UNIVERSIDADE FEDERAL DE SANTA CATARINA

TECNOLOGIAS DA INFORMAÇÃO E COMUNICAÇÃO

ÁLVARO WILLIAM DA CONCEIÇÃO

UM SISTEMA VOLTADO AO ARMAZENAMENTO E RECUPERAÇÃO DE CONTEÚDO TEXTUAL DE DIFERENTES CONTEXTOS

Araranguá, 26 de fevereiro de 2013

ÁLVARO WILLIAM DA CONCEIÇÃO

UM SISTEMA VOLTADO AO ARMAZENAMENTO E RECUPERAÇÃO DE CONTEÚDO TEXTUAL DE DIFERENTES CONTEXTOS

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Tecnologias da Informação e Comunicação. Sob a orientação do Professor Alexandre Leopoldo Gonçalves.

Araranguá, 2013

Álvaro William da Conceição

UM SISTEMA VOLTADO AO ARMAZENAMENTO E RECUPERAÇÃO DE CONTEÚDO TEXTUAL DE DIFERENTES CONTEXTOS

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina, como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Tecnologias da Informação e Comunicação.

Professor Alexandre Leopoldo Gonçalves, Dr.
Presidente da Banca - Orientador

Professora Olga Yevseyeva, Dra
Membro

Professor Eugênio Simão, Dr.
Membro

Araranguá, 26 de fevereiro de 2013

Dedico este trabalho a todos que contribuíram direta ou indiretamente em minha formação acadêmica.

AGRADECIMENTOS

Agradeço a todos que direta ou indiretamente contribuíram no decorrer desta caminhada, especialmente:

A Deus, a quem devo minha vida.

A minha família que sempre me apoiou nas escolhas tomadas.

Ao orientador Prof. Alexandre Leopoldo Gonçalves que teve papel fundamental na elaboração deste trabalho e de toda a minha caminhada acadêmica.

Aos meus colegas pelo companheirismo e disponibilidade para me auxiliar, incentivar e apoiar em vários momentos difíceis.

RESUMO

Atualmente a quantidade de informação gerada cresce exponencialmente, sendo que uma quantia significativa de informações encontra-se em formato não estruturado, ou seja, textos. Com o grande volume de informações disponíveis, seja na web ou mesmo nas organizações, coletar e disponibilizar essas informações constitui-se em um desafio computacional. Superando os obstáculos, a área de Recuperação de Informação (RI) pode constituir uma importante ferramenta para a integração e a disponibilização de conteúdo. Neste sentido, surgem os Sistemas de Recuperação de Informação (SRIs) que promovem meios de se localizar documentos ou informações dentro de documentos através de uma solicitação do usuário. Esses documentos podem estar disponíveis na Web ou em uma organização. Com esse pressuposto é apresentado neste trabalho um Sistema de Recuperação de Informação visando o armazenamento de conteúdos textuais com diferentes contextos. Propondo atingir esse objetivo, o trabalho baseia-se na definição das visões lógica e física com a capacidade de suportar o desenvolvimento de um sistema que possibilite a recuperação de documentos textuais. A demonstração de viabilidade é realizada através da implementação de um protótipo que possui os serviços de indexação e consulta. A aplicação do protótipo em um cenário permitiu demonstrar que o sistema proposto é capaz de obter resultados consistentes e satisfatórios, tanto para a indexação quanto para a consulta de documentos.

Palavras-chave: Recuperação de Informação; Sistema de Recuperação de Informação, Modelo Vetorial; Índice Invertido.

ABSTRACT

Nowadays, the amount of information increases exponentially, whereas a significant part of this is in unstructured format, that is, text. With the large volume of information available, whether on the web or even in organizations collect and make this information available becomes a computational challenge. Overcome obstacles, the area of Information Retrieval (IR) can be an important tool for the integration and delivery of content. In this sense, arise Information Retrieval Systems (IRSs) aiming to promote ways to locate documents or information within documents base on a user request. These documents may be available on the web or in an organization. With this assumption is presented in this work an Information Retrieval System to store textual content from different contexts. In order to achieve this goal, the work is based on the definition of the logical and physical views to support the development of a system that enables the retrieval of textual documents. A feasibility demonstration is carried out through the implementation of a prototype containing indexing and searching services. The use of the prototype in a scenario has demonstrated that the proposed system is able to obtain consistent and satisfactory results for indexing and searching documents.

