito da apresentação do resultado ou da interação com o usuário.
- **Recuperação Focalizada**: O cenário imaginado para essa tarefa é retornar, ao usuário, uma lista de elementos ou passagens para o tópico pesquisado. Nesta lista, apenas os resultados mais bem "focalizados" devem ser retornados, sem que haja sobreposição de elementos (os menores são preferidos no caso de haver igualdade de relevância). Uma vez que os elementos de nível mais alto na hierarquia, e também as passagens mais longas, sempre são relevantes, é um grande desafio escolher a granularidade correta. Para essa tarefa, algumas suposições são feitas: os resultados são apresentados na forma de uma lista ordenada, que é examinada pelos usuários de cima para baixo e item a item.
- **Relevante no Contexto**: o cenário vislumbrado para essa tarefa consiste no retorno de uma lista ordenada de artigos e, dentro desses artigos, as informações relevantes (capturadas por um conjunto de elementos ou passagens não

sobrepostos). Um artigo relevante provavelmente terá informações relevantes distribuídas em diferentes elementos. Essa tarefa requer que os sistemas encontrem os resultados correspondentes às informações relevantes dentro de cada artigo recuperado. Nessa tarefa, são levadas em consideração as seguintes suposições: (i) os resultados apresentados devem ser agrupados por artigo, na ordem original do documento, e o acesso será disponibilizado, posteriormente, por meio de navegação; (ii) os usuários consideram que o artigo seja a unidade de recuperação de informação mais natural, e preferem que a visão geral de relevância seja realizada nesse contexto.

- **Melhor no Contexto**: O cenário subjacente à esta tarefa é o retorno de uma lista ordenada de artigos e da identificação no melhor ponto de entrada a partir do qual o usuário deve iniciar a leitura desses artigos de modo a satisfazer sua necessidade de informação. Mesmo que um artigo seja completamente dedicado ao tópico solicitado, sempre haverá um melhor ponto para se iniciar a leitura (ainda que esse ponto seja o início do artigo). Essa tarefa compreende as seguintes suposições: (i) apenas um resultado será apresentado por artigo; (ii) os usuários consideram que os artigos são a unidade mais natural para a recuperação da informação, mas preferem ser guiados ao ponto que propicie iniciar a leitura do artigo pelo conteúdo mais relevante.

2.5 – Recuperação focalizada

Em um documento, podemos identificar tipicamente três tipos de relações estruturais: (i) hierarquia, (ii) sequência e (iii) hiperlink, conforme pode ser visto na figura abaixo:

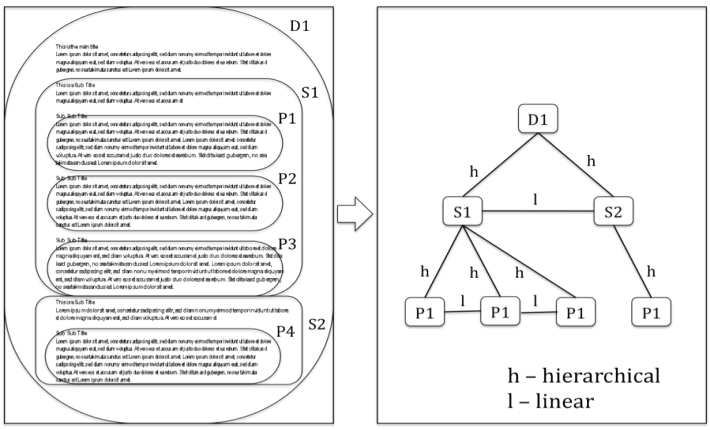

Figura 2.6 – Estrutura de um documento e visão esquemática. (KAZAI, G. et al., 2001, p. 124)

Se indexarmos as páginas web ou outros tipos de documentos estruturados baseados no conhecimento combinado de estrutura e conteúdo, aumentaremos a qualidade dos resultados obtidos. Além disso, a combinação dos dois possibilita a recuperação de partes relevantes de documentos de granulosidade variável: em alguns casos apenas um elemento do documento pode ser retornado, em outros, o resultado poderá ser composto por um grupo de elementos ou até por um documento inteiro. (KAZAI, G. et al. 2001)

O conhecimento da estrutura também pode ser levado em conta na hora de apresentar os resultados para o usuário. Os mecanismos clássicos de recuperação de informação geralmente apresentam seus resultados na forma de uma lista, ordenada por relevância, de ponteiros para páginas web ou documentos. No caso da recuperação de documentos estruturados, os ponteiros não apontam para documentos inteiros, mas sim para partes deles.

Os objetos resultantes de uma busca – páginas web ou partes de documentos –, mesmo quando possuem forte relação estrutural, podem estar em posições bem diferentes na lista em função de sua relevância para a consulta em questão. Essa desorganização dificulta a compreensão do resultado, gastando tempo do usuário e causando desorientação (CHIARAMELA, Y. 1997), principalmente quando se trata de grandes coleções de documentos (KAZAI, G. et al., 2001, p. 123).

> "Por exemplo, suponha que dois nós de um documento "c" e "s" tenham sido recuperados em uma consulta, sendo que "s" é uma seção e "c" um capítulo. Classicamente, essa informação não seria explicitada ao usuário até que este navegasse de "c" para baixo ou de "s" para cima. Além disso, dependendo do método de classificação, esses dois nós poderiam ser apresentados em posições distantes no resultado." (KAZAI, G. et al., 2001, p. 123)

O conhecimento da estrutura pode facilitar a visualização do resultado na medida em que essas relações podem ser explicitadas. O método que foi adotado pelo Google consiste em agrupar objetos relacionados em uma sublista. Outra alternativa possível é a "recuperação focalizada", que consiste na identificação dos melhores pontos de entrada (LALMAS, M. 1997, CHIARAMELA, Y. et al., 1996). Esses pontos correspondem a ponteiros para partes relevantes de documentos a partir das quais os usuários podem facilmente navegar para outras partes relevantes. Dessa forma, ao identificar os melhores pontos de entrada, o sistema não está levando em consideração apenas a relevância, mas também as conexões que refletem as relações entre os diversos

objetos encontrados. Ou seja, está combinando os paradigmas de navegação e recuperação para identificar os melhores pontos de entrada em um documento estruturado (KAZAI, G. et al., 2001, p. 123).

2.6 – XQuery

Embora ainda não haja consenso sobre quais os melhores métodos para recuperação de documentos estruturados, muitos pesquisadores acreditam que XQuery será a linguagem padrão para este tipo de consulta. (MANNING, 2008, p. 179)

No entanto, XQuery – em sua versão 1.0 – não é uma linguagem capaz de realizar consultas textuais, pois não suporta os conceitos de correspondência inexata e classificação por relevância apresentados anteriormente.

Uma proposta para expandir a linguagem e acrescentar os requisitos básicos de RI está sendo gestada no W3C. Trata-se da recomendação candidata de "XQuery and XPath Full Text 1.0". Embora a recomendação ainda tenha o status de "candidata", algumas implementações já existem, mesmo que não sejam totalmente completas (BANFORD, R. et al., 2009). Esta nova versão de XQuery é bastante versátil e, como possui os requisitos básicos de RI, merece ser cuidadosamente avaliada no contexto desse trabalho.

As extensões para realização de buscas textuais se encaixam no modelo tradicional de FLWOR (For, Let, Where, Order By, Result) da linguagem. Foi proposto um novo tipo de restrição que se baseia nos conceitos de RI, que está ilustrado na Figura 2.7, abaixo:

```
for $b in /livros/livro
where $b/titulo contains text "cachorro" ftand "gato"
return $b/autor
```

Figura 2.7 – Busca textual realizada em um conjunto de livros: o título do livro deve conter as palavras "cachorro" e "gato".

O mesmo exemplo, escrito na forma de XPath, pode ser visto na Figura 2.8, abaixo:

```
/livros/livro[titulo contains text "cachorro" ftand "gato"]/autor
```

Figura 2.8 – Busca textual utilizando XPath Full Text.

Por ocasião da elaboração da presente dissertação, não tivemos acesso a um processador XQuery capaz de computar o *score* dos resultados, no entanto, a sintaxe para esse tipo de busca já está definida e pode ser observada na Figura 2.9, abaixo:

```
for $b scores $s in /livros/livro[titulo contains text "cachorro" ftand
"gato"]
order by $s descending
return Titulo: {$b//titulo}, Score: {$s}
```

Figura 2.9 – Busca textual ordenada pelo *score*.

Caso existissem implementações robustas de XQuery and XPath Full Text, poderíamos nos valer de sua capacidade para agregar automaticamente a um elemento o texto de seus sub-elementos, evitando parte dos trabalhos de agregação que realizamos em nosso estudo de caso, como será visto mais adiante.

Entretanto, por mais flexível que seja o modelo de consulta proposto por XQuery, ainda não é suficiente para resolver todos os problemas impostos pelo contexto de buscas em normas jurídicas, pois a maior parte das dificuldades não reside em agregar textos, mas sim em omitir resultados desnecessários, recompor a árvore, ordenar corretamente e apresentar sub-elementos relevantes. Ou seja, embora XQuery possa realizar a primeira etapa de busca, os tópicos por ela retornados ainda necessitam de um pesado pós processamento antes de serem apresentados.

Para melhor compreender as características da busca textual da linguagem XQuery, realizamos uma pesquisa na Constituição Federal pelos termos "`garantia propriedade herança`", a consulta formulada pode ser vista na Figura 2.10, abaixo:

```
for $m in fn:doc('cf1988s.xml')//*[. ftcontains 'garantia' ftor 'herança'
ftor 'propriedade']
  return $m
```

Figura 2.10 – Busca textual em norma jurídica utilizando XQuery and XPath Full Text 1.0.

Os resultados obtidos vieram na forma de uma lista plana e foram demasiadamente extensos para serem incluídos nessa dissertação, mas podem ser explicados da seguinte forma: primeiramente foi retornado o elemento raiz da Constituição, ou seja, o conteúdo completo, em seguida, obtivemos o "Título II – Dos Direitos e Garantias Individuais", o "Capítulo I – Dos Direitos e Deveres Individuais e Coletivos", o Artigo 5º e assim por diante, até que cada um dos elementos que contém

pelo menos uma das três palavras fosse listado. Dentro de cada nível hierárquico, a ordenação natural do documento foi mantida, pois a única implementação da linguagem que pudemos utilizar (MXQuery 0.6, 2010) ainda não contemplava a ordenação por pesos.

Este resultado, embora impressionante para uma consulta descrita de forma tão simples, ainda está muito distante do que consideramos ideal para as necessidades de informação dos usuários do meio jurídico, como veremos no capítulo seguinte.

2.7 – Visualização de recuperação de subdocumentos

A área de visualização de informações é um campo emergente cujo objetivo principal é a disposição espacial da informação para que os usuários possam interagir com ela.

RI é um campo de pesquisa importante que já existe há bastante tempo. RI se refere ao processo de busca, exploração e descoberta de informações contidas em repositórios organizados para satisfazer as necessidades de usuários. RI contém dois componentes principais: recuperação da informação e organização da informação, que são como dois lados de uma mesma moeda. Embora a organização não seja percebida pelos usuários, pela perspectiva do sistema, ela é totalmente indispensável e é ela quem determina os caminhos e métodos da recuperação. (ZHANG, 2008, p. 4)

Na RI convencional as relações entre os resultados encontrados são raramente apresentadas, e os ambientes de recuperação não possuem os mecanismos interativos necessários para a navegação. (ZHANG, 2008, p. 1)

Existem dois paradigmas que são amplamente reconhecidos para RI: navegação e busca textual. Esses paradigmas refletem dois tipos básicos de comportamentos em relação à procura de informações. Cada um deles têm seus pontos fortes e fracos, e são complementares.

A busca textual é uma tarefa complexa que envolve a tradução de uma necessidade de informação em um conjunto de palavras relevantes. Após o processamento da consulta, uma lista de resultados, ordenados por relevância, é retornada ao usuário. Um ponto fraco da busca textual é que nem sempre o vocabulário escolhido pelo usuário é o mesmo que é utilizado na base de dados; quando isso ocorre, o fracasso é inevitável.

A navegação se refere a ver, folhear, e examinar a informação de acordo com as possibilidades do ambiente. Navegar é uma maneira extremamente importante de

explorar e descobrir informações. De acordo com Zhang (2008, p. 5), um ambiente bem organizado de apresentação e interação é o que garante uma boa experiência de navegação. Os métodos mais utilizados de organização para navegação são estruturas de hiperlinks e estruturas hierárquicas, no entanto, embora essas estruturas sejam muito superiores a listas lineares, outros métodos podem viabilizar patamares ainda melhores de navegação.

A Figura 2.11, abaixo, mostra que a navegação, diferentemente da busca textual, além de influenciar a seleção de itens/objetos a serem exibidos, pode interagir diretamente com a camada de organização.

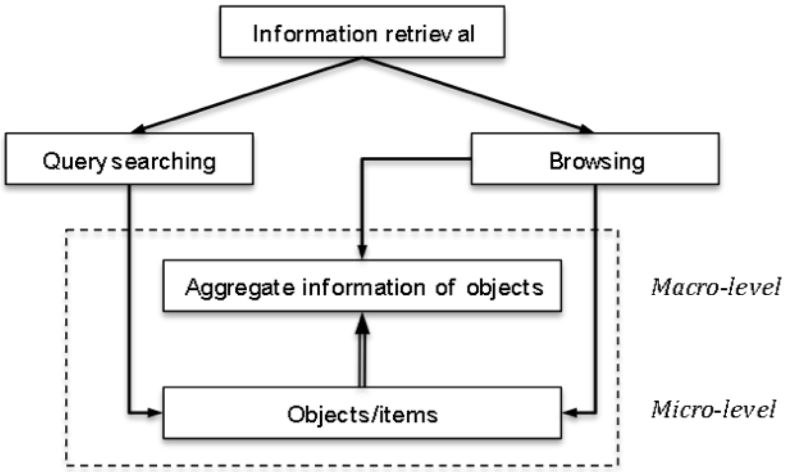

Figura 2.11 – RI e informação em dois níveis. (ZHANG, 2008, p. 9)

Abaixo, no Quadro 2.4, apresentamos as diferenças entre a busca textual e a navegação em relação a diversos aspectos relacionados à recuperação de informações:

Aspecto	Busca Textual	Navegação
Julgamento de relevância	Realizado automaticamente pelo sistema em função dos termos presentes na consulta	Julgado pelo usuário, num processo que envolve conceitos
Continuidade	Descontínuo, pois o usuário só volta a ter controle sobre o processo depois que o sistema apresenta o resultado	Contínuo, já que cada passo (seleção, exame, julgamento de relevância) é controlado pelo usuário
Gasto de tempo e esforço	Eficiente, uma vez que a busca se completa em poucos passos	Menos eficiente, pois o usuário precisa digerir o conteúdo e realizar escolhas

Comportamento	O usuário especifica sua necessidade de informação, portanto requer que o usuário saiba precisamente o que está buscando	O usuário aprende o que o sistema tem a oferecer, sendo mais apropriado quando o usuário ainda não formulou precisamente sua necessidade de informação ou quando está se deparando com um novo assunto

Quadro 2.4 – Diferenças entre busca textual e navegação (ZHANG, 2008, p. 5-7).

Existem três paradigmas para a visualização de RI: (i) QB: busca textual seguida de navegação; (ii) BQ: navegação seguida de busca textual, nessa modalidade o sistema realça as palavras buscadas ou apresenta resultados restritos ao contexto da navegação; e (iii) BO: somente navegação. A busca textual simples não é considerada um paradigma de visualização de RI pois não requer um espaço visual sofisticado. (ZHANG, 2008, p. 16)

Estes três modelos de visualização de RI apresentados acima – QB, BQ e BO – foram contemplados no desenvolvimento do Seeklex, conforme veremos mais à frente.

2.8 – Critérios de avaliação para visualização em RI

A avaliação dos sistemas sempre vem depois da modelagem e da implementação, talvez por isso ainda não sejam muito desenvolvidas as técnicas para a avaliação da visualização em sistemas de RI. No entanto, conforme a disciplina vai atingindo sua maturidade, e vão surgindo diversas aplicações comerciais, a avaliação de sistemas e modelos se torna mandatória.