Keywords: Information Retrieval; Information Retrieval System, Vector Space Model; Inverted Index.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

BT – *Broader Term*

CSS – *Cascading Style Sheets*

CPF – *Cadastro de Pessoas Físicas*

HTML – *Hypertext Markup Language*

HTTP – *Hypertext Transfer Protocol*

J2EE – *Java 2 Platform Enterprise Edition*

JDBC – *Java Database Connectivity*

JSON – *JavaScript Object Notation*

JSF – *Java Server Faces*

JSP – *Java Server Pages*

NT – *Narrower Term*

PHP – *Hypertext Preprocessor*

PDF – *Portable Document Format*

RI – *Recuperação de Informação*

RT – *Related Term*

SRIs – Sistemas de *Recuperação de Informação*

SQL – *Structured Query Language*

URL – *Uniform Resource Locator*

XML – *eXtensible Markup Language*

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos, vem sendo observado um crescimento na quantidade de informação tanto na web quanto nas organizações. Lyman (2003) afirma que isso tem ocorrido em escala exponencial. Segundo Bovo (2011), tal situação se deve, principalmente, aos avanços nas tecnologias da informação e comunicação que permitem não somente lidar com esse aumento, mas também tornar as informações cada vez mais acessíveis.

Mesmo antes do advento da internet já era possível perceber um grande aumento na quantidade de informação em diferentes áreas acadêmicas e mesmo nas organizações em geral. Segundo Himma (2007), já em 1987 a quantidade de informação técnica e/ou científica dobrava a cada 5 a 15 anos. Exemplo disso é a base MEDLINE, um banco de dados bibliográfico com artigos científicos publicados nas áreas de ciências biomédicas (medicina, farmácia, etc.), que contém mais 22[1] milhões de registros.

Lyman (2003) afirmava que o mundo estaria produzindo entre um e dois *exabytes*[2] de informação em um único ano, ou seja, cerca de 250 megabytes por cada pessoa na terra. Outro aspecto que chama a atenção foi relatado por Hilbert e Lopez (2011), em que estes afirmam que a informação armazenada globalmente tem um aumento anual de 23%. Contudo, a capacidade da humanidade em agregar/incorporar essa informação para si própria é de apenas 6%.

A isso se somam as informações das organizações disponíveis como relatórios técnicos, manuais sobre procedimentos e/ou softwares, registros de sugestões/reclamações de clientes, entre outros.

Com isso surgem duas situações: a primeira se refere à interferência da informação no processo de tomada de decisão, uma vez que se torna difícil agir com grandes quantidades

[1] http://www.nlm.nih.gov/bsd/revup/revup_pub.html#med_update, acessado em 10 de junho de 2012.

[2] Um *exabyte* representa um bilhão de Gigabytes.

de informação, muitas vezes dispersas. A segunda é mais estruturante, pois surgem problemas no estabelecimento de soluções definitivas em como localizar, armazenar e disponibilizar essa informação de maneira fácil e rápida. (BEPPLER, 2008).

Segundo Ren e Bracewell (2009), nas décadas de 60 e 70, os computadores começaram a ter poder de processamento suficiente para lidar com a recuperação da informação e ter resultados satisfatórios. Com o início da Internet, a área de Recuperação de Informação (RI) tornou-se cada vez mais relevante e pesquisada. Atualmente, tornou-se comum que pessoas com a necessidade de localizar determinada informação na web se utilize de sistemas de recuperação de informação (motores de busca), como por exemplo, o Google™ e o Bing™. Esse cenário é também cada vez mais comum dentro das organizações com a implantação e utilização de sistemas de busca corporativa.

A busca corporativa (Enterprise Search) nada mais é que um sistema de recuperação de informação dentro de uma empresa. No entanto, os colaboradores das empresas vêm criando expectativas devido à eficácia de suas experiências satisfatórias em motores de busca na Web. Neste sentido, essa ferramenta está tornando-se cada vez mais importante devido à necessidade de manipulação de grande quantidade de dados no contexto empresarial. (THROOP, 2006; DEMARTINI, 2007).

Com o objetivo de prover Sistemas de Recuperação de Informação (SRIs) adequados surgem três desafios principais sendo:

a) a capacidade de armazenar documentos com estruturas diferenciadas;

b) a possibilidade de integrar informações estruturadas (vindas de bancos de dados relacionais, por exemplo) e não estruturadas (documentos em geral);

c) a capacidade de interpretar adequadamente consultas realizadas por usuários e a partir disso recuperar documentos que sejam de interesse desses usuários.

Segundo Manning (2009), a Recuperação de Informação objetiva facilitar o acesso aos documentos de maior relevância conforme a necessidade de informação do usuário, em que, essa necessidade é, normalmente, expressa por meio de uma consulta a qual se utiliza palavras-chave. A recuperação de informação nesse contexto consiste basicamente na determinação de quais documentos de uma coleção contém as palavras-chave da consulta realizada pelo usuário. A dificuldade está não somente em localizar os documentos, mas também em decidir a sua relevância.

Nesse sentido, Russell e Norvig (2004), afirmam que um sistema de recuperação de informação tem como características:

a) armazenar uma quantidade de documentos;

b) atender a demandas de consulta através de uma linguagem de consulta;

c) encontrar um conjunto de resultados;

d) apresentar um conjunto de resultados que atenda à consulta.

1.1 PROBLEMATIZAÇÃO

Um fator importante refere-se aos múltiplos contextos que determinada informação, pertencente a um domínio em particular, pode obter. Um documento qualquer terá: título, data, resumo e texto principal. Já um Curriculum Vitae, terá informações não estruturadas, o título e o resumo das publicações de determinado autor, por exemplo, mas também terá outras informações estruturadas ou semiestruturadas que podem servir para recuperar determinada informação.

Estes desafios podem ser considerados clássicos na área, mas existem outros, principalmente se considerada a questão de integração de informação, seja ela estruturada, vinda de bancos de dados relacionais, por exemplo, ou não estruturada, na forma de textos. No contexto da informação não estruturada um documento qualquer terá título, data, resumo e texto principal. Já um Curriculum Vitae, terá além de informações não estruturadas, como o título e o resumo das publicações de determinado autor, outras informações estruturadas que podem servir para recuperar determinada informação, tais como, o ano de determinada publicação ou o curso de formação da pessoa vinculada ao currículo.

Deste modo, permitir que estas informações residam em um mesmo repositório promove melhor desempenho nas consultas, uma vez que todo o conteúdo necessário para a apresentação já está disponível no repositório. Além disso, permite a realização de consultas mais complexas através da combinação de buscas em texto completo e em campos que representam informações estruturadas. Um exemplo seria uma consulta com o termo “Recuperação de Informação” que consta no resumo ou texto completo, mas também a indicação de um ano específico ou intervalo de anos.

Nesse sentido, projetar Sistemas de Recuperação de Informação capazes de agregar em um único repositório, informações de diferentes contextos, pode constituir em importante ferramenta para a integração e a disponibilização de conteúdo para determinada organização.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo principal do trabalho é desenvolver um sistema computacional que possibilite o armazenamento integrado de informação estruturada e não estruturada de diferentes domínios, assim como, a disponibilização desse conteúdo.

1.2.2 Objetivos Específicos

Visando atingir o objetivo principal, alguns objetivos específicos são requeridos, entre eles:

- Estudar e avaliar *frameworks* e soluções para a Recuperação de Informação;
- Propor funcionalidades básicas voltadas à indexação e recuperação de diferentes conteúdos textuais que possibilitem atender demandas de um sistema de recuperação de informação.
- Propor os modelos lógicos e físicos, que permitam suportar o desenvolvimento de um Sistema de Recuperação de Informação;
- Implementar um protótipo de Sistema de Recuperação de Informação para demonstrar a viabilidade da funcionalidade de indexação e recuperação de conteúdo textual.
- Testar e discutir a implementação das funcionalidades, assim como, discutir os resultados obtidos através da utilização do protótipo de Sistema de Recuperação de Informação.

1.3 METODOLOGIA

Para atingir os objetivos deste trabalho de conclusão de curso os procedimentos metodológicos foram assim divididos:

- Revisão da literatura científica sobre Recuperação de Informação suas principais definições, fluxos e processos, estrutura de dados e modelos;
- Demonstração da viabilidade do trabalho através do desenvolvimento de um protótipo que contenha as funcionalidades de indexação e consulta;

- Desenvolvimento de uma camada que possibilite a comunicação entre as funcionalidades de indexação e consulta e uma *interface* de Sistema de Recuperação de Informação;
- Detalhamento das funcionalidades e discussão de um cenário de aplicação;
- Apresentação das considerações finais do trabalho, assim como potenciais pontos de aprimoramento futuro.