Diferentemente da visualização científica, na visualização de RI não existe uma estrutura física bem definida para ser apresentada em um espaço visual. Em decorrência disto, diversos modelos podem ser produzidos para refletir e revelar as relações semânticas entre os diferentes objetos que habitam as bases de dados de sistemas de RI. Essa diversidade acaba por complicar ainda mais a avaliação dos modelos de visualização, pois falta um padrão a partir do qual se possa fazer comparações. (ZHANG, 2008, p. 239)

Em um sistema de RI tradicional, usuários geralmente digitam algumas palavras para compor uma consulta, e recebem o resultado na forma de uma simples lista de documentos. No caso dos ambientes de visualização, os usuários precisam compreender

os conceitos, ícones e metáforas para interpretar os resultados apresentados e possuir meios de manipular o espaço visual para interagir com o sistema. Ou seja, o processo de pesquisa em um ambiente de visualização é mais complexo do que em um sistema tradicional de RI e, consequentemente, a avaliação destes ambientes também é mais difícil. (ZHANG, 2008, p. 240)

Ainda segundo Zhang (2008, p. 242), desenvolver critérios de avaliação de visualização para RI, que sejam amplamente aceitos, é um tópico de pesquisa importante e urgente.

2.8.1 – Fatores que afetam os padrões de avaliação

Está claro que um único modelo de visualização não se presta a resolver todos os tipos de problemas (ZHANG, 2008, p. 243, ipud GRINSTEIN et al., 2005). Sendo assim, os sistemas de visualização geralmente são construídos de modo a endereçar um problema específico, e só podem ser mensurados no contexto para o qual foram concebidos.

Um segundo fator que deve ser considerado é que a interatividade, que é um ponto forte dos sistemas de visualização, torna mais complexa a avaliação do desempenho destes sistemas.

Um terceiro fator se relaciona às questões de habilidades de orientação espacial, percepção e as funções cognitivas dos usuários do sistemas, pois o processo de RI em um ambiente visual é uma experiência complexa de julgamento de relevância e tomada de decisão.

Por fim, como já foi citado anteriormente, também devemos considerar a diversidade de modelos e ferramentas existentes.

Critérios de avaliação devem ser compreensíveis e abranger todos os aspectos dos ambientes de visualização de RI, que vão desde a representação visual da informação até o controle da interação do usuário com o sistema, passando pelos mecanismos de busca textual e pelos modelos de navegação.

2.8.2 – Principais critérios de avaliação

Dentre os critérios presentes na literatura científica, elencamos no Quadro 2.5, abaixo, alguns que são particularmente relevantes para a presente dissertação e que foram respeitados na implementação de nosso estudo de caso:

Aspecto	Critério	Descrição
Navegação	Orientação	Os usuários precisam ser guiados durante a navegação. Isso pode ocorrer na forma de uma indicação hierárquica de localização na estrutura. Para não confundir o usuário, em cada ponto devem ser apresentadas apenas as opções de interação cabíveis.
	Exploração	Os usuários devem ser capazes de navegar em todo o espaço de informações a partir de um ponto de partida único. Informações mais detalhadas de um objeto devem ser apresentadas quando o objeto for selecionado.
	Conexões	Quando um objeto é apresentado no espaço visual, suas relações com outros objetos também devem ser explicitadas.
Busca textual	Simplicidade	Assim como os sistemas de RI, ambientes de visualização devem permitir buscas textuais simples.
	Reformulação	O processo de busca é contínuo, de modo que deve ser permitido ao usuário reformular suas consultas após visualizar resultados iniciais.
Representação visual da informação	Intuitividade	A informação deve ser apresentada de forma organizada e fácil de compreender. Objetos e conceitos familiares facilitam a compreensão, reduzindo o tempo de aprendizagem e a ansiedade dos usuários.
Controle da interação	*Zoom*	O sistema deve permitir que o usuário defina o nível de detalhe a ser visualizado, e sempre deve poder fazer *zoom in/out* nos objetos que desejar.
	Histórico	A exploração por vezes é um processo de tentativa e erro, portanto o usuário deve ser capaz de voltar facilmente a um ponto anterior.

Quadro 2.5 – Principais critérios de avaliação para visualização de RI. (ZHANG, 2008, p.246)

Na prática, devido à natureza da visualização de RI, é extremamente difícil encontrar critérios mensuráveis para cada um dos quesitos que precisam ser avaliados, (ZHANG, 2008, p. 244) de modo que, atualmente não existem critérios amplamente utilizados, confiáveis e precisos para avaliação de visualização de RI. (ZHANG, 2008, p. 253)

2.9 – Métodos computacionais aplicados a normas jurídicas

Anselmo Maciel Nunes e Renato Feito (2007) propuseram uma arquitetura para elevar a precisão e a cobertura de consultas à jurisprudência que se baseia na utilização de ontologias e na marcação semântica dos documentos. Na mesma linha, Cláudio Gottschalg-Duque (2006) desenvolveu um experimento interessante, que embora não esteja diretamente ligado à área do direito, poderia ser facilmente adaptado. A proposta consiste em um sistema de indexação baseado em Processamento de Linguagem Natural (PLN). Após a identificação morfológica, sintática e semântica, uma ontologia-leve é automaticamente produzida e indexada para posterior consulta. Os resultados obtidos foram significativamente melhores se comparados à simples aplicação do modelo vetorial.

Berthier Ribeiro Neto e Rodrigo Torres Assumpção (2001) utilizaram o tesauro elaborado pelo Conselho de Justiça Federal (CJF) e uma rede Bayesiana para melhorar a precisão de consultas em jurisprudências. Para cada jurisprudência, trabalharam com os textos contidos na ementa, na indexação e no acórdão, no entanto, não endereçaram a questão de documentos longos que precisam ser divididos.

Em um trabalho correlato, porém voltado para a área médica, melhores índices de precisão e cobertura foram obtidos utilizando uma rede Bayesiana de crenças para adicionar ao modelo vetorial de *ranking* informações sobre a categorização de doenças de acordo com a Classificação Internacional de Doenças (CID) da Organização Mundial de Saúde (OMS) (VALE, R. F. et al. 2001).

Hermes R. Freitas-Junior et al. (2006) realizaram outro estudo voltado para a área da medicina, no qual utilizaram técnicas de RI em diferentes línguas (inglês, português e espanhol) combinadas a um vocabulário controlado denominado Medical Subject Headings. Os resultados obtidos para consultas em diversas línguas aplicadas a um corpus em inglês foram semelhantes aos resultados do modelo vetorial trabalhando com consultas somente em inglês.

Em sua dissertação de mestrado, Tânia Cristina D'Agostini Bueno propôs um

modelo para recuperação de jurisprudência que se vale da técnica de IA conhecida como Raciocínio Baseado em Casos (RBC). Sua abordagem é refinada por uma cuidadosa compreensão da teoria jurídica, pelo uso de um vocabulário controlado e também de um dicionário de termos normativos.

Além do exposto acima, também encontramos, na literatura científica, outras referências aplicadas diretamente à ordem legal, em geral métodos de classificação e agrupamento de documentos em bases de leis e casos (ROSE & BELEW, 1989; BIAGIOLI ET AL., 2005) ou mesmo para análise semântica e extração de informações específicas do domínio legal (BARON & THOMPSON, 2007; FRANCESCONI & PERUGINELLI, 2007; MOENS, 2005). Existem também abordagens para problemas específicos de sistemas legais locais como, por exemplo, a extração de informação se aproveitando da característica multilíngue de algumas regiões, e.g. Suíça e Canadá (SHERIDAN, BRASCHLER & SCHÄUBLE, 1997).

Não foram encontrados trabalhos correlatos, aplicados diretamente ao ordenamento jurídico, que tivessem foco na busca em subdocumentos ou na organização dos resultados das consultas para o fácil acesso à informação desejada. Aparentemente, a recuperação de subdocumentos é, hoje em dia, uma técnica mais utilizada no âmbito da Engenharia (LIU et al. 2008) do que no do Direito.

2.10 – LexML

O portal LexML é uma iniciativa conjunta de diversos órgãos participantes do GT LexML da Comunidade TIControle, liderada pelo Senado Federal.

Os objetivos do LexML são identificar e estruturar as informações legislativas e jurídicas através da integração de processos de trabalho e compartilhamento de dados utilizando padrões abertos, nas três esferas administrativas (federal, estadual e municipal) e entre os órgãos dos três poderes da República (Executivo, Judiciário e Legislativo), por meio de hiperlinks persistentes, sistemas online e tratamento padronizado da estrutura textual. Portanto, o LexML não é um sistema de busca textual no conteúdo das leis, mas sim em seus metadados.

No LexML, cada documento legislativo e jurídico possui um identificador unívoco e persistente (chamado URN), que pode ser referenciado sem o temor de que o endereço seja alterado no futuro. Os tipos de referências que comumente são realizadas entre normas legais são:

Dependências diretas: algumas normas citam outras para revogá-las ou detalhá-

las. Conjuntos de normas assim relacionadas formam uma unidade coesa. Essas dependências também são facilmente registradas em banco de dados, e dão origem ao chamado "texto compilado" da norma original.

Dependências hierárquicas: usualmente normas mais específicas e detalhadas regulamentam as normas mais gerais. Essa é uma relação típica entre, por exemplo, as Leis Federais e a Constituição da República. A hierarquia, num banco de dados, permite organizar de forma mais coerente grandes quantidades de normas.

Vinculação por assunto: apesar de ser um dos aspectos mais difíceis de se estabelecer pela via da automação, o agrupamento de normas em função do assunto que tratam é de grande importância. Metodologias da Biblioteconomia e da Jurisprudência garantem a confiabilidade de procedimentos computacionais assistidos por pessoas especializadas.

Relações espaciais e temporais: para responder a perguntas como "Em quais locais do mapa do Brasil vigoram as normas?", ou "A quais normas do ordenamento jurídico federal, estadual e municipal um cidadão está submetido?", é necessário manter-se o registro de relações geográficas e temporais.

Boa parte dessas informações se encontram explicitadas no texto da norma. O Projeto LexML estabeleceu tecnologias e recomendações para o registro de citações que permitem a criação de hiperlinks persistentes, o estabelecimento de relacionamentos semânticos entre documentos e a semi-automação da geração de texto compilado.

Nosso estudo de caso, o Seeklex, foi desenvolvido em colaboração com membros do Senado Federal, que nos forneceram ferramentas e documentos estruturados para os testes iniciais de nosso mecanismo de buscas. Por serem soluções complementares, existe espaço para a integração futura entre o Seeklex e o portal LexML.

2.11 – Considerações finais

Neste capítulo realizamos a revisão da literatura sobre recuperação de informações e, em especial, sobre os estudos envolvendo documentos estruturados, suas partes e os vínculos hierárquicos entre elas.

Também deixamos claro que a maior parte dos estudos envolvendo RI e documentos estruturados têm ocorrido em função da necessidade de realizar buscas em documentos XML, e que o veículo que centraliza esses esforços é a INEX.

Os diversos cenários em que uma busca pode ser realizada são representados

pelas quatro tarefas comumente propostas pela INEX. Fazendo uma adequação às necessidades de informação presentes em uma busca por normas jurídicas, acreditamos que o melhor caminho consiste em combinar conceitos da Recuperação Completa com estratégias de organização e visualização propostas na modalidade Relevante no Contexto. Dessa forma, podemos apresentar todas as ocorrências encontradas (não apenas as mais relevantes) e, simultaneamente, manter a relação estrutural original da informação, facilitando assim a compreensão dos resultados por parte dos usuários.

No entanto, para realizar a combinação das modalidades descritas acima, torna-se necessário criar estratégias e algoritmos capazes de executar corretamente as buscas, reorganizar estruturalmente os resultados e propiciar uma visualização consistente. O próximo capítulo apresentará os esforços que foram conduzidos nesse sentido.

Nossa revisão literária também apontou diversas questões relativas à interação entre o usuário e o sistema de busca, inclusive levantando questões importantes sobre a utilização de paradigmas combinados de navegação e busca textual. Todo esse conhecimento foi aproveitado, como será visto mais à frente, na implementação de nosso estudo de caso.

Convém ainda destacar que o estudo da literatura, no que tange à questão da avaliação, tanto do algoritmo de busca, quanto dos modelos de visualização, revelou que não é tarefa fácil produzir uma avaliação consistente dos métodos propostos. Não existem coleções estruturadas de teste envolvendo normas jurídicas, e os modelos de visualização são por demasiado complexos e sutis para serem avaliados objetivamente. Portanto, concluímos – como outros também o fizeram (ZHANG, 2008, p. 253) – que esforços de desenvolvimento antecedem e abrem caminho para os de avaliação. Nessa dissertação, nos focaremos nos primeiros e realizaremos a avaliação através de pesquisa com os usuários.

Capítulo 3 – Buscas hierárquicas

3.1 – Introdução

Diversos experimentos foram realizados com o objetivo de avaliar quais as estratégias mais adequadas para a recuperação de informações contidas em normas jurídicas. Dentre esses experimentos, alguns trouxeram resultados expressivos o suficiente para que optássemos por incluí-los dentre as estratégias do Seeklex.

A seguir, descreveremos cada um desses experimentos, apresentando a motivação, o algoritmo desenvolvido e os resultados obtidos. Alguns algoritmos são voltados para a localização de tópicos relevantes e outros para a melhor visualização dos resultados obtidos.

Procuraremos, também, apresentar os algoritmos de acordo com a ordem cronológica em que foram concebidos, dessa forma, acreditamos que fique mais simples a compreensão do trabalho realizado e mais claras as motivações de cada uma das etapas.

Embora tenha sido expandido posteriormente para incluir outras normas jurídicas, inicialmente trabalhamos apenas com a Constituição Federal do Brasil de 1988, que nos foi gentilmente cedida por funcionários do Senado Federal envolvidos com o projeto LexML, portanto, todos os exemplos que daremos a seguir serão baseados em buscas realizadas neste documento.

3.2 – Conceitos gerais

Para este trabalho, usamos apenas documentos que contenham nós relacionados hierarquicamente. Também é levada em consideração a relação de linearidade presente na sequência natural do texto. Representaremos cada documento através de uma árvore que reflete a sua estrutura hierárquica (por exemplo, Inciso I do Artigo I é um nó filho do Artigo I). Assumimos, também, que a sequência linear do texto define a ordem entre os nós irmãos (por exemplo, Artigo 1 vem antes de Artigo 2).

Definição 3.1 – Documento hierárquico

É um documento representado por uma árvore na qual cada nó contém parte do conteúdo do documento e está relacionado com os demais nós por vínculos de hierarquia e linearidade. Denominamos de **corpus** o conjunto de documentos hierárquicos que compõem a base de dados de um sistema.

Definição 3.2 – Consulta

Um consulta representa uma necessidade de informação do usuário e é expressa por termos que devem estar presentes no resultado a ser apresentado. Por uma questão de simplificação, trabalharemos apenas com consultas que combinem termos de forma conjuntiva (E) ou disjuntiva (OU).

Definição 3.3 – Nó

Os nós que compõem os documentos estruturados dispõem dos seguintes atributos básicos:

1) **Rótulo**: é um nome único entre os irmãos desse nó, por exemplo, "Título I";
2) **Texto associado**: é o texto intrínseco de cada nó, por exemplo, o nó rotulado "Título I" da Constituição Federal Brasileira de 1988 possui o seguinte texto associado: "Dos princípios fundamentais".
3) **Identificador**: é a concatenação dos rótulos dos nós no caminho da raiz até o nó em questão;

Definição 3.4 – Texto completo de um nó

O texto completo de um nó é uma operação de agregação definida da seguinte forma:

- Se o nó for uma folha, seu texto completo é seu texto associado;
- Senão, seu texto completo é dado pela concatenação de seu texto associado com os textos completos de seus nós filhos, do primeiro para o último.

Definição 3.5 – Nó relevante

Um nó cujo texto completo possui algum dos termos utilizados na consulta é chamado de nó relevante.

Definição 3.6 – Nó candidato

Um nó cujo texto completo satisfaça todos os critérios da consulta é chamado de nó candidato. Como o texto completo de um nó inclui os textos completos de seus filhos, o pai de um nó candidato é também candidato.

Um nó candidato é dito **mínimo** se não possui nenhum filho que seja nó candidato.

Definição 3.7 – Nó dependente

Um nó é dependente se ele é um nó candidato e que deixa de ser candidato após a remoção de todos seus filhos que sejam nós candidatos. Ou seja, são nós dependentes todos os nós que não se sustentam sem seus filhos autônomos.

Definição 3.8 – Nó selecionado

Um nó que faz parte de uma das sub-árvores que compõem o resultado da consulta e que é suficientemente importante para que seu texto associado seja apresentado é chamado de um nó selecionado. Os demais nós que compõem o resultado, mas apresentam apenas seus rótulos e omitem o texto associado são denominados **nós desativados**. Um nó que pertença ao conjunto que compreende a totalidade dos nós presentes nas sub-árvores da resposta, composto pela união dos nós selecionados com os nós desativados, recebe o nome de **nó resultante**.

3.3 – Definição do problema

O objetivo perseguido consiste em aplicar uma **consulta** ao **corpus** e obter como resultado sub-árvores de todos os **documentos hierárquicos** que possuírem ao menos um **nó candidato**.

Estas sub-árvores deverão ser organizadas de modo a atender as necessidades de informação da forma mais completa e compacta possível, portanto os **nós resultantes** deverão ser cuidadosamente escolhidos, alguns serão **selecionados** para apresentar seus **textos associados** enquanto outros, **desativados**, mostrarão apenas seus **rótulos**. Também deverão ser aplicados critérios de ordenação para que as informações mais relevantes sejam vistas primeiro.

3.4 – Busca TF-IDF em subdocumentos

De modo a executar a tarefa descrita pela INEX (SHLOMO G. et al., 2009) como “Recuperação Completa”, inicialmente, optamos por particionar o documento hierárquico em cada um de seus elementos básicos, e realizar uma busca ordenada por TF-IDF. Isso nos serviu de ponto de partida, e, analisando os resultados obtidos, pudemos identificar possíveis deficiências e propor diversas estratégias que serão

discutidas nas próximas seções.

Para exemplificar, apresentamos abaixo, na Figura 3.1, os primeiros cinco resultados obtidos em uma busca pelos termos "`garantia propriedade herança`" na Constituição Federal Brasileira de 1988:

- TÍTULO II, CAPÍTULO I, Art. 5º, XXX –
  - é **garantido** o direito de **herança**;
- TÍTULO II, CAPÍTULO I, Art. 5º, XXII –
  - é **garantido** o direito de **propriedade**;
- TÍTULO VII, CAPÍTULO I, Art. 176.
  - As jazidas, em lavra ou não, e demais recursos minerais e os potenciais de energia hidráulica constituem **propriedade** distinta da do solo, para efeito de exploração ou aproveitamento, e pertencem à União, **garantida** ao concessionário a **propriedade** do produto da lavra.
- TÍTULO II, CAPÍTULO I, Art. 5º
  - Todos são iguais perante a lei, sem distinção de qualquer natureza, **garantindo**-se aos brasileiros e aos estrangeiros residentes no País a inviolabilidade do direito à vida, à liberdade, à igualdade, à segurança e à **propriedade**, nos termos seguintes:
- TÍTULO VII, CAPÍTULO I, Art. 170., II -
  - **propriedade** privada;

Figura 3.1 – Cinco primeiros resultados da busca TF-IDF.

3.5 – Formatação adequada

Seguindo a formatação proposta pela própria lei, determinados trechos devem ser apresentados diretamente depois de sua referência, sem que haja quebra de parágrafo. Esse é o caso dos artigos, incisos, parágrafos e alíneas. O resultado da pesquisa realizada na seção anterior, agora com a devida formatação, pode ser observado na Figura 3.2, abaixo:

- TÍTULO II, CAPÍTULO I, Art. 5º, XXX – é **garantido** o direito de **herança**;
- TÍTULO II, CAPÍTULO I, Art. 5º, XXII – é **garantido** o direito de **propriedade**;
- TÍTULO VII, CAPÍTULO I, Art. 176. As jazidas, em lavra ou não, e demais recursos minerais e os potenciais de energia hidráulica constituem **propriedade** distinta da do solo, para efeito de exploração ou aproveitamento, e pertencem à União, **garantida** ao concessionário a **propriedade** do produto da lavra.
- TÍTULO II, CAPÍTULO I, Art. 5º Todos são iguais perante a lei, sem distinção de qualquer natureza, **garantindo**-se aos brasileiros e aos estrangeiros residentes no País a inviolabilidade do direito à vida, à liberdade, à igualdade, à segurança e à **propriedade**, nos termos seguintes:
- TÍTULO VII, CAPÍTULO I, Art. 170., II – **propriedade** privada;

Figura 3.2 – Formatação semelhante à utilizada nas leis.

Não resta dúvida de que existe uma considerável redução do número de quebras de linha, e consequentemente um melhor aproveitamento do espaço vertical da página. Por acreditarmos que esse melhor aproveitamento é decorrente de uma organização mais clara, mais de acordo com a maneira como os textos legais são comumente visualizados, e que não traz nenhum comprometimento em relação à facilidade de compreender o resultado, optamos por incorporar essa estratégia.

3.6 – Reestruturação hierárquica

Com o intuito de facilitar a leitura dos resultados obtidos e reduzir a desorientação sofrida pelos usuários decorrente de nós com forte coesão estrutural serem apresentados em posições distantes (Chiaramela, Y. 1997), procedemos à reordenação hierárquica.

Operação 3.1 – Reestruturação Hierárquica

Para cada **nó candidato**, a partir de seu **identificador**, recompor os **identificadores** de todos os seus superiores hierárquicos. Criar **nós desativados** para cada **identificador** obtido. Em uma estrutura de árvores, inserir todos esses nós. Os **nós desativados** serão desprezados se já existirem nós com o mesmo **identificador**. Os **nós candidatos** substituirão **nós desativados** preexistentes nas árvores.

Algoritmo 3.1 – Reestruturação Hierárquica

Entrada: Uma lista A de **nós candidatos** de uma **consulta**

Saída: A', a árvore de nós reestruturados hierarquicamente.

1) Para cada nó a_i em A, faça:
   a) Seja P o conjunto de nós desativados superiores hierárquicos de a_i ordenados do maior nível hierárquico para o menor
   b) Para cada nó p_i em P
      i) Se não existir um nó n em A' tal que o identificador de n seja igual ao identificador de p_i, inserir p_i em A'
   c) Se existir um nó n em A' tal que o identificador de n seja igual ao identificador de a_i, substituir n por a_i em A'
   d) Senão, inserir a_i em A'
2) Pare e retorne A'

Antes de reorganizarmos hierarquicamente o resultado, precisamos recriar os componentes estruturais de cada um dos nós encontrados. Ou seja, temos que analisar a cadeia hierárquica presente em cada nó e reconstruir seus componentes separadamente.

O algoritmo de reordenação, portanto, consiste em identificar os componentes da cadeia hierárquica de cada nó encontrado e inseri-los em uma árvore, garantindo que um componente que já tenha sido previamente inserido será reaproveitado sempre que possível. Esse algoritmo é distribuído em três métodos que são explicados a seguir:

Um método é responsável por identificar todos os componentes da cadeia hierárquica de um nó, construir nós específicos para cada um desses componentes e inseri-los em uma lista de modo que o primeiro item desta lista seja o nó de mais alto nível hierárquico.

```
public List<Topic> getChain() {
     int i = indentLevel();
     List<Topic> l = new ArrayList<Topic>(i);
     for (Topic t = this; i > 0; i--) {
          l.add(0, t);
          t = t.getParent();
     }
     return l;
}
```

Figura 3.3 – Rotina para recuperar a cadeia hierárquica.

O método acima deve ser executado para cada um dos nós que foram retornados pela consulta TF-IDF, e as listas resultantes precisam ser inseridas em uma árvore.

```
for (Topic topic : setResult) {
     for (Topic t : topic.getChain())
          tree.append(t);
}
```

Figura 3.4 – Rotina para popular a árvore.

A rotina que insere os nós na árvore se vale de um mapa para detectar se um nó equivalente já foi previamente inserido, se for esse o caso, ela apenas agrega informações do nó atual em sua versão já cadastrada na árvore. Caso o contrário, o nó será inserido no mapa, e acrescentado no final da lista de filhos de seu superior hierárquico.

```
public T append(T data) {
     Node<T> n = map.get(data.getPath());
     if (n != null) {
          n.data.merge(data);
          return n.data;
     }

     Node<T> newNode = new Node<T>(data);
     map.put(data.getPath(), newNode);

     Node<T> nParent = map.get(data.getParentPath());
     if (nParent != null)
          nParent.addChild(newNode);
     else
          rootElements.add(newNode);
     return data;
}
```

Figura 3.5 – Rotina para inserir os tópicos na árvore.

No algoritmo utilizado, a ordem TF-IDF original é mantida, exceto quando um nó, por estar na cadeia hierárquica de um nó previamente inserido, é reposicionado acima de outro que possui uma melhor pontuação. No exemplo abaixo, pode-se observar que apenas o texto do quinto artigo foi reposicionado.

Um efeito benéfico da hierarquização do resultado é a supressão de repetições desnecessárias de um mesmo **rótulo**, como pode ser observado na Figura 3.6, abaixo.

- TÍTULO II
    - CAPÍTULO I
        - Art. 5º Todos são iguais perante a lei, sem distinção de qualquer natureza, **garantindo**-se aos brasileiros e aos estrangeiros residentes no País a inviolabilidade do direito à vida, à liberdade, à igualdade, à segurança e à **propriedade**, nos termos seguintes:
            - XXX - é **garantido** o direito de **herança**;
            - XXII - é **garantido** o direito de **propriedade**;
- TÍTULO VII
    - CAPÍTULO I
        - Art. 176. As jazidas, em lavra ou não, e demais recursos minerais e os potenciais de energia hidráulica constituem **propriedade** distinta da do solo, para efeito de exploração ou aproveitamento, e pertencem à União, **garantida** ao concessionário a **propriedade** do produto da lavra.
        - Art. 170.
            - II – **propriedade** privada;

Figura 3.6 – Reestruturação hierárquica dos tópicos encontrados.

3.7 – Compactação hierárquica

A existência de um número excessivo de recuos na exibição do texto nos levou a desenvolver um algoritmo de compactação hierárquica. Nesse algoritmo, um **nó desativado**, ou seja, que não deve apresentar seu **texto associado**, e que não possui nenhum "irmão", é agrupado com o "pai" de maneira a reduzir recuos desnecessários para a compreensão do resultado da busca.

Operação 3.2 – Compactação Hierárquica

Todo **nó** que não possuir "irmãos", e que for "filho" de um **nó desativado**, será agrupado com o "pai" para fins de apresentação.

Algoritmo 3.2 – Compactação Hierárquica

Entrada: Um conjunto R de **nós resultantes** de uma **consulta**

Saída: R', o conjunto de nós compactado hierarquicamente.

1) R' <- R

2) Para cada nó r_i em R', faça:

   a) Enquanto r_i for um nó desativado e possuir somente um filho f_i

      i) Remover f_i da lista de filhos de r_i

      ii) Inserir f_i na lista de irmãos de r_i para fins de apresentação

      iii) Se f_i for um nó selecionado, r_i passa a ser um nó selecionado

3) Pare e retorne R'

O método de compactação deve ser executado em cada nó raiz das sub-árvores que representam o resultado:

```
public void compact() {
     for (Node<T> n : getRootElements())
          n.compact();
}
```

Figura 3.7 – Rotina para iniciar a compactação.

Se um nó não deve apresentar seu **texto associado**, e se possui apenas um filho, esse filho é promovido para o mesmo nível hierárquico do pai, e seus filhos são transferidos para a lista de filhos de seu pai. Posteriormente, o procedimento é chamado recursivamente para cada um dos nós de nível inferior que não puderam ser assimilados.

```
public void compact() {
     if (data.getNumberOfCharacters() == 0)
          while (getChildren().size() == 1) {
               Node<T> n = getChildren().get(0);
               getSibling().add(n);
               setChildren(n.getChildren());
               n.children = null;
               if (n.data.getNumberOfCharacters() != 0)
                    break;
          }
     for (Node<T> n : getChildren())
          n.compact();
}
```

Figura 3.8 – Rotina de compactação hierárquica.

O resultado final são sub-árvores com menos níveis hierárquicos, que consequentemente ocupam menos espaço verticalmente. É importante ressaltar que não observamos nenhuma perda em relação à utilização desse tipo de compactação. Pelo contrário, a visualização se torna mais simples e clara. A compactação, se aplicada à

sub-árvore apresentada na seção anterior, produz o resultado que pode ser visto na Figura 3.9, abaixo:

- TÍTULO II, CAPÍTULO I, Art. 5º Todos são iguais perante a lei, sem distinção de qualquer natureza, **garantindo**-se aos brasileiros e aos estrangeiros residentes no País a inviolabilidade do direito à vida, à liberdade, à igualdade, à segurança e à **propriedade**, nos termos seguintes:
    - XXX - é **garantido** o direito de **herança**;
    - XXII - é **garantido** o direito de **propriedade**;
- TÍTULO VII, CAPÍTULO I
    - Art. 176. As jazidas, em lavra ou não, e demais recursos minerais e os potenciais de energia hidráulica constituem **propriedade** distinta da do solo, para efeito de exploração ou aproveitamento, e pertencem à União, **garantida** ao concessionário a **propriedade** do produto da lavra.
    - Art. 170., II - **propriedade** privada;

Figura 3.9 – Compactação hierárquica.

3.8 – Gatilho de ordenação sequencial

A falta de ordenação sequencial dos resultados pode confundir a leitura, principalmente quando se misturam tipos diferentes de nós "irmãos" em uma mesma lista.

Operação 3.3 – Gatilho de Ordenação Sequencial

Quando um **nó resultante** possuir um número pequeno de descendentes, seus "filhos" serão ordenados de acordo com a sequência natural do documento.

Algoritmo 3.3 – Gatilho de Ordenação Sequencial

Entrada: Um **nó resultante** N, o conjunto composto por seus "filhos" F, um documento D, o número máximo de nós que podem ser visualizados em uma tela M

Saída: F', o conjunto F reordenado na sequência natural de D.

1) F' <- F
2) Se N possuir M ou menos descendentes
    a) Para cada nó f_i em F', faça:
        i) Seja $índice_i$ a posição de f_i em D
    b) Ordene F' por $índice_i$
3) Pare e retorne F'.

O trecho abaixo, na Figura 3.10, representa o resultado completo da busca de "`garantia propriedade herança`" no artigo quinto e seus "filhos". Nele é possível observar como a falta de ordenação sequencial e a presença de parágrafos em meio a incisos dificulta a compreensão do resultado.

- TÍTULO II, CAPÍTULO I, Art. 5º Todos são iguais perante a lei, sem distinção de qualquer natureza, **garantindo**-se aos brasileiros e aos estrangeiros residentes no País a inviolabilidade do direito à vida, à liberdade, à igualdade, à segurança e à propriedade, nos termos seguintes:
    - XXX - é **garantido** o direito de **herança**;
    - XXII - é **garantido** o direito de **propriedade**;
    - XXIII - a **propriedade** atenderá a sua função social;
    - § 1º As normas definidoras dos direitos e **garantias** fundamentais têm aplicação imediata.
    - XXV - no caso de iminente perigo público, a autoridade competente poderá usar de **propriedade** particular, assegurada ao proprietário indenização ulterior, se houver dano;
    - XXVI - a pequena **propriedade** rural, assim definida em lei, desde que trabalhada pela família, não será objeto de penhora para pagamento de débitos decorrentes de sua atividade produtiva, dispondo a lei sobre os meios de financiar o seu desenvolvimento;
    - XXIX - a lei assegurará aos autores de inventos industriais privilégio temporário para sua utilização, bem como proteção às criações industriais, à **propriedade** das marcas, aos nomes de empresas e a outros signos distintivos, tendo em vista o interesse social e o desenvolvimento tecnológico e econômico do País;
    - VI - é inviolável a liberdade de consciência e de crença, sendo assegurado o livre exercício dos cultos religiosos e **garantida**, na forma da lei, a proteção aos locais de culto e a suas liturgias;
    - § 2º Os direitos e **garantias** expressos nesta Constituição não excluem outros decorrentes do regime e dos princípios por ela adotados, ou dos tratados internacionais em que a República Federativa do Brasil seja parte.

Figura 3.10 – Nós fora da sequencia natural.

Não é possível reordenar sequencialmente todo o resultado sem que seja totalmente destruída a informação de *ranking* produzida inicialmente pela aplicação do algoritmo TF-IDF. Portanto, se por um lado a ordem sequencial facilita a compreensão do resultado, por outro não podemos aplicá-la indiscriminadamente.

A solução encontrada para esse dilema consistiu na reordenação apenas de pequenos trechos do resultado. Por exemplo, no caso da Constituição Federal Brasileira, podemos reordenar os incisos de um artigo, mas não os artigos de um capítulo.