1.4 ESTRUTURA DO TRABALHO

O trabalho está dividido em cinco capítulos: o primeiro capítulo, a Introdução, apresenta um aspecto geral do trabalho; o Capítulo 2 faz uma revisão bibliográfica abordando os conceitos que permearão este trabalho. De modo geral, o capítulo aborda as principais definições, os fluxos, os processos (pré-processamento, indexação) e os modelos para o desenvolvimento de um Sistema de Recuperação de Informação; o Capítulo 3 apresenta a visão lógica e física visando detalhar o desenvolvido do trabalho; no Capítulo 4 são apresentadas as funcionalidades de indexação e consulta e apresentação de um cenário de aplicação; por fim, o Capítulo 5 apresenta as considerações finais e as possíveis evoluções do trabalho.

2. RECUPERAÇÃO DE INFORMAÇÃO

Manning (2009) afirma que a recuperação de informação pode ser definida como a busca de documentos não estruturados, que satisfaçam uma necessidade de informação a partir de grandes coleções de documentos que, geralmente são armazenadas em computadores.

Recuperação de informação (RI), de acordo com Manning (2009), "é a área de pesquisa que se preocupa com a estrutura, análise, organização, armazenamento, recuperação e busca de informação". Desta forma, ela deve permitir que usuários obtenham acesso à informação esperada de maneira facilitada.

Os sistemas de recuperação de informação lidam com a procura de documentos ou informações dentro de documentos através de uma solicitação do usuário. Esses documenos podem estar disponíveis na Web ou em uma organização. Para viabilizar o processo de recuperação de informação torna-se necessário a utilização de ferramentas, chamadas motores de busca (LIAW; HUANG, 2004) Considerando o crescente aumento das informações disponíveis, a elaboração de motores de busca capazes de atender de maneira satisfatória usuários em suas buscas torna-se um desafio. Contudo, para que o mecanismo de busca possa cumprir o seu papel existe um processo até que determinado documento esteja disponível à consulta do usuário. (CARDOSO, 2000; GREENGRASS, 2000). A Figura 1 apresenta o processo de recuperação da informação que se divide em três grandes fases, sendo, pré-processamento, indexação e consulta.

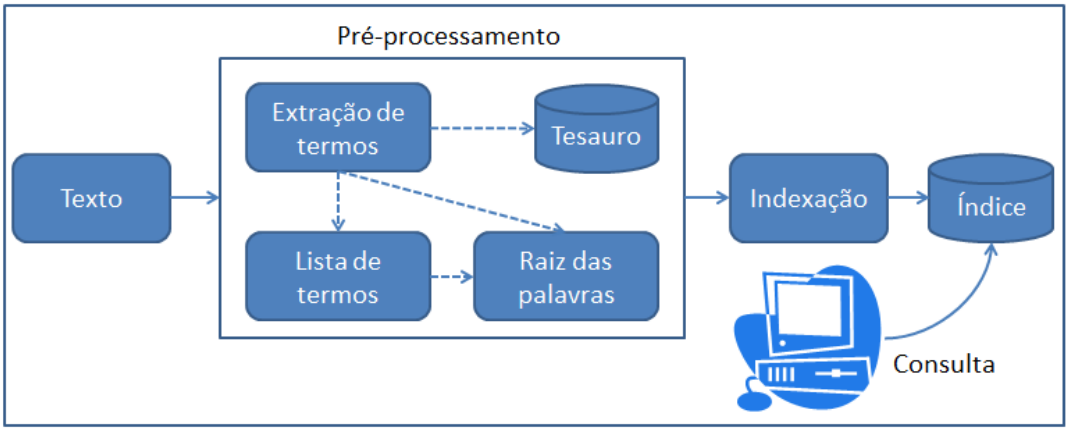


Figura 1- Processo de Recuperação de Informação

O primeiro passo, chamado de pré-processamento, consiste em um conjunto de operações sobre o texto antes que este seja enviado ao passo de indexação. Inicialmente, ocorre a extração de todas as palavras dos documentos que compõem o *corpus* (coleção de documentos). Esse processo é chamado de extração de termos (*tokeninzation*) e consiste na quebra/partição do texto em palavras, símbolos ou qualquer outro elemento significativo, identificado como um termo (*token*). Após a extração dos termos, existe um conjunto de operações que são opcionais dependendo do domínio e requisitos do sistema de recuperação de informação.