O critério de reordenação pode ser baseado no nível hierárquico dos nós, como exposto acima, ou pode ser ditado pela quantidade de tópicos que se pretende reordenar. Por exemplo, podemos definir que os "filhos" de um nó serão reordenados desde que ela possua dez ou menos descendentes. Na implementação do Seeklex, consideramos que o critério mais importante para definir a aplicação do gatilho de ordenação sequencial é o fato de que uma lista que seja suficientemente pequena para ser visualizada em uma página sem a necessidade de *scroll* pode ter seus elementos analisados por um simples passar de olhos, portanto, não precisa estar ordenada por relevância. Sendo assim, optamos por essa segunda alternativa, e o algoritmo que decide o método de ordenação foi definido assim:

```
public void sort(HireIndexReader reader, Query query) {
     if (getChildren().size() >= 1) {
          if (getTotalNumberOfChildren() < 10)
               sortSequentially();
          else
               sortByImportance(reader, query);
     }
     for (Node<T> n : getChildren())
          n.sort(reader, query);
}
```

Figura 3.11 – Critério para seleção do método de ordenação.

O resultado da consulta anterior, agora reordenado sequencialmente, pode ser observado na Figura 3.12, abaixo:

- TÍTULO II, CAPÍTULO I, Art. 5º Todos são iguais perante a lei, sem distinção de qualquer natureza, **garantindo**-se aos brasileiros e aos estrangeiros residentes no País a inviolabilidade do direito à vida, à liberdade, à igualdade, à segurança e à propriedade, nos termos seguintes:
    - VI - é inviolável a liberdade de consciência e de crença, sendo assegurado o livre exercício dos cultos religiosos e **garantida**, na forma da lei, a proteção aos locais de culto e a suas liturgias;
    - XXII - é **garantido** o direito de **propriedade**;
    - XXIII - a **propriedade** atenderá a sua função social;
    - XXV - no caso de iminente perigo público, a autoridade competente poderá usar de **propriedade** particular, assegurada ao proprietário indenização ulterior, se houver dano;
    - XXVI - a pequena **propriedade** rural, assim definida em lei, desde que trabalhada pela família, não será objeto de penhora para pagamento de débitos decorrentes de sua atividade produtiva, dispondo a lei sobre os meios de financiar o seu desenvolvimento;
    - XXIX - a lei assegurará aos autores de inventos industriais privilégio temporário para sua utilização, bem como proteção às criações industriais, à **propriedade** das marcas, aos nomes de empresas e a outros signos distintivos, tendo em vista o interesse social e o desenvolvimento tecnológico e econômico do País;
    - XXX - é **garantido** o direito de **herança**;
    - § 1º As normas definidoras dos direitos e **garantias** fundamentais têm aplicação imediata.
    - § 2º Os direitos e **garantias** expressos nesta Constituição não excluem outros decorrentes do regime e dos princípios por ela adotados, ou dos tratados internacionais em que a República Federativa do Brasil seja parte.

Figura 3.12 – Nós reordenados sequencialmente.

3.9 – Gatilho de contextualização

A falta de contextualização de determinados resultados da busca leva a pensar na possibilidade de incluir o **texto associado** a nós hierarquicamente superiores, apenas para contextualizar um resultado encontrado.

Operação 3.4 – Gatilho de Contextualização

Um **nó desativado** será promovido a **selecionado** sempre que for necessário para garantir a contextualização do **texto associado** de um **nó selecionado** "filho".

Algoritmo 3.4 – Gatilho de Contextualização

Entrada: Um conjunto S de **nós resultantes** de uma **consulta**, um limite de contextualização L

Saída: S', o conjunto de nós contextualizados.

1) S' <- S
2) Para cada nó s_i em S', faça:
   a) Se s_i for um nó desativado e estiver em L ou abaixo
      i) s_i passa a ser um nó selecionado
3) Pare e retorne S'.

O problema pode ser observado no fragmento do resultado de busca apresentado na Figura 3.13, abaixo, onde não fica claro o que se pretende em relação a "garantir o desenvolvimento nacional":

- CF1988, TÍTULO I, Art. 3º, II - **garantir** o desenvolvimento nacional;

Figura 3.13 – Resultado descontextualizado.

O resultado abaixo ficaria melhor explicado se o **texto associado** do nó "pai" – o artigo 3º – também fosse incluído na resposta. Produzindo, consequentemente, o seguinte resultado:

- CF1988, TÍTULO I, Art. 3º Constituem objetivos fundamentais da República Federativa do Brasil:
  - II - **garantir** o desenvolvimento nacional;

Figura 3.14 – Resultado contextualizado.

O algoritmo utilizado para a contextualização dos resultados consiste em varrer toda a árvore, localizar os nós que são inferiores a um nível mínimo de hierarquia que já garanta a contextualização, e recuperar o **texto associado** desses nós para que não apenas os **rótulos** sejam apresentados.

No Seeklex, optamos por considerar que o nível mínimo hierárquico necessário para a contextualização é o **texto associado** do "Artigo".

O trecho de código abaixo representa o método responsável pela contextualização:

```
public void contextualize(String limit, HireIndexReader reader)
          throws IOException {
     if (children == null || getChildren().size() == 0)
          return;
     for (Node<T> n : getChildren()) {
          n.contextualize(limit, reader);
     }
     if (data.isReferenceOnly()
               && (limit == null || data.getPath().contains(limit))) {
          int id = reader.getDocId(data.getPath());
          data.merge(reader.document(id), id);
     }
}
```

Figura 3.15 – Rotina para contextualizar os resultados.

3.10 – Ordenação de tópicos por acúmulo de pontuação

Quando os nós não são ordenados sequencialmente, devem ser ordenados em função da sua pontuação em relação à **consulta** vigente. Isso não representa nenhum desafio quando o nó encontrado não possui descendentes, mas pode ser mais complexo quando se tratar de um nó com "filhos".

Acreditamos que a ordenação deve privilegiar o nó que, somado a todos os seus "filhos", possua a maior pontuação. Nesse caso, consideramos importante levar em conta todos os "filhos", inclusive aqueles que não possuem nenhuma significância para a **consulta** em questão, pois, dependendo da função matemática escolhida para computar a pontuação, pode haver impacto. Portanto, para obtermos a pontuação acumulada, precisamos, para cada nó, possuir também um vetor de palavras referente à soma do texto específico do nó com o texto de todos os filhos, ou seja, o **texto completo do nó**.

No momento da reordenação, simplesmente calculamos a pontuação de cada nó com base na **consulta** em questão aplicada ao **texto completo**.

Essa abordagem, inicialmente pretendida apenas para a ordenação dos resultados, acabou nos conduzindo a realizar todas as operações de busca no conteúdo agregado dos nós, como será melhor explicado na seção seguinte.

3.11 – Busca no conteúdo agregado

Em determinados casos, podemos estar procurando termos que não existem no **texto associado** de um nó, mas que podem ser encontrados se consideramos seu **texto completo**. Esse ponto importante nos fez adotar uma nova estratégia que trouxe

diversos desdobramentos.

Passamos a considerar, para efeito de busca, não somente o **texto associado**, mas sim o **texto completo**. Dessa forma, o contexto é preservado. No entanto, isso causa uma distorção, pois, em uma **consulta** textual, o nó "pai" sempre terá uma classificação maior ou igual à dos **nós candidatos** filhos. Violando, portanto, o princípio da recuperação de documentos estruturados mencionado anteriormente. Uma alternativa viável para solucionar esse problema é fazer com que os pesos dos termos sejam atenuados quando eles são propagados para o nó "pai". (FUHR, N., GROßJOHANN, K., 2001)

Qualquer que seja a estratégia, em um modelo de recuperação de documentos estruturados baseado em agregação, a representação de um nó é obtida através da agregação de seu conteúdo particular aos conteúdos de seus nós filhos, ou de outra forma relacionados. Para tanto, é necessário que exista um método matemático capaz de combinar os atributos dos nós e produzir um resultado agregado único. (KAZAI, G. et al, 2001, p. 123)

A abordagem que adotamos é de simples implementação, podendo ser facilmente atingida através da manipulação dos termos que são relacionados com cada nó armazenado. Para contornar o problema levantado por Furh e Großjohann acima, desenvolvemos algumas estratégias que serão apresentadas mais à frente.

O **texto completo** foi obtido através da concatenação do **texto associado do nó** aos **textos completos** de cada um de seus "filhos", conforme a implementação recursiva abaixo:

```
public StringBuilder getCompleteContent(StringBuilder sb) {
     sb.append(content);
     sb.append(" ");
     for (Topic t : getChildren()) {
          t.getCompleteContent(sb);
     }
     return sb;
}
```

Figura 3.16 – Método utilizado para agregar o texto completo dos nós.

Posteriormente, observamos que a concatenação dos textos não deveria se propagar até os níveis mais altos da hierarquia, pois isso acarreta em resultados que – embora estejam corretos – são compostos por nós muito distantes e, consequentemente, sem muita correlação. Pensando assim, optamos por interromper a agregação em um

determinado nível. Escolhemos o "Artigo" para ser o mais alto nível da hierarquia a possuir o texto completo de seus "filhos".

3.12 – Desativação de superiores hierárquicos dependentes

Em decorrência da acumulação do **texto completo**, sempre que encontrarmos uma ou mais palavras em um nó, certamente as encontraremos também nos nós hierarquicamente superiores.

No entanto, se um nó superior não agrega nenhuma informação a mais, ou seja, se ele é um **nó dependente**, ele não é importante para o resultado da busca, consequentemente deve ser **desativado**. Isso atende ao "Princípio da Recuperação de Documentos Estruturados" (FUHR, N., GROßJOHANN, K., 2001), apresentado anteriormente.

Para tal, introduzimos um novo tipo de operação, a desativação de superiores hierárquicos dependentes, que verifica se um **nó candidato** continuaria sendo encontrado pela **consulta** mesmo que seus **nós candidatos** "filhos" fossem removidos.

Operação 3.5 – Desativação de Superiores Hierárquicos Dependentes

Todo **nó dependente** deve ser **desativado**.

Algoritmo 3.5 – Desativação de Superiores Hierárquicos Dependentes

Entrada: Um conjunto de **nós candidatos** A, uma **consulta** C e o documento D

Saída: A', o conjunto A com os **nós dependentes desativados**.

1) A' <- A
2) Para cada nó candidato a_i em A', faça:
   a) Seja F o conjunto dos nós filhos de a_i em D
   b) Para cada nó f_i em F, faça:
      i) Se f_i for candidato em uma busca com C, remova f_i de D
   c) Se a_i deixar de ser candidato em uma busca com C, desativar a_i
   d) Retornar para D os nós removidos
3) Pare e retorne A'.

Antes de explicar os detalhes do algoritmo de eliminação, é necessário esclarecer o método utilizado para implementá-lo. Não é fácil computar a relevância de um nó em relação a uma determinada **consulta**, descontando a influência causada por

alguns de seus descendentes, uma vez que as estruturas de dados normalmente utilizadas não foram especificamente projetadas para essa finalidade. Conseguimos obter os resultados desejados partindo do **texto completo** do nó e da subtração dos **texto completo** dos descendentes **candidatos**, mas, para explicar como isso foi realizado, precisaremos nos adiantar um pouco no assunto do capítulo seguinte, e apresentar alguns detalhes da implementação do Seeklex. Tivemos que alterar o funcionamento de determinados componentes básicos do Lucene – a ferramenta de busca textual adotada. Em especial, desenvolvemos uma subclasse de IndexReader, chamada HireIndexReader, que permite que seja indicada uma coleção de nós que podem ser retornados (desiredDocs) e uma coleção de nós cujo conteúdo deve ser descontado (removedDocs).

O algoritmo de eliminação de superiores hierárquicos dependentes foi, portanto, implementado conforme descrito a seguir. Para cada **nó candidato**, verificamos se ele possui descendentes **candidatos**. Se não possuir (**nó mínimo**), então será acrescentado no resultado (coleção lEnabled). Se possuir, só será acrescentado no resultado se continuar sendo um **nó candidato** para a **consulta** mesmo depois que seu **texto completo** seja subtraído dos termos constantes nos **textos completos** de seus descendentes **candidatos**.

```
private void selectIndependent(IndexSearcher searcher,
          HireIndexReader reader, Query query,
          List<Topic> lCandidates, List<Topic> lEnabled) {
     for (Topic t : lCandidates) {
          reader.removedDocs.clear();
          reader.desiredDocs.clear();
          reader.desiredDocs.add(t.getPath());
          for (Topic t2 : lCandidates)
               if (t2.childOf(t))
                    reader.removedDocs.add(t2.getPath());
          if (reader.removedDocs.size() == 0) {
               lEnabled.add(t);
               continue;
          }

          TopDocs rs = searcher.search(query, null, 1, Sort.INDEXORDER);
          if (rs.totalHits > 0) {
               lEnabled.add(t);
          }
     }
}
```

Figura 3.17 – Rotina para selecionar apenas os tópicos que independem de seus filhos.

3.13 – Inclusão de descendentes relevantes

Embora a busca seja realizada no **texto completo** dos nós, para simplificar a leitura, o resultado apresentado deve contemplar apenas o **texto associado** e os descendentes que contribuíram positivamente para os *scores*, ou seja, os descendentes **relevantes**.

Pensando dessa forma, optamos por somar à árvore produzida pela **consulta** inicial todos os nós, descendentes dos **nós selecionados**, que possuíssem pelo menos um dos termos pesquisados.

Operação 3.6 – Inclusão de Descendentes Relevantes

A cada **nó selecionado** serão acrescentados seus descendentes que sejam **candidatos** de uma consulta disjuntiva por todos os termos da **consulta** original.

Algoritmo 3.6 – Inclusão de Descendentes Relevantes

Entrada: Um conjunto de **nós selecionados** S, uma **consulta** C e o documento D

Saída: S', o conjunto S acrescido de seus descendentes **candidatos** da consulta disjuntiva formada por todos os termos da **consulta** original.

1) C' <- versão disjuntiva de C, S' <- S
2) Para cada nó selecionado s_i em S', faça:
    a) Seja d um descendente de s_i em D
        i) Se d for candidato em uma busca com C', incluir d em S'.
3) Pare e retorne S'.

Para realizar essa operação, impetramos uma busca booleana disjuntiva por todos os termos da **consulta** original, identificamos os nós resultantes descendentes de nós previamente selecionados, e os acrescentamos nas sub-árvores de resultados.

```
private void includeRelevantSubtopics(Query orQuery,
          IndexSearcher searcher, List<Topic> lEnabled,
          SortedSet<Topic> setResult) {
     TopDocs rs = searcher.search(orQuery, null, MAX_RESULTS_PER_PAGE,
               Sort.INDEXORDER);

     HashSet<String> hs = new HashSet<String>();
     for (Topic t : lEnabled)
          hs.add(t.getPath());

     for (int i = 0; i < Math.min(rs.totalHits, MAX_RESULTS_PER_PAGE); i++) {
          Document doc = searcher.doc(rs.scoreDocs[i].doc);
          Topic topic = new Topic(doc, rs.scoreDocs[i].doc);
          if (hs.contains(topic.getParentPath()))
               setResult.add(topic);
     }
}
```

Figura 3.18 – Rotina para incluir os descendentes relevantes.

Esse procedimento produz uma boa visualização dos resultados e garante que, conforme necessitamos no caso das buscas jurídicas, todas as referências encontradas sejam apresentadas. Pode ser visto a seguir, na Figura 3.19, um exemplo de inclusão de descendentes **relevantes**. Foi realizada uma **consulta** por "herança rural", e apenas o Art. 5º da Constituição Federal foi identificado como um **nó candidato**. No entanto, em seu **texto associado**, ele não traz nem o termo "herança", nem o termo "rural", sendo portanto necessário identificar os descendentes que possuem esses termos e incluí-los na visualização do resultado.

- CF1988, TÍTULO II, CAPÍTULO I, Art. 5º Todos são iguais perante a lei, sem distinção de qualquer natureza, garantindo-se aos brasileiros e aos estrangeiros residentes no País a inviolabilidade do direito à vida, à liberdade, à igualdade, à segurança e à propriedade, nos termos seguintes:
    - XXVI - a pequena propriedade **rural**, assim definida em lei, desde que trabalhada pela família, não será objeto de penhora para pagamento de débitos decorrentes de sua atividade produtiva, dispondo a lei sobre os meios de financiar o seu desenvolvimento;
    - XXX - é garantido o direito de **herança**;

Figura 3.19 – Visualização dos descendentes relevantes.

3.14 – Ordenação de nós em função do impacto no score do superior hierárquico

Em alguns casos, a quantidade de nós "filhos" apresentados na visualização de um nó encontrado é muito grande. Quando isso ocorre, é importante que exista uma estratégia para ordenar por relevância esses descendentes. No entanto, não é possível utilizar o critério de TF-IDF para obter essa ordenação, uma vez que os descendentes não são necessariamente **nós candidatos**, pois eles também podem ter sido incluídos apenas por serem **nós relevantes**. Sendo assim, não será possível calcular o *score* pela **consulta** original, já que eles só possuirão alguns dos termos pesquisados.

Para solucionar essa questão, optamos por produzir um algoritmo de ordenação de descendentes em função do impacto que a remoção de cada um deles acarreta na pontuação do nó "pai".

Operação 3.7 – Ordenação de Nós em Função do Impacto no *Score* do Superior Hierárquico

Cada "filho" de um **nó resultante** será priorizado na ordenação em função do impacto sofrido no *score* do "pai" quando a **consulta** é realizada após a remoção deste "filho".

Algoritmo 3.7 – Ordenação de Nós em Função do Impacto no *Score* do Superior Hierárquico

Entrada: Um **nó selecionado** N, o conjunto composto por seus "filhos" F, uma **consulta** C, um documento D

Saída: F', o conjunto F reordenado em função do impacto no score de N.

1) C' <- versão disjuntiva de C, F' <- F
2) Para cada nó f_i em F', faça:
    a) Seja $índice_i$ a posição f_i em D
    b) Remova f_i de D
    c) Seja $impacto_i$ o score de N para a busca C
    d) Seja $impactoDisjuntivo_i$ o score de N para a busca C'
    e) Retorne f_i para D
3) Ordene F' por $impacto_i$, $impactoDisjuntivo_i$, $índice_i$
4) Pare e retorne F'

A pontuação do nó "pai" é recalculada subtraindo-se de seu **texto completo** o **texto completo** de cada um dos seus "filhos". Posteriormente, essa pontuação é utilizada para reordenar a lista dos "filhos", sendo certo que os que – quando removidos – causarem maior impacto no *scores* do "pai", serão os primeiros da lista.

Quando existe empate em relação ao impacto no superior hierárquico, o critério de ordenação passa a ser a relevância dos termos para uma consulta booleana do tipo "ou". Se, ainda assim, o empate persistir, a ordem sequencial natural do documento será mantida.

```
private void sortByImportanceToParentScore(HireIndexReader reader,
          Query query, Query queryOr) {
     IndexSearcher searcher = new IndexSearcher(reader);
     reader.removedDocs.clear();
     reader.desiredDocs.clear();
     reader.desiredDocs.add(data.getPath());
     for (int i = 0; i < getChildren().size(); i++) {
          Node<T> node = getChildren().get(i);
          String path = node.data.getPath();
          reader.removedDocs.add(path);
          TopDocs rs = searcher.search(query, null, 1);
          node.importance = rs.totalHits == 1 ? rs.scoreDocs[0].score
                    : 0f;
          TopDocs rsOr = searcher.search(queryOr, null, 1);
          node.scoreOr = rs.totalHits == 1 ? rsOr.scoreDocs[0].score : 0f;
          reader.removedDocs.remove(path);
     }
     Collections.sort(getChildren(), comparator);
}
```

Figura 3.20 – Rotina para ordenar em função do impacto no *score* do superior hierárquico.

O método de comparação, em si, pode ser observado no trecho de código abaixo:

```
private Comparator<Node<T>> comparator = new Comparator<Node<T>>() {
     public int compare(Node<T> node1, Node<T> node2) {
          int i = Float.compare(node1.importance, node2.importance);
          if (i != 0)
               return i;
          i = Float.compare(node1.scoreOr, node2.scoreOr);
          if (i != 0)
               return i;
          return node1.data.getId() - node2.data.getId();
     }
};
```

Figura 3.21 – Critérios de desempate na ordenação.

3.15 – Considerações finais

Neste capítulo, apresentamos os diversos algoritmos que foram especialmente desenvolvidos para lidar com o processo de recuperar informações provenientes de normas jurídicas.

Por não estarmos buscando, em um primeiro momento, extrema eficiência, mas sim encontrar uma boa solução para a priorização e a estruturação do resultado da busca, durante essa etapa do estudo, optamos por processar todos os nós encontrados e armazená-los em uma estrutura de árvore na memória, a partir da qual diversas operações puderam ser realizadas antes da apresentação. Essa abordagem mostrou-se bastante flexível e nos possibilitou testar uma grande quantidade de estratégias, como por exemplo: hierarquizar, filtrar, complementar, reordenar e compactar.

Uma análise preliminar parece indicar, para o caso específico das normas jurídicas, que bons resultados são obtidos através da busca no texto completo, com desativação de superiores hierárquicos dependentes, inclusão de descendentes relevantes, ordenação em função do impacto na pontuação do superior hierárquico, contextualização e compactação. No entanto, é certo que ainda existe um grande espaço a ser explorado em relação à busca e recuperação de informação em documentos hierarquizados.

Todos os algoritmos referenciados acima foram utilizados na implementação de nosso estudo de caso – o Seeklex –, conforme veremos no capítulo seguinte.

Capítulo 4 – Seeklex: implementação

4.1 – Introdução

Neste capítulo apresentaremos as características técnicas da implementação de nosso estudo de caso – o Seeklex –, um mecanismo de busca em normas jurídicas disponível para acesso gratuito no site www.seeklex.com.br.

Também detalharemos o processo de pré-processamento e importação das normas legais disponibilizadas para consulta.

Por fim, apresentaremos os resultados da avaliação do sistema, obtida através de pesquisa que realizamos junto a estudantes de direito.

4.2 – Ambiente de experimentação

A implementação do Seeklex foi realizada em um ambiente de experimentação. Esse ambiente consiste em um sistema desenvolvido em linguagem Java que se vale do componente de código aberto Lucene (2010) para armazenar os índices invertidos de termos e para realizar as consultas. Além do sistema e da ferramenta de busca textual, diversos outros componentes foram utilizados, conforme veremos a seguir:

4.2.1 – Lucene

De acordo com o site The Open Source Census (2010), a biblioteca de busca Lucene é um dos 15 principais projetos de código aberto, sendo, em especial, um dos 5 projetos mais importantes da The Apache Foundation (2010). O Lucene está instalado em mais de 4.000 empresas e é baixado mais de 6.000 vezes por dia.

Algumas das razões para a popularidade e a adoção do Lucene são (Lucid Imagination, 2010):

- Melhores práticas de cálculo de relevância;
- Excelente desempenho para as consultas;
- Escalabilidade comprovada (responsável pelas buscas em sites como o LinkedIn, Wikipédia, Apple, Technorati e Netflix);
- Ótima flexibilidade e capacidade de personalização (bem escrito e facilmente modificável);
- Baixo custo e independência de fornecedor (código aberto).

4.2.2 – Brazilian Stemmer

Para garantir a redução das palavras aos seus radicais, utilizamos no Seeklex a implementação padrão de *stem* do Lucene para o português brasileiro. Esta implementação consistem em um analisador (BrazilianAnalyzer), um filtro de palavras (BrazilianStemFilter), e, finalmente, da classe que implementa o algoritmo de stem propriamente dito (BrazilianStemmer).

Este componente foi desenvolvido por João Kramer, um funcionário do Tribunal de Contas da União e está disponível no pacote de analisadores que faz parte da distribuição padrão do Lucene (lucene-analyzers.jar).

4.2.3 – JTidy

JTidy (2010) é um porte para Java da ferramenta HTML Tidy (2010), originalmente implementada em ANSI C. Como parte do processo de preparação de dados, os arquivos HTML contendo as normas jurídicas foram preprocessados, para fins de padronização, como o uso do componente JTidy (2010). O JTidy, além de corrigir eventuais problemas no HTML, é capaz de converter o documento para o formato XHTML, o que facilita a etapa posterior de leitura e interpretação.

4.2.4 – Streaming API for XML (StAX)

StAX (2010) é a implementação de referência de uma API que permite a leitura e a escrita de documentos XML. Algumas das principais características desse componente são:

- A aplicação controla o processo de leitura (pull-parser);
- Não realiza a validação do documento em relação a DTD (non-validating parser);
- Possui um bom desempenho;
- Não necessita de muita memória de trabalho;
- É uma biblioteca de código aberto.

4.2.5 – Google App Engine

A hospedagem do site www.seeklex.com.br foi deixada a cargo do Google App Engine. Um serviço lançado recentemente pela Google que permite que aplicações web sejam disponibilizadas através da própria infra-estrutura desta empresa. A proposta do App Engine consiste em facilitar o desenvolvimento e a manutenção das aplicações, ao

mesmo tempo em que viabiliza níveis excelentes de escalabilidade conforme as demandas de tráfego e armazenamento crescem.

O Google App Engine suporta aplicações escritas em diferentes linguagens de programação, incluindo tecnologias padrão do Java, como a JVM, Java Servlets e a linguagem Java propriamente dita.

Para volumes baixos de consumo de CPU, tráfego e armazenagem, o serviço é gratuito.

Os principais recursos e características técnicas oferecidos pelo App Engine são:

- Serviço web dinâmico, com total suporte às tecnologias web mais comuns;
- Persistência de dados com a possibilidade de realizar consultas, ordenações e transações;
- Escalabilidade e distribuição de carga automáticas;
- APIs para autenticar usuários e enviar emails utilizando credenciais do Google;
- Um completo ambiente de desenvolvimento local que simula o Google App Engine em um computador pessoal;
- Filas de tarefas para executar trabalhos fora do escopo de uma requisição web;
- Tarefas agendadas para disparar eventos em horários específicos ou intervalos regulares.

4.3 – Carga de dados

4.3.1 – CF1988

João Alberto de Oliveira Lima, funcionário do Senado Federal e um dos responsáveis pelo projeto LexML, nos forneceu uma versão da Constituição Federal de 1988 em formato XML e de acordo com o esquema do LexML, para que iniciássemos os trabalhos que culminaram nessa dissertação. Com base neste arquivo XML, todos os algoritmos apresentados anteriormente foram desenvolvidos.

No entanto, para tornar o Seeklex mais atrativo e consequentemente poder melhor avaliá-lo, tivemos que buscar maneiras de aumentar a oferta de conteúdo, incluindo minimamente os demais Códigos Federais.

Para tanto, o próprio João Lima nos ofereceu um parser, desenvolvido como parte do projeto LexML.

4.3.2 – Parser do LexML

O projeto LexML produziu um parser para transformar o texto integral de leis em documentos XML bem formatados e compatíveis com o esquema do LexML.

Esse parser, que pode ser executado a partir de arquivos RTF, HTML ou TXT, foi desenvolvido em Java com auxílio do componente ANTLR 1.1.7.

Embora tenhamos empregado muito esforço em utilizar o parser do LexML para interpretar os demais Códigos Federais e incorporá-los ao Seeklex, não obtivemos resultados satisfatórios. Isso se deve basicamente ao fato de o parser do LexML esperar que as leis sejam corretamente formatadas, o que raramente é o caso.

Por mais difícil que seja acreditar, o fato é que as leis que são disponibilizadas pelos canais oficiais, como por exemplo o site do Planalto, estão repletas de erros de formatação, datilografia, etc. Além disso, em diversas ocasiões, existem discrepâncias entre a estrutura do documento legal e as normas de redação ditadas pela Lei Complementar 95 de 26 de fevereiro de 1998, o que confunde o parser do LexML e fez com que fosse impossível utilizá-lo para converter todos os Códigos Federais que pretendíamos.

4.3.3 – Parser do Seeklex

A solução encontrada para contornar esse problema consistiu em desenvolver um novo parser, mais flexível e adaptável para que pudéssemos incluir vários os Códigos Federais em nosso estudo de caso.

Nosso parser foi desenvolvido em Java, e utilizou expressões regulares para localizar as marcações nas normas legais e produzir os documentos estruturados necessários ao Seeklex.

Por fim, optamos por não gerar arquivos XML intermediários, mas sim utilizar a saída do parser diretamente para alimentar a base de dados do Lucene. Essa base de dados fornece todas as informações necessárias para realização das consultas no Seeklex, além de ser capaz de recompor o texto completo das normas indexadas para apresentação de resultados ou para a navegação.

4.3.4 – Alterações necessárias nos arquivos de origem

Mesmo com um parser extremamente flexível, todos os arquivos precisaram sofrer correções manuais antes que fosse possível convertê-los.

Essas alterações, de modo geral, foram realizadas em erros de digitação,

pequenos detalhes como algarismos romanos que utilizavam o “L” minúsculo no lugar de um “I” maiúsculo. Ou em casos em que a pontuação, como por exemplo um travessão, vinha colado ao número de um inciso, em vez de haver um espaço entre eles. Em outros casos, foi necessário marcar manualmente a divisão da lei em suas partes ou livros.

Determinados trechos que fugiam completamente à estrutura proposta pela LC 95, citada anteriormente, receberam uma marcação especial para que fossem considerados simplesmente como texto pelo parser.

Depois de todas essas correções, conseguimos realizar as importações e as normas legais foram finalmente disponibilizadas no Seeklex.

4.3.5 – Sistema de verificação diária de atualizações

As leis não são estáticas, e os Códigos Federais estão constantemente sendo atualizados em função de emendas, no caso da Constituição Federal, ou de leis que são editadas e alteram porções significativas dos Códigos vigentes.

Para garantir que o Seeklex sempre possuirá a versão mais atual das normas jurídicas que indexa, desenvolvemos um mecanismo de monitoramento das fontes e notificação de alterações.

Diariamente, o site busca as normas indexadas na fonte e calcula o resumo (MD5) destes arquivos. Esses resumos são comparados com os resumos dos arquivos que utilizamos como base para a importação, e sempre que existe uma discrepância são enviados emails de notificação, além de ficar registrado na página inicial do Seeklex desde quando a norma está desatualizada.

Consideramos esse mecanismo fundamental, pois ele garante que as normas estão sempre atualizadas, tranquilizando os usuários e aumentando, consequentemente, o grau de confiança no Seeklex.

4.4 – Algoritmos de busca e organização

Conforme vimos no capítulo anterior, nosso estudo levou a diversos algoritmos, tanto de busca, como de organização dos resultados para posterior apresentação.

Esses algoritmos foram implementados em Java, em um Servlet, para que as consultas possam ser realizadas via web. Os resultados sempre são formatados em padrão HTML com links que propiciam a navegação ou a reestruturação do que está sendo apresentado, como é o caso da visualização de “índice”, ou de “texto completo”.

Este Servlet, bem como todo o resto da aplicação e sua base de dados foram instalados no Google App Engine para que o Seeklex pudesse ser disponibilizado para o público.

4.5 – Algoritmos de navegação

4.5.1 – Geração de HTML

De modo a produzir um link único, imutável e auto-explicativo, desenvolvemos um Servlet que transforma uma requisição do tipo HTML em uma consulta à base de dados. Assim sendo, uma solicitação do tipo: http://www.seeklex.com.br/cf1988-titulo-ii-capitulo-i-art-5.html será convertida em uma consulta cujo parâmetro de busca é: nav:cf1988-titulo-ii-capitulo-i-art-5.

Existe, ainda, uma segunda etapa, na qual o termo de consulta nav:cf1988-titulo-ii-capitulo-i-art-5 é convertido na sintaxe que o processador de consultas do Lucene é capaz de compreender, ficando consequentemente da seguinte forma: nav:[nav:cf1988-titulo-ii-capitulo-i-art-5 TO nav:cf1988-titulo-ii-capitulo-i-art-5/\\]. Isso garante a criação de um *ConstantScoreRangeQuery*, que é capaz de processar a consulta com excelente desempenho.

4.5.2 – Otimização para mecanismos de busca

Uma das formas mais comuns de divulgação de um site na Internet consiste em adequá-lo da melhor maneira possível aos mecanismos de busca, como o Google e outros. Para tanto, algumas otimizações precisam ser realizadas e alguns padrões devem ser seguidos. Estas otimizações, conhecidas pelo acrônimo SEO – Search Engine Optimization, foram cuidadosamente introduzidas no Seeklex.

As solicitações realizadas através do servlet que intercepta arquivos HTML também têm uma característica importante em relação à otimização para melhor lidar com os mecanismos de busca, como o Google e outros. Elas informam no cabeçalho do HTTP o parâmetro *Last-Modified*, contendo a data de última alteração do arquivo que contém a norma legal em questão. E, quando a solicitação é realizada com o parâmetro *IF-Modified-Since*, o erro 304 é retornado se não houver alteração na norma legal. Dessa forma, os robôs são capazes de manter os dados dos mecanismos de busca atualizados com muito mais facilidade.

4.5.3 – Paradigmas de interação: BQ, QB e BO

Os três paradigmas para a visualização de RI foram contemplados no projeto do Seeklex. Exemplos práticos de cenários onde esses paradigmas são aplicados podem ser vistos a seguir:

(i) QB: busca textual seguida de navegação. Logo após realizar a busca textual, o usuário recebe os resultados na forma de uma árvore com links para a visualização de cada um dos tópicos apresentados. Ao clicar nestes links, o usuário estará realizando uma navegação após a busca textual.