Entre essas operações adicionais pode-se citar o uso de estruturas de conhecimento formais, tais como dicionários controlados, taxonomias ou tesauros, para filtrar termos relevantes ao processo e/ou para expandir determinado contexto de um documento adicionando termos relacionados e sinônimos. De modo geral, essas estruturas representam determinado conhecimento de domínio e vão de uma simples relação de palavras, no caso de dicionários controlados, até estruturas mais complexas como tesauros que são representados por uma hierarquia de conceito e relacionamentos associativos. A próxima seção do trabalho irá detalhar a estrutura de um tesauro.

Além das estruturas formais acima citadas, o pré-processamento pode ainda se utilizar de uma lista de termos comuns (*Stop list*) para efetuar a limpeza do texto e a extração de radicais (*Stemming*). No primeiro caso, uma lista de termos comuns é utilizada buscando retirar palavras que não são adequadas para expressarem a essência de determinado documento, tais como artigos e preposições. Já na extração de radicais, são aplicados algoritmos que analisam determinada palavra tentando reduzir a mesma para o seu radical. Um dos algoritmos mais utilizados foi proposto por Porter (1980) em que é utilizado um conjunto e regras para se atingir a raiz de uma palavra eliminando-se seus sufixos. Esses

algoritmos possuem uma dependência em relação a uma língua exigindo assim, regras específicas.

Em seguida, ocorre o passo de indexação onde são criados os índices que armazenam e mapeiam o conteúdo (palavras, números, termos em geral) criando assim, uma ligação de cada um dos termos com um conjunto de documentos em que estes ocorrem. No índice também são armazenadas informações adicionais, que facilitam e otimizam o processo de localização e determinação da relevância de documentos. Entre essas informações encontram-se a frequência e as posições em que o termo ocorre no documento. Com isso, torna-se possível o cálculo da relevância do termo no documento o que facilita e melhora o desempenho de um sistema de recuperação de informação ao atender determinada requisição de busca de um usuário.

As próximas seções abordam em mais detalhes as fases de pré-processamento e indexação.

2.1 PRÉ - PROCESSAMENTO

2.1.1 Tokenização

Tokenização é uma técnica essencial para o processo de RI. Isso se justifica, uma vez que a maioria dos métodos de RI (por exemplo, os baseados em índices invertidos), não executam buscas completas em texto de maneira eficiente. Em geral, são utilizados algoritmos de busca de *strings* para localizar, em um índice estruturado, um conjunto de termos pré-definidos obtidos a partir da coleção de documentos indexada. Um termo de busca que não exista no índice não irá produzir uma resposta, ou seja, não irá recuperar documentos. Suponha que um termo “negócioepreço” tenha sido adicionado para o índice, ao invés dos termos separados “negócio” e “preço”. Desse modo, uma busca por **negócio preço** que, frequentemente, é executado por um usuário humano com espaço não será capaz de produzir qualquer resultado relevante. Nesse sentido, o mesmo processo de tokenização deve ser utilizado tanto na indexação quanto na consulta de modo que se obtenham resultados consistentes (WU, 2011).

Segundo Manning (2009), a tokenização é realizada a partir de uma sequência de caracteres e tem por função a extração de termos, ou seja, separar o texto em pedaços chamado de *tokens*. Durante o processo determinados elementos do texto, tais como pontuação, podem ser descartados. Esses *tokens* são muitas vezes tratados como termos ou

palavras, mas é importante fazer uma distinção ainda que eles possam ser lexicamente idênticos. Um *token* é uma instância de uma sequência de caracteres em algum documento particular, enquanto termos e palavras possuem um contexto semântico, ou seja, referem-se a conceitos de um domínio. Por exemplo: o termo "Recuperação de Informação" seria representado no índice textual através de três *tokens*, um para cada sequência de caracteres.

Barcala (2002) diz também que a tokenização pode conter um sub-módulo considerando abreviaturas, siglas, números com casas decimais, ou datas em formato númerico. Para esse efeito, são utilizados dois dicionários (um das abraviaturas e outro de acrónimos) além de um conjunto de regras para detectar números e datas.