(ii) BQ: navegação seguida de busca textual. Nessa modalidade o sistema realça as palavras buscadas ou apresenta resultados restritos ao contexto da navegação. Ao realizar uma navegação, a caixa de pesquisa conterá um termo especial que representa o elemento que está sendo atualmente exibido. O usuário, poderá, então, acrescentar nessa caixa de pesquisa outros termos para realizar um busca restrita ao elemento selecionado.

(iii) BO: somente navegação. A navegação pode ser realizada da página inicial ou de qualquer página de resultados e permite, ainda, a visualização apenas do índice ou a expansão do texto completo.

4.6 – Conformidade com critérios de visualização

Descreveremos a seguir como foram implementadas, no Seeklex, as recomendações a respeito do modelo de visualização e de interação com os usuários propostas por Zhang (2008, p. 246)

1. Navegação:
    a. Orientação: Durante a navegação, sempre são apresentados todos os links para os níveis hierárquicos superiores, de forma que o usuário pode facilmente obter informações sobre o contexto e navegar tanto para os tópicos inferiores, quanto para os superiores.
    b. Exploração: Na página inicial do Seeklex, existe uma listagem de todas as normas legais disponíveis. Clicando sobre o número da norma, o usuário será levado para um índice dos capítulos e seções desta norma. De lá, ele pode seguir navegando para os artigos e o texto completo. Dessa forma, garantimos que é possível, de um ponto único de partida, navegar para qualquer tópico de informação.

c. Conexões: Como já explicitado anteriormente, ao apresentarmos um tópico de informação, também disponibilizamos links para seus superiores hierárquicos e para a visualização de seus filhos. Portanto, todas as conexões presentes na estrutura da informação são visualizadas.

2. Busca textual:
    a. Simplicidade: A busca textual simples é a forma mais comum de os usuários interagirem como Seeklex. Tal como no Google, ou em outro mecanismo de busca qualquer, basta ao usuário escrever as palavras que deseja consultar e clicar em "Pesquisar" para obter os resultados da busca textual. A única diferença é que esses resultados serão apresentados de maneira estruturada e que haverá a possibilidade de iniciar uma navegação a partir de links presentes nos tópicos retornados.
    b. Reformulação: A reformulação também é disponibilizada da mesma forma que nos principais mecanismos de busca. No resultado, é apresentada uma caixa de pesquisa semelhante à presente na página inicial, e já preenchida com os termos da consulta original. Dessa forma, para reformular a consulta, basta que o usuário substitua os termos dessa caixa de textos e clique novamente em "Pesquisar".
3. Representação visual da informação
    a. Intuitividade: Para facilitar a compreensão dos mecanismos de interação com o sistema, optamos por utilizar uma interface semelhante à do Google e dos principais sites de busca. A página inicial é extremamente simples, contendo apenas uma caixa de texto e um botão para pesquisa, além dos links citados anteriormente para navegação no conteúdo completo. Existe uma página de "Pesquisa avançada" onde podem ser fornecidas informações mais detalhadas sobre os termos pesquisados sem que haja necessidade, por parte do usuário, de conhecer a sintaxe exata das consultas no Lucene. Os resultados da busca são apresentados na forma de uma árvore de links e textos.
4. Controle de interação
    a. Zoom: Além da navegação para os tópicos superiores ou inferiores hierarquicamente, o zoom também é garantido através da opção que o sistema disponibiliza entre apresentar o índice ou o texto completo da norma jurídica selecionada.

b. Histórico: Por se tratar de uma aplicação web, o histórico é facilmente obtido através dos recursos próprios do navegador. O único cuidado especial que tomamos foi de não realizar operações de POST entre as páginas, somente GET, de modo que não há confirmação de resubmissão de páginas nem confusão em relação a quais campos precisam ser alterados para justificar uma nova entrada no histórico, ou no *cache*.

4.7 – Medidas de comparação

4.7.1 – O estudo de caso e a metodologia

O estudo presente nessa dissertação tem por objetivo investigar o grau de facilidade e utilidade do Seeklex. Discutiremos as percepções de usuários do sistema que responderam um questionário sobre diversos aspectos como, por exemplo, a cobertura, a precisão, a navegação e a organização dos resultados. No entanto, convém lembrar que estes usuários não tiveram acesso a detalhes técnicos da implementação, como os algoritmos ou as estruturas de dados. Suas opiniões são baseadas apenas na experiência de uso do Seeklex e na comparação deste com o Google.

4.7.2 – Fonte de dados: o questionário

A única fonte de informações utilizada para o estudo de caso foi um questionário disponibilizado via web para estudantes de direito interessados em avaliar o Seeklex (Anexo 1). A divulgação do questionário foi feita através da lista de discussão GEDEL que reúne profissionais e estudantes do direito e de TI.

De 43 questionários recebidos, 2 foram descartados por possuírem informações incompletas, resultando em um total de 41 questionários aproveitados para as análises (Anexo II) que serão apresentadas a seguir.

4.7.3 – Identificação dos usuários

As três primeiras perguntas do questionário têm por objetivo caracterizar o grupo de respondedores, para tanto, solicitamos que fosse informada a Faculdade/Universidade em que estudam, o período que cursam e a idade. O Quadro 4.1, abaixo, ilustra a diversidade de instituições de ensino envolvidas na pesquisa. Alguns professores se interessaram pelo assunto e solicitaram que seus alunos participassem da pesquisa, isto explica a concentração de questionários provenientes,

por exemplo, da Faculdade de Direito do Sul de Minas e da Universidade Católica do Salvador.

Faculdade/Universidade	Total
Cândido Mendes	1
Centro Universitário de Sete Lagoas - UNIFEMM	4
Centro Universitário Newton Paiva	1
Faculdade de Ciências Jurídicas de Diamantina	1
Faculdade de Direito do Sul de Minas	7
FUMEC	1
FUNCEC	1
UNIJORGE	1
PUCMINAS	1
UERJ	1
UFF	1
UFMG	3
Univercidade	1
Universidade Católica do Salvador	9
Universidade Estácio de Sá	3
Universidade Federal de Minas Gerais	1
Não informado	4
TOTAL	41

Quadro 4.1 – Faculdades/Universidades dos respondedores

Percebemos, na Figura 4.1 abaixo, que o período cursado foi bem distribuído entre os respondedores, sendo observada apenas uma menor quantidade de questionários provenientes dos primeiros três períodos. Este fato provavelmente decorre de o processo de divulgação da pesquisa ter atingido majoritariamente alunos com mais anos de curso.

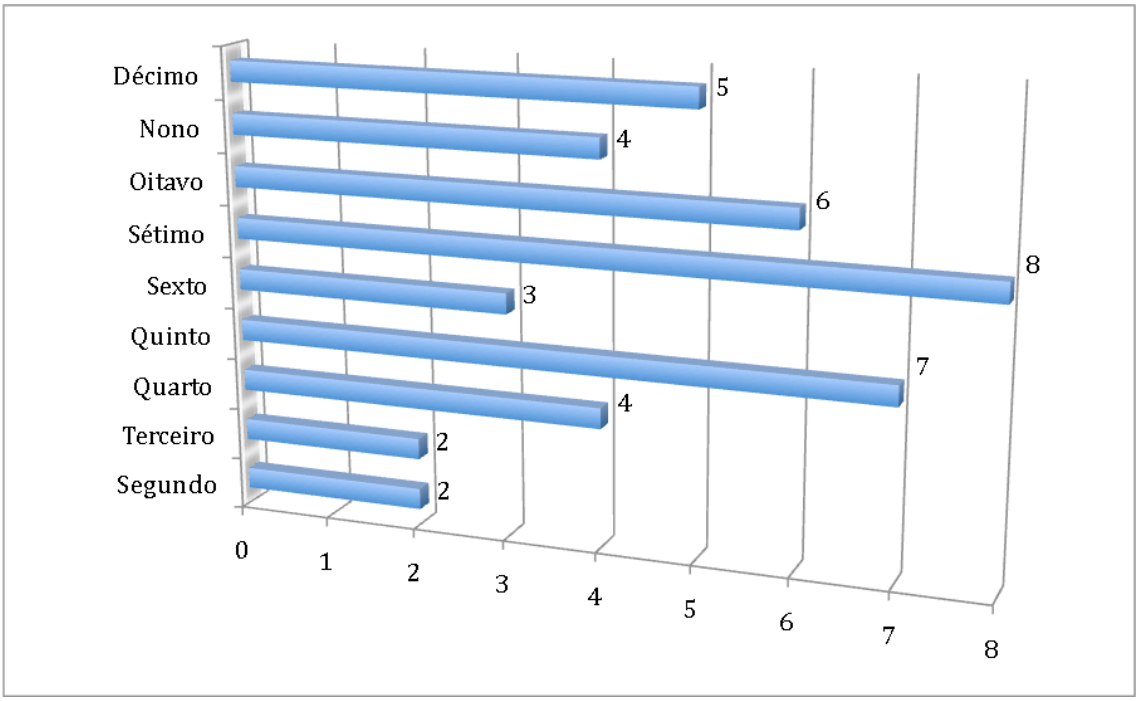

Figura 4.1 – Período do curso de direito.

A idade dos respondedores, como era de se esperar, é compatível com a idade média de um estudante universitário. Sendo observado, na Figura 4.2 abaixo, que 63% dos questionários foram respondidos por alunos de 25 anos ou menos.

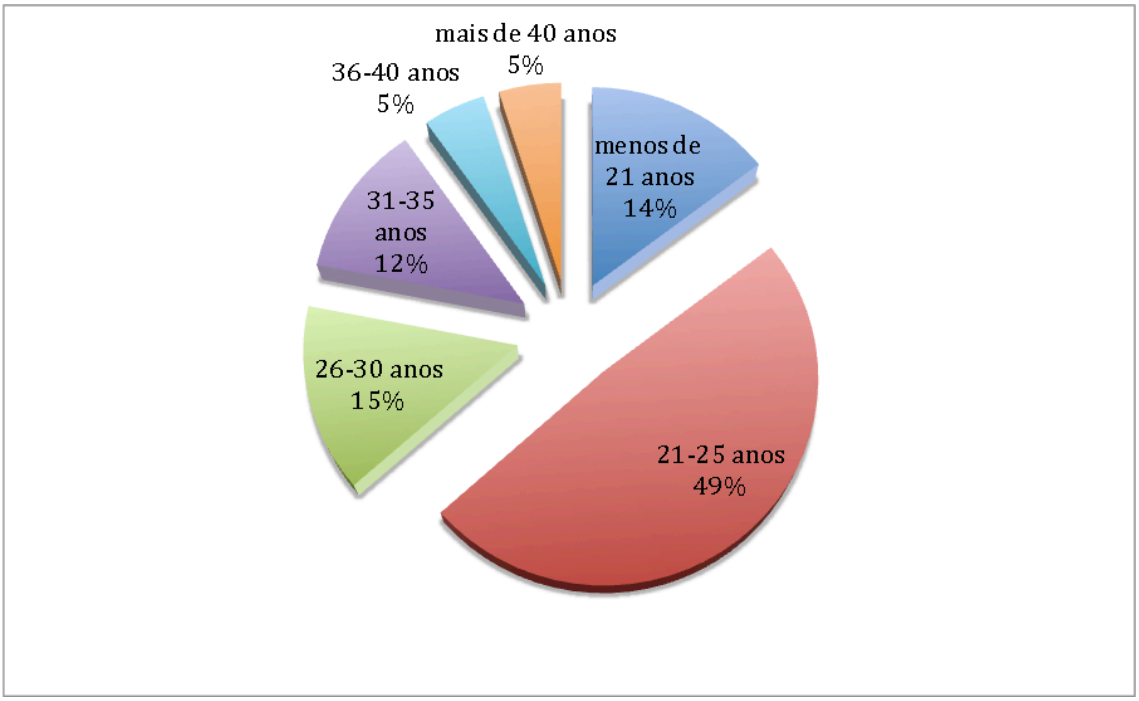

Figura 4.2 – Idade.

4.7.4 – Cobertura e precisão

Antes de apresentar os números referentes à cobertura e precisão, é importante esclarecer que o questionário solicitava que os alunos comparassem o Seeklex com o Google, sendo que a base de dados do Seeklex é composta apenas de 11 Códigos Federais, enquanto o Google indexa uma grande parte de todos os documentos disponíveis na Internet. Sendo assim, é de se esperar que a cobertura do Google seja bem melhor do que a do Seeklex.

Por não termos disponível uma coleção de testes de documentos estruturados compatível com a proposta do Seeklex, não pudemos fazer uma avaliação quantitativa padrão de cobertura x precisão, no entanto, procuramos obter informações qualitativas suficientes para sustentar estatisticamente uma comparação entre os sistemas.

Para avaliar a cobertura, solicitamos que o usuário respondesse se conseguiu encontrar no Seeklex, e também no Google, o que estava procurando. Para caracterizar o nível de precisão, perguntamos se os sistemas apresentavam mais resultados relevantes do que irrelevantes.

Por fim, para explicitar ainda mais as diferenças, solicitamos que os respondedores comparassem a relevância dos resultados dos dois sistemas e respondessem se o Seeklex apresentava resultados mais relevantes do que o Google. Os resultados podem ser observados no gráfico da Figura 4.3, abaixo:

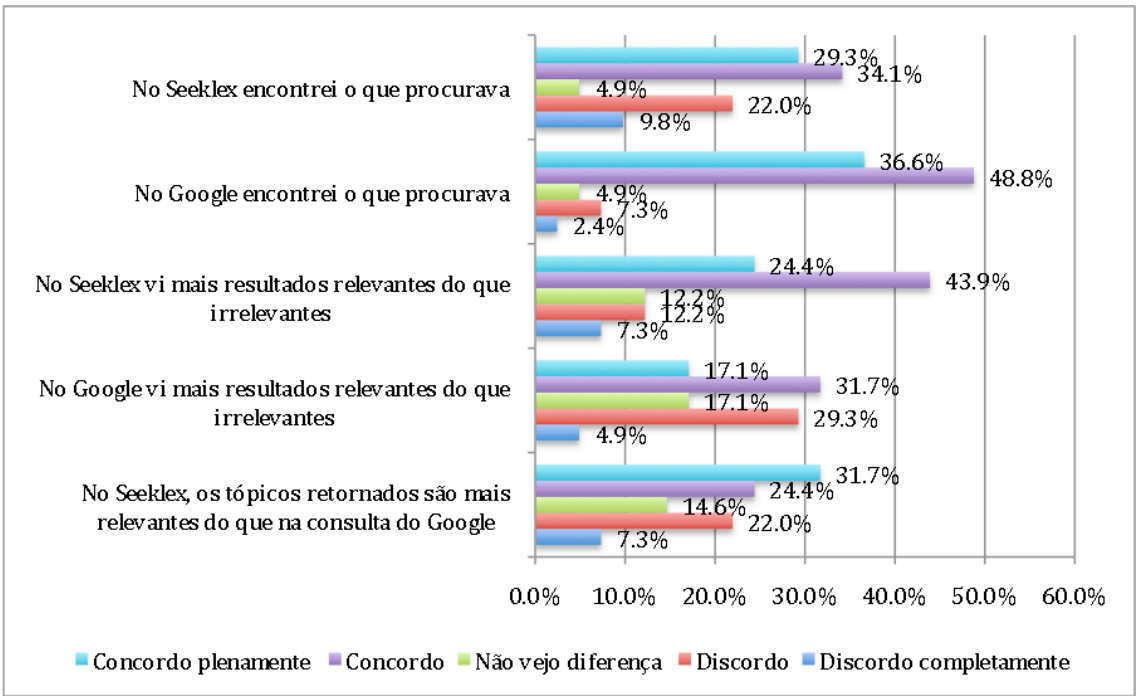

Figura 4.3 – Cobertura e Relevância.

Como era de se esperar, embora 63,4% dos respondedores tenham concordado ou concordado completamente que conseguiram encontrar no Seeklex respostas para suas necessidades de informação, a cobertura do Google se mostrou ainda superior, obtendo para a mesma caracterização uma pontuação de 85,4%. Essa diferença, embora significativa, consideramos pequena se comparada à discrepância existente entre a base de dados dos dois sistemas.

Já no quesito relevância, o Seeklex obteve melhor pontuação, pois os respondedores que concordaram ou concordaram plenamente que os resultados do Seeklex são mais relevantes do que irrelevantes somou 68,3%, contra 48,8% que avaliaram da mesma forma o Google.

Quando solicitamos que a relevância dos resultados do Seeklex fosse diretamente comparada com a relevância dos resultados do Google, o Seeklex mostrou-se significativamente melhor, pois 56,1% concordaram ou concordaram completamente e apenas 29,3% discordaram ou discordaram completamente. Avaliando mais cuidadosamente, observamos que 31,7% concordaram plenamente e apenas 7,3% discordaram completamente.

Embora os resultados relativos à relevância tenham apontado para uma superioridade do Seeklex, isso já era esperado, considerando que a base de dados do Google, por ser muito mais abrangente, está mais sujeita a fornecer resultados irrelevantes para consultas sobre tópicos da legislação. Esse balanço entre relevância e cobertura nos levou a analisar mais profundamente essa relação.

Para tentar entender o que levou algumas pessoas a considerar os resultados do Google mais relevantes do que os do Seeklex, realizamos alguns cruzamentos de dados e concluímos que, muitas vezes, quando uma pessoa não conseguia satisfazer através do Seeklex sua necessidade de informação (cobertura), ela normalmente avaliava negativamente também a relevância. Essa constatação pode ser observada pela similaridade entre curvas apresentadas no gráfico de radar da Figura 4.4, abaixo:

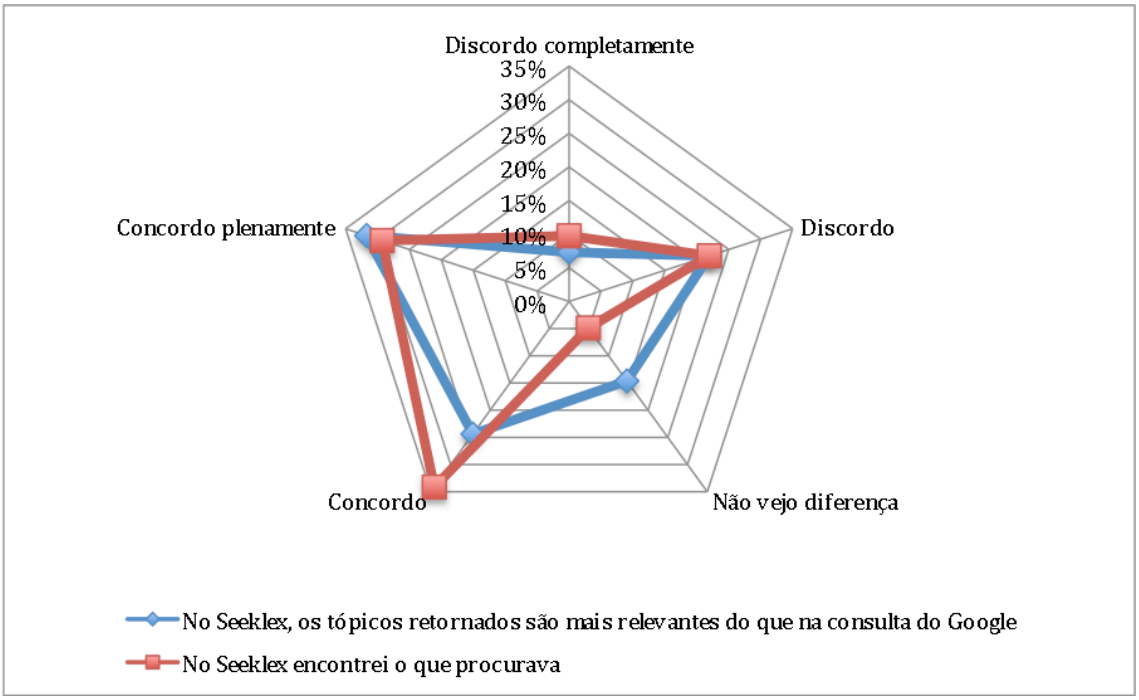

Figura 4.4 – Cobertura influenciando fortemente na comparação da relevância.

Convém ressaltar que a precisão é dada pela expressão: número de documentos relevantes encontrados dividido pelo número total de documentos encontrados. Se nenhum documento é retornado – se a cobertura é zero –, a precisão não pode ser definida, pois a fórmula resultaria em uma divisão por zero. Acreditamos portanto que, se o Seeklex possuísse uma base mais abrangente de normas legais, os resultados de relevância seriam ainda melhores.

4.7.5 – Visualização, organização e interação

A segunda parte do questionário visava a avaliar a adequação do Seeklex aos critérios de visualização, organização e interação com o usuário propostos por Zhang (2008).

Em relação à busca textual, o Seeklex obteve excelentes resultados nos quesitos propostos: a simplicidade e a facilidade de reformulação da busca, como pode ser visto na Figura 4.5, abaixo:

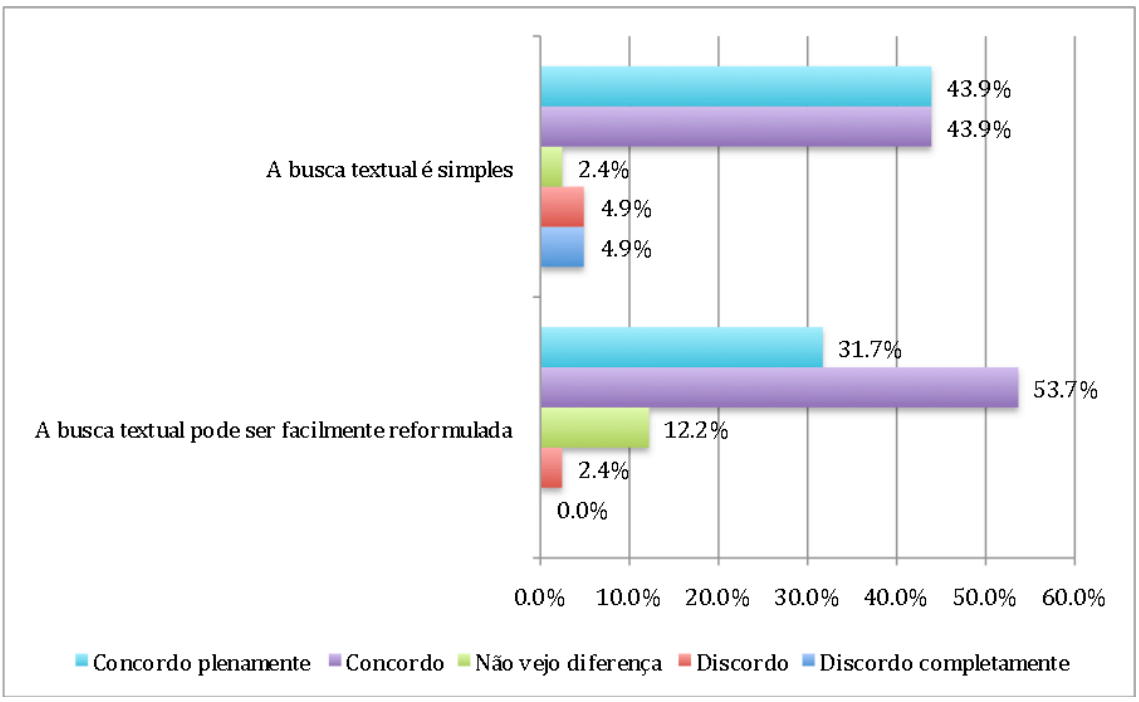

Figura 4.5 – Busca Textual.

As diretivas propostas para navegação também foram muito bem avaliadas pelos respondedores, sendo muito baixo o número de discordâncias ou discordâncias completas em relação à correta implementação destas diretivas. Os resultados obtidos podem ser apreciados na Figura 4.6. abaixo:

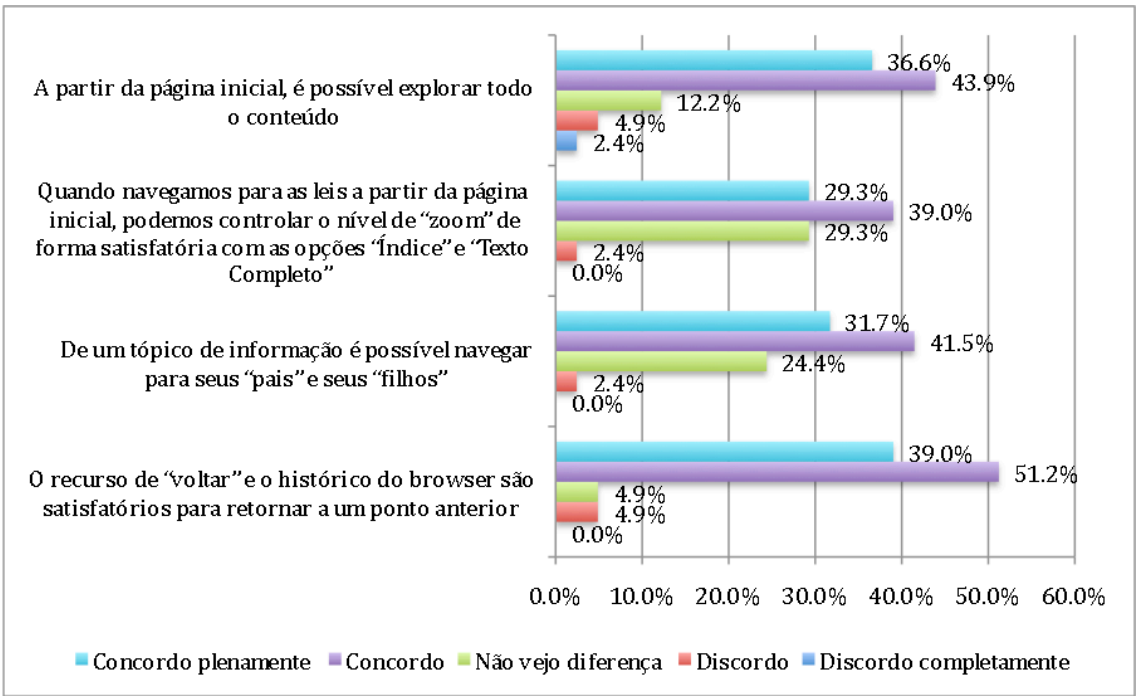

Figura 4.6 – Navegação.

No que tange aos critérios de organização e interatividade, também obtivemos

avaliações muito positivas, demonstrando que no Seeklex a utilização do sistema é simples e intuitiva, que os tópicos apresentados estão bem identificados e que a organização hierárquica implementada no sistema parece ser, para buscas em normas jurídicas, mais adequada do que a organização linear utilizada pelo Google. A Figura 4.7, abaixo, apresenta a avaliação destes critérios:

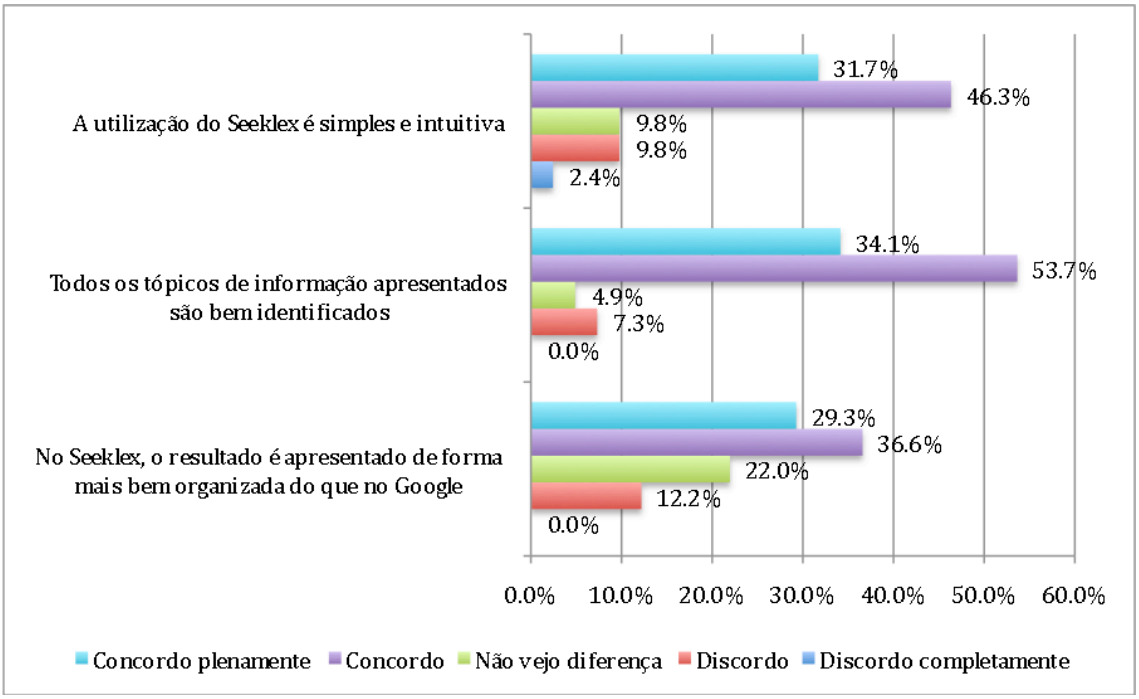

Figura 4.7 – Interface e Organização.

4.7.6 – Considerações finais

Neste capítulo expomos o ambiente no qual todos os algoritmos que compõem o Seeklex foram implementados.

Apresentamos, também, as dificuldades enfrentadas no processo de obtenção da base de normas jurídicas, que nos levaram a desenvolver um parser flexível e até a fazer adaptações manuais nos documentos legais disponibilizados pelo governo.

Além disso, detalhamos os esforços que foram realizados no sentido de melhor adaptar o Seeklex aos critérios de visualização, organização e interatividade descritos pela literatura.

No que tange à avaliação do estudo de caso, apresentamos todos os resultados obtidos e procuramos interpretá-los e justificá-los. Consideramos que os resultados foram bastante positivos, demonstrando que existe sentido em utilizar um mecanismo de buscas baseado em subdocumentos hierarquizados para atender às necessidades de informações relativas a normas jurídicas.

Capítulo 5 – Conclusão

Nesta dissertação procuramos desenvolver métodos específicos para realizar busca e recuperação de informações em subdocumentos hierarquizados, em especial nas normas jurídicas.

Realizamos uma pesquisa bibliográfica visando a conhecer o estado da arte em relação à busca em subdocumentos e constatamos que não existe uma solução única. Para cada necessidade de informação (no nosso caso a busca em leis) é necessário que a ferramenta utilizada seja cuidadosamente adaptada.

Procuramos nos focar em duas principais estratégias: elaborar um método de busca para encontrar e calcular a relevância de subdocumentos, e desenvolver técnicas adequadas de visualização, organização e interação com os resultados obtidos. Para tanto, foram experimentados diversos algoritmos, e os que apresentaram melhor desempenho foram selecionados para participar de nosso estudo de caso.

Desenvolvemos uma aplicação web que foi disponibilizada no endereço http://www.seeklex.com.br. Esta aplicação necessitou de uma ampla base de dados de normas jurídicas para se tornar útil e, consequentemente, poder ser devidamente avaliada. Investimos consideráveis esforços na criação de um parser e na importação de diversos Códigos Federais para suprir essa necessidade.

O desempenho do Seeklex foi avaliado em uma pesquisa que contou com a participação de 41 estudantes de direito. Eles realizaram buscas no Seeklex e no Google e compararam os resultados. Também responderam a diversas perguntas referentes à facilidade de uso, visualização, organização, interação, etc. Através da tabulação desses questionários, foi constatado que o Seeklex obteve um rendimento superior em diversos aspectos, mesmo possuindo um número muito inferior de normas legais indexadas em sua base de dados. Portanto, no caso das normas jurídicas, o sistema de busca em subdocumentos que desenvolvemos mostrou-se apropriado.

5.1 – Trabalhos futuros

Como possíveis trabalhos futuros desta dissertação, destacamos a exploração de estruturas de dados específicas para armazenar índices invertidos hierarquizados e o desenvolvimento de algoritmos mais otimizados para implementar as estratégias

propostas.

Outra linha de trabalho que pode ser seguida consiste em aplicar as técnicas desenvolvidas para o Seeklex em outras coleções que também possam se beneficiar das estratégias que sugerimos para buscas em subdocumentos.

Referências

BAEZA-YATES, R., NAVARRO, G., 1996, "Integrating contents and structure in text retrieval", ***ACM Sigmod record***, v. 25, n. 1, pp. 67–79.

BAEZA-YATES, R., RIBEIRO-NETO, B., 1999, ***Modern information retrieval***, Boston, USA, Addison-Wesley Longman Publishing Company.

BAMFORD, R., BORKAR, V., BRANTNER, M., et al., 2009, "XQuery Reloaded", In: *Proceedings of the VLDB Endowment*, v. 2, n. 2, pp. 1342–1353, Lyon, França, Agosto.