2.1.2 Stop list

Na recuperação de informação, uma *stop list* é usada na indexação automática para filtrar as palavras sem finalidade real no conteúdo do documento, chamadas de *stop words*. Além disso, essas palavras podem ocupar uma grande fração do texto. Em torno de 20 a 30 por cento dos *tokens* do documento, o que pode ser visto como uma fração considerável de um documento. Eliminando essas palavras no inicio da indexação deixa o processamento mais rápido e não danifica a eficácia da recuperação da informação (ZHOU, 2011).

Manning (2009) diz que uma *stop list* é uma lista em que os seus elementos são ignorados no processo de indexação. Com o seu uso pode-se reduzir significativamente o número de termos durante a indexação e também o tempo de indexação. Em geral, as palavras que constam na *stop list* não são relevantes em uma consulta, impactando pouco na precisão do resultado. Afirmação similar é declarada por Zhou (2011). Contudo, os mecanismos de busca atuais, principalmente aqueles projetados para a Web, geralmente não usam *stop lists*, visto que buscam explorar de maneira mais precisa a semântica da língua (KORPHAGE, 1997; MANNING, 2009). Por exemplo, o termo "Vitamina A" contém uma palavra que consta na *stop list* e, no processo de indexação, o *token* "A" seria removido. Isto promove uma redução na precisão das consultas, uma vez que qualquer documento mencionando o token "Vitamina" seria recuperado e não somente os documentos que tivessem o termo "Vitamina A".

2.1.3 Stemming

O processo de stemming é utilizado para melhorar a abordagem semântica da consulta a fim de obter cada vez mais documentos relacionados e relevantes. Manning (2009)

refere-se a *stemming* como um processo heurístico que serve para cortar os extremos das palavras, que muitas vezes inclui na remoção de afixos derivacionais. Em função do contexto gramatical, documentos são escritos usando-se diferentes formas de uma palavra, por exemplo, organizar, organizando e organização. Alem disso, há familias de palavras derivadas relacionadas com significados semelhantes, como estudante, estudantes, estudar e estudando. Algumas vezes, é util para a busca através de uma dessas palavras retornar documentos que contenham outras palavras no conjunto. Aplicando o processo de *stemming* essas quatro palavras são reduzidas para um único radical que no caso é estuda, que será computando 4 vezes.

Um dos principais algoritmos de *stemming* é o algoritmo de Porter (PORTER, 1980), em que são utilizadas uma série de 5 regras na transformação das palavras. As regras são divididas em etapas e são examinadas em seqüência, e apenas uma regra de uma etapa pode ser aplicada. O sufixo mais longo possível é sempre removido por causa da ordenação das regras dentro de uma etapa. Por exemplo, se o termo entrada for “dependent”, ele é reduzido a “depend” (FRAKES; BAEZA-YATES, 1992). O algoritmo de Porter é dependente da língua de modo que se torna necessária a utilização de regras específicas para cada língua.

Outra técnica de possível é chamada de n-*gram*. Essa técnica não produz uma real derivação (*stemming*) de uma palavra, mas sim o cálculo, indicando o quão semelhantes são duas palavras quaisquer. Pode ser estendido para calcular, por meio de uma matriz de similaridade, a força da relação (proximidade) entre cada par de termos em um documento. Uma vez que a matriz seja obtida, esta pode ser usada para projetar um agrupamento de palavras semelhantes. Nessa abordagem, embora determinada palavra não possua derivações, as palavras constantes no agrupamento podem funcionar como tal. A deficiência dessa abordagem reside no fato de que para cada palavra a ser "derivada" deve haver um procedimento para identificar seu agrupamento correspondente e, assim, o seu representante (NASCIMENTO, 1998).

2.1.4 Tesauro

Segundo Han (1998), tesauros consistem em conceitos ligados em nós que representam as relações entre eles, podendo ter alguns termos mais amplos, mais restritos, sinônimos, e termos relacionados. Várias tentativas têm sido realizadas envolvendo tesauros na avaliação de consultas, principalmente, no aumento da qualidade dos documentos que são recuperados. Essas tentativas incluem:

1) Utilizar a distância conceitual entre os termos de um tesauro;

2) Capturar a intenção do usuário de forma inteligente, explorando os tesauros construídos por várias técnicas de representação semântica, como redes e quadros.

A interpretação semântica do objeto de um tesauro pode ocorrer a partir de duas perspectivas: a perspectiva de conceito e a perspectiva de relacionamento. Na perspectiva de conceito, em contraste com tesauros convencionais que tratam um nó como um conceito atômico, o tesauro de sinônimos vê um conceito como uma classe de objeto, tendo outros conceitos mais especializados, como sub-conceitos ou instâncias. Uma vez que, os sub-conceitos podem, por sua vez, ter suas próprias instâncias, um conceito de nível superior constitui uma hierarquia de conceitos de classe juntamente com os seus sub-conceitos. Já na perspectiva de relacionamento, o tesauro de sinônimos refina relações tradicionais com BT[3]/NT[4]e RT[5] na generalização, especialização, agregação e associação segundo a sua semântica.

Entre os projetos de maior sucesso na elaboração de um tesauro encontra-se o WordNet™ vinculado à universidade de Princeton. O WordNet é um banco de dados léxico em inglês inspirado na teoria psicolinguística (MILLER et al., 1990; HUA, 2011) em que os termos estão interligados por meio de ligações conceituais semânticas e léxicas. WordNet vai além de um tesauro tradicional, na medida em que palavras são agrupadas de acordo com seus significados.

A Figura 2 apresenta uma representação simples da estrutura de um tesauro.

[1] BT (Broader Term): Símbolo utilizando para indicar um relacionamento de generalização.

[4] NT (Narrower Term): Símbolo utilizado para indicar um relacionamento de especialização.

[5] RT (Related Term): Símbolo utilizando para designar um relacionamento de associação entre os termos.

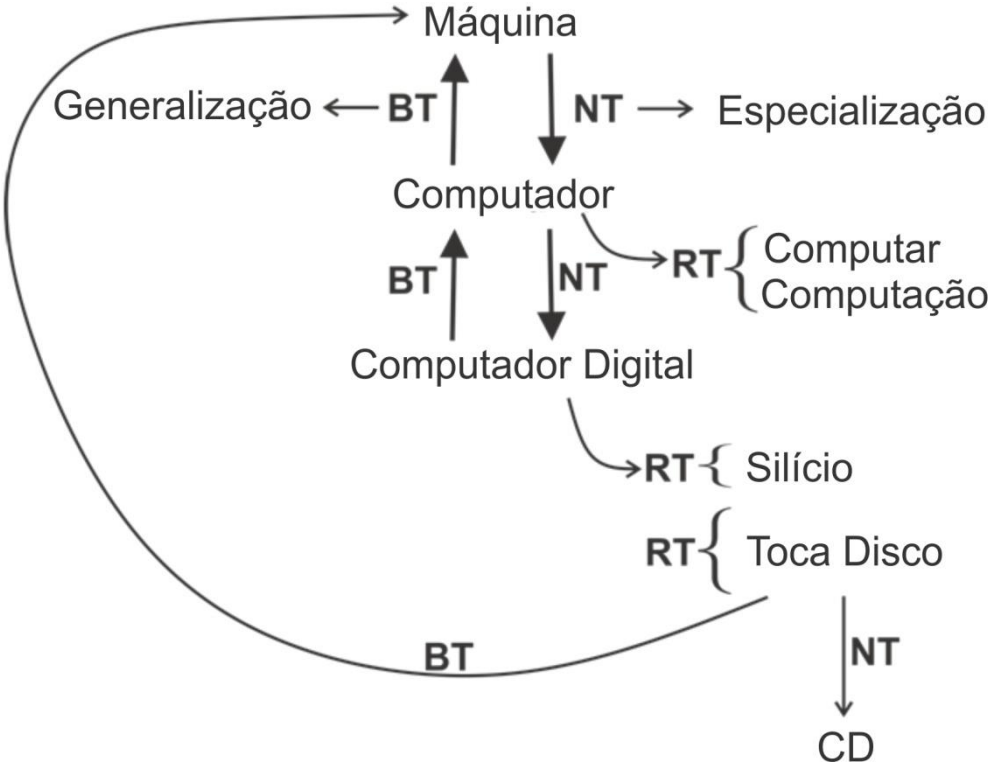


Figura 2 - Estrutura de um tesauro

2.2 PROCESSO DE INDEXAÇÃO

O processo de indexação é a etapa em que a partir do documento é extraído o seu conteúdo original, e este é então usado para criar uma instância do mesmo em um formato apropriado, chamado de vetor do documento. O conjunto desses vetores irá formar uma estrutura chamada de índice invertido.