BARON, J. R., THOMPSON, P. 2007, "The search problem posed by large heterogeneous data sets in litigation: possible future approaches to research". In: *Proceedings of the 11th international Conference on Artificial intelligence and Law*, pp. 141-147, California, USA, Junho.

BIAGIOLI, C., FRANCESCONI, E., PASSERINI, A., MONTEMAGNI, S., SORIA, C. 2005, "Automatic semantics extraction in law documents". In: *Proceedings of the 10th international Conference on Artificial intelligence and Law*, pp. 133-140, Bologna, Italia, Junho.

BUENO, T. C. D., 1999, *Uso da Teoria Jurídica para Recuperação de Jurisprudência Criminal em Sistemas Baseado em Casos*, Dissertação de M.Sc., Universidade Federal de Santa Catarina, Santa Catarina, Brasil.

BURKOWSKI, F. J., 1992, "An Algebra for Hierarchically Organized Text-Dominated Databases", ***Information Processing and Management***, v. 28, n. 3, pp. 333–348.

BURKOWSKI, F. J., 1992, "Retrieval activities in a database consisting of heterogeneous collections of structured text". In: *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 112–125, Copenhague, Dinamarca, Junho.

CHIARAMELLA, Y., 1997, "Browsing and Querying: Two Complementary Approaches for Multimedia Information Retrieval". In: *Hypermedia-Information Retrieval-Multimedia*, pp. 9-26, Universitätsverlag, Konstanz.

CHIARAMELLA, Y., MULHEM, P., FOUREL, F., 1996, "A model for multimedia information retrieval", ***Technical Report Fermi ESPRIT BRA 8134***, Universidade de Glasgow. Disponível em: http://www.dcs.gla.ac.uk/research/projects/fermi/tech_reports/reports/fermi96-4.ps.gz. Acesso em: 20/01/2010.

DUQUE, C. G., 2006, "Uma Abordagem Ontológica para a Indexação de Documentos Eletrônicos através da Utilização de Lingüística Computacional". In: *XI Simpósio Nacional de Letras e Lingüística e I Simpósio Internacional de Letras e Lingüística*. Universidade Federal de Uberlândia, Minas Gerais, pp. 798-804. Disponível em: http://www.filologia.org.br/ileel/artigos/artigo_295.pdf. Acesso em: 16/08/2010.

FRANCESCONI, E., PERUGINELLI, G., 2007, "Searching and retrieving legal literature through automated semantic indexing". In: *Proceedings of the 11th international Conference on Artificial intelligence and Law*, California, USA, pp. 131-139, Junho.

FREITAS-JUNIOR H. R., RIBEIRO-NETO, B. A., VALE, R. F., LAENDER, A. H. F., LIMA L. R. S., 2006, "Categorization-driven cross-language retrieval of medical information", ***Journal of the American Society for Information Science and Technology***, v. 57, n. 4, pp. 501-510.

FUHR, N., GROßJOHANN, K., 2001, "XIRQL: A Query Language for Information Retrieval in XML Documents". In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 172–180. New Orleans, Louisiana, USA, Setembro.

GOOGLE APP ENGINE. Disponível em: http://code.google.com/appengine/docs/whatisgoogleappengine.html, Acesso em: 23/02/2010.

GRINSTEIN, G. G., HOFFMAN, P. E., PICKETT, R. M., LASKOWSKI, S. J., 2002, "Benchmark development for the evaluation of visualization for data mining". In: *Information visualization in data mining and knowledge discovery*, pp. 129–176, San Francisco, California, USA. Julho.

HTML TIDY. Disponível em: http://www.w3.org/People/Raggett/tidy/. Acesso em: 22/02/2010.

JENKINS, J., 2008. "What Can Information Technology Do for Law", ***Harvard Journal of Law & Technology***, v. 21, n. 2, pp. 589-607.

JONES, S. K., 1972, "A Statistical Interpretation of Term Specificity and its Application in Retrieval", ***Journal of Documentation***, v. 28, n. 1, po. 11–21.

JTIDY. Disponível em: http://jtidy.sourceforge.net/. Acesso em: 22/02/2010.

KAZAI, G., LALMAS, M., RÖLLEKE, T., 2001, "A Model for the Representation and Focussed Retrieval of Structured Documents Based on Fuzzy Aggregation". In: *The 8th Symposium on String Processing and Information Retrieval*, pp.123-135, Laguna de San Rafael, Chile.

KAZAI, G., LALMAS, M., RÖLLEKE, T., 2001, "Aggregated Representation for the Focussed Retrieval of Structured Documents". In: *SIGIR 2001 Workshop, Mathematical/Formal Methods in IR*, New Orleans, Louisiana, USA, Setembro.

LALMAS, M., 1997, “Dempster-Shafer's theory of evidence applied to structured documents: modelling uncertainty”. In: *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 110–118, New York, NY, USA.

LIU, S., MCMAHON, C. A., CULLEY, S. J., 2008, “Review article: A review of structured document retrieval (SDR) technology to improve information access performance in engineering document management”, ***Computers in Industry***, v. 59, n. 1, pp. 3–16.

LUCID IMAGINATION. Disponível em: http://www.lucidimagination.com/About/Overview. Acesso em 22/02/2010.

LUCINE. Disponível em: http://lucene.apache.org/java/2_4_1/. Acesso em 22/02/2010.

MANNING, C. D., RAGHAVAN, P., SCHÜTZE, H., 2008, ***Introduction to Information Retrieval***, New York, NY, USA, Cambridge University Press.

MOENS, M. 2005. “Combining structured and unstructured information in a retrieval model for accessing legislation”. In: *Proceedings of the 10th international Conference on Artificial intelligence and Law*, pp. 141-145, Bologna, Italia, Junho.

MXQUERY 0.6. Disponível em: http://mxquery.org/. Acesso em 09/08/2010.

NAVARRO, G., BAEZA-YATES, R., 1997, “Proximal nodes: A model to query document databases by content and structure”, ***ACM Transactions on Information Systems (TOIS)***, v. 15, n. 4, pp. 400–435.

NUNES, A. M., FILETO, R., 2007, “Uma Arquitetura para Recuperação de Informação Baseada em Semântica e sua Aplicação no Apoio a Jurisprudência”. In: *III Escola Regional de Banco de Dados (ERBD)*, Universidade de Passo Fundo, Caxias do Sul, RS, Brasil, Abril.

RIBEIRO-NETO, B., ASSUMPÇÃO, R., 2001, “Recuperação de Documentos Jurídicos Baseada em um Tesauro”. In: *Simpósio Brasileiro de Banco de Dados*, pp. 213-227, Rio de Janeiro, Brasil.

ROSE, D. E., BELEW, R. K., 1989, “Legal information retrieval a hybrid approach”. In *Proceedings of the 2nd international Conference on Artificial intelligence and Law*, pp. 138-146, Vancouver, British Columbia, Canada.

SALTON, G., 1988, ***Automatic text processing : the transformation, analysis, and retrieval of information by computer***, Boston, USA, Addison-Wesley Longman Publishing Company.

SHERIDAN, P., BRASCHLER, M., AND SCHÄUBLE, P., 1997, “Cross-Language Information Retrieval in a Multilingual Legal Domain”. In: *Proceedings of the First European Conference on Research and Advanced Technology For Digital Libraries*, pp. 253-268, Londres, Inglaterra, Setembro.

SHLOMO G., KAMPS J., TROTMAN, A., 2009, "Overview of the INEX 2009 Ad Hoc Track". In: *INEX 2009 Workshop Pre-proceedings*, pp. 18-19, Brisbane, Australia, Dezembro.

STREAMING API FOR XML – STAX. Disponível em: http://stax.codehaus.org/Home. Acesso em: 22/02/2010.

THE APACHE FOUNDATION. Disponível em: http://people.apache.org/~vgritsenko/stats/daily.html. Acesso em 22/02/2010.

THE OPEN SOURCE CENSUS. Disponível em: https://www.osscensus.org/summary-report-public.php. Acesso em 22/02/2010.

VALE, R. F., LIMA, L. R. S., RIBEIRO-NETO, B. A., LAENDER, A. H. F., FREITAS-JUNIOR, H. R., 2001, "Recuperação de Informação em Coleções Médicas Utilizando Categorização Automática de Documentos". In: *Simpósio Brasileiro de Banco de Dados*, pp. 243-258, Rio de Janeiro, Brasil.

WOODLEY, A., GEVA, S., EDWARDS, S. L., 2007, "What XML-IR Users May Want", ***Lecture Notes In Computer Science***, v. 4518, pp. 423-431.

XQUERY 1.0: AN XML QUERY LANGUAGE. Dispnível em: http://www.w3.org/TR/xquery/. Acesso em 09/08/2010.

XQUERY AND XPATH FULL TEXT 1.0. Disponível em: http://www.w3.org/TR/xpath-full-text-10/. Acesso em 09/08/2010.

ZHANG, J., 2008, ***Visualization for information retrieval***, The information retrieval series, v. 23, Berlin, Alemanhã, Springer.

ZHANG, P., KOPPAKA, L., 2007, "Semantics-based legal citation network". In: *Proceedings of the 11th international Conference on Artificial intelligence and Law*, pp. 123-130, California, USA, Junho.

ANEXO I

Seeklex: Pesquisa

Prezado Estudante de Direito, este questionário é parte da pesquisa para minha dissertação do curso de mestrado, no Programa de Engenharia de Sistemas e Computação, na Universidade Federal do Rio de Janeiro. O questionário tem o intuito de avaliar aspectos do mecanismo de busca em normas jurídicas disponível no site www.seeklex.com.br. Portanto, antes de preencher o questionário é necessário visitar o Seeklex e conhecer seu funcionamento. Agradeço a sua participação.

Renato Crivano
Universidade Federal do Rio de Janeiro (UFRJ/COPPE) Departamento de Engenharia de Sistemas e Computação Caixa Postal: 68511 - CEP: 21945-970 - Rio de Janeiro – RJ
renato@cos.ufrj.br / crivano@jfrj.jus.br

*Obrigatório

1) Qual a sua Faculdade/Universidade?

2) Qual período você está cursando? *

3) Qual a sua idade? *

Realizando uma consulta no Seeklex e repetindo a mesma consulta no Google, podemos observar: *

	Discordo comple-tamente	Discordo	Não vejo diferença	Concordo	Concordo plena-mente
4) No Seeklex, encontrei o que procurava					
5) No Google, encontrei o que procurava					
6) No Seeklex, vi mais resultados relevantes do que irrelevantes					
7) No Google, vi mais resultados relevantes do que irrelevantes					
8) No Seeklex, os tópicos retornados são mais relevantes do que na consulta do Google					
9) No Seeklex, o resultado é apresentado de forma mais bem organizada do que no Google					

Com relação à facilidade de utilização do Seeklex: *

	Discordo comple-tamente	Discordo	Não sei	Concordo	Concordo plena-mente
10) Todos os tópicos de informação apresentados são bem identificados					
11) De um tópico de informação é possível					

	Discordo comple-tamente	Discordo	Não sei	Concordo	Concordo plena-mente
navegar para seus “pais” e seus “filhos”					
12) A partir da página inicial, é possível explorar todo o conteúdo					
13) A busca textual é simples					
14) A busca textual pode ser facilmente reformulada					
15) A utilização do Seeklex é simples e intuitiva					
16) Quando navegamos para as leis a partir da página inicial, podemos controlar o nível de “zoom” de forma satisfatória com as opções “Índice” e “Texto Completo”					
17) O recurso de “voltar” e o histórico do browser são satisfatórios para retornar a um ponto anterior					

ANEXO II

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	Sexto	21-25 anos	5	2	5	2	4	5	5	5	5	5	5	5	5	5
	Terceiro	menos de 21 anos	2	3	4	3	2	4	4	4	4	3	3	3	4	4
	Décimo	31-35 anos	4	4	4	4	4	4	4	5	5	5	5	5	5	5
	Quarto	menos de 21 anos	4	4	5	5	3	3	3	4	3	4	3	4	5	4
UFF	Décimo	21-25 anos	5	5	4	4	4	4	5	5	5	5	5	5	4	4
UERJ	Quarto	menos de 21 anos	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Universidade Estácio de Sá	Sétimo	31-35 anos	4	5	4	5	5	5	4	3	3	5	4	3	3	5
Univercidade	Terceiro	31-35 anos	3	3	3	3	3	3	4	4	4	4	4	3	3	3
Cândido Mendes	Sexto	menos de 21 anos	5	5	5	5	5	5	5	5	5	5	5	5	5	5
PUCMINAS	Quinto	menos de 21 anos	2	5	4	1	4	4	5	4	2	2	3	3	4	5
FDSM	Quinto	21-25 anos	4	4	4	2	2	4	4	4	4	4	4	4	3	4
FDSM	Quarto	mais de 40 anos	5	5	5	4	3	5	5	5	5	5	5	5	3	5
Jorge Amado - UNIJORGE	Quinto	26-30 anos	5	5	4	5	4	5	5	5	5	4	4	4	5	5
Universidade Estácio de Sá	Nono	36-40 anos	5	5	5	2	5	5	5	5	5	5	5	5	5	5
UFMG	Oitavo	21-25 anos	2	4	2	2	4	4	4	4	3	4	4	4	4	4
UNIFEMM	Sétimo	26-30 anos	5	5	4	4	5	5	5	4	5	4	4	4	4	4
UNIFEMM	Sétimo	21-25 anos	2	4	4	2	2	4	4	4	4	4	4	4	4	4
Universidade Católica do Salvador	Décimo	21-25 anos	4	4	4	2	5	4	4	4	4	4	4	4	3	4
FDSM	Sétimo	mais de 40 anos	4	2	4	1	4	4	4	4	4	4	4	4	4	4
UFMG	Oitavo	21-25 anos	1	5	2	5	2	4	4	4	4	1	4	4	4	2
Universidade Católica do Salvador	Oitavo	21-25 anos	2	4	1	4	1	2	4	2	2	5	2	2	5	4
Universidade Estácio de Sá	Sétimo	26-30 anos	4	4	4	4	5	5	5	5	4	5	4	5	3	2
UNIFEMM	Quinto	21-25 anos	2	2	2	2	2	3	4	4	4	4	4	4	3	4

FDSM	Quinto	menos de 21 anos	2	4	3	3	4	3	4	4	3	2	3	2	3	3
Universidade Católica do Salvador	Nono	21-25 anos	1	1	1	4	1	2	2	3	4	4	5	1	2	5
Universidade Católica do Salvador	Nono	21-25 anos	2	4	4	3	2	2	4	3	1	4	3	2	4	5
UNIFEMM	Quinto	21-25 anos	5	4	5	2	5	5	5	5	5	5	5	5	5	5
Universidade Católica do Salvador	Décimo	31-35 anos	3	4	3	3	3	4	4	3	3	4	4	4	3	4
UFMG	Oitavo	21-25 anos	4	4	4	4	2	3	3	3	4	4	4	4	4	4
Universidade Católica do Salvador	Nono	21-25 anos	2	5	2	2	4	3	4	3	4	5	4	4	5	4
FUMEC	Quarto	26-30 anos	4	4	3	3	3	3	4	4	4	4	4	4	3	4
Universidade Católica do Salvador	Décimo	21-25 anos	4	5	5	2	3	4	4	3	5	4	4	4	3	4
Centro Universitário Newton Paiva	Oitavo	21-25 anos	5	5	5	5	5	5	5	5	5	5	5	5	4	5
FDSM	Quinto	21-25 anos	1	5	1	5	1	3	4	3	4	5	5	5	4	4
FUNCEC	Sétimo	21-25 anos	5	5	5	4	5	5	5	5	5	5	5	5	5	5
UFMG	Sexto	21-25 anos	5	4	5	4	5	5	5	5	5	5	5	4	4	4
FDSM	Segundo	36-40 anos	5	5	3	3	5	3	5	5	5	5	5	5	5	5
Faculdade de Ciências Jurídicas de Diamantina	Oitavo	26-30 anos	4	4	4	2	5	2	2	3	4	5	4	5	4	5
FDSM	Segundo	26-30 anos	4	4	4	4	2	4	2	4	4	4	4	4	4	4
Universidade Católica do Salvador	Sétimo	21-25 anos	4	4	4	4	5	4	4	3	5	5	4	4	5	5
Universidade Católica do Salvador	Sétimo	31-35 anos	1	4	2	2	2	2	4	4	4	1	4	2	3	4

Legenda:

1) Discordo completamente
2) Discordo
3) Não sei/Não vejo diferença
4) Concordo
5) Concordo plenamente