Segundo Harman (1992), existem várias estruturas que podem ser usadas na implementação de um índice invertido como Matrizes Ordenadas, Árvores B e Tabelas *Hash* ou até mesmo a combinação destas estruturas.

Na Matriz Ordenada o índice armazena a lista de palavras-chaves, incluindo o número de documentos associados a cada palavra-chave e um *link* para os documentos que contenham aquela palavra-chave. Essa matriz é, geralmente, acessada utilizando-se um padrão de pesquisa binária. Considerando que, para manter a eficiência da pesquisa binária, torna-se necessária uma lista ordenada, a utilização dessa abordagem pode ser uma desvantagem se utilizada sobre grandes índices.

As Árvores B utilizam prefixos de palavras como chaves primárias no seu índice que são adequadas para armazenamento de índices textuais. Cada nó da árvore tem um número

variável de chaves. Cada chave representa um prefixo que distingue as chaves armazenadas no próximo nível, em que, o último nível armazena as palavras-chave com as demais informações necessárias, tais como, o apontamento para os documentos e a frequência e um documento específico. Em comparação com matrizes ordenadas, as árvores-B utilizam mais espaço; no entanto, as atualizações são muito mais fáceis de serem realizadas e o tempo de busca é geralmente mais rápido, especialmente se o armazenamento secundário é usado para o índice invertido em vez da memória principal. A implementação do índice invertido usando árvores-B é mais complexo do que usando matrizes ordenadas.

Por outro lado, a Tabela *Hash* é baseada em uma distribuição uniforme de valores por um intervalo de endereço de tabela. Para a atribuição de um valor utiliza-se uma função *Hash*. Neste método os registros armazenados em uma tabela são diretamente endereçados a partir de uma transformação aritmética sobre a chave de pesquisa, constituindo-se em um processo de duas etapas principais:

- Computar o valor da função de transformação (função *hashing*), a qual transforma a chave de pesquisa em um endereço de tabela;
- Considerando que duas ou mais chaves, podem ser transformadas para o mesmo endereço deve existir um mecanismo para tratar colisões.

Uma função de transformação deve mapear, através de uma função de transformação, chaves em inteiros dentro do intervalo [0..M-1], de modo que M é o tamanho da tabela. A função de transformação ideal é aquela que atenda aos seguintes requisitos: (a) seja simples de ser computada; e (b) para cada chave de entrada, qualquer chave de saída é igualmente provável de ocorrer.

De modo geral, um índice invertido serve para o armazenamento e mapeamento de conteúdo (palavras, números, *termos* em geral) criando uma ligação com um conjunto de documentos. Existem basicamente duas vertentes: sendo uma em que, somente se guarda a referência do(s) documento(s) para cada palavra e outra em que, armazena-se também a posição (*offset*) de cada palavra dentro de um documento em particular (WAN, 2009).

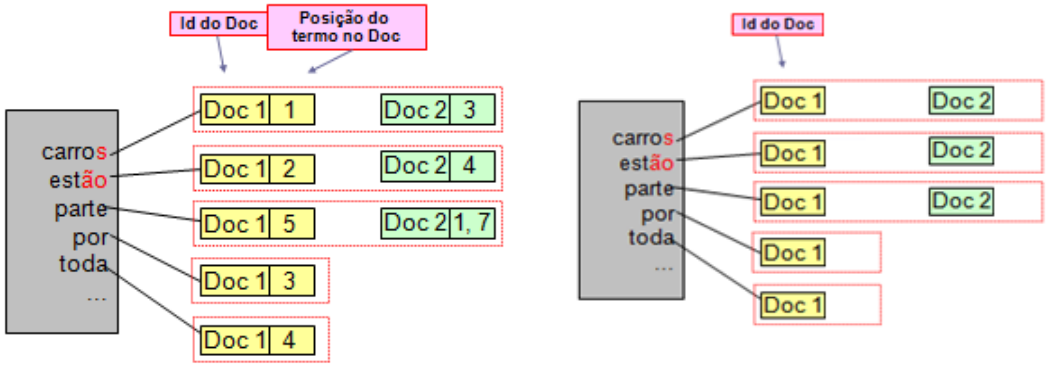
Segundo Mccreadie (2010), para cada termo, o índice invertido contém uma lista de postagem, que lista os documentos com um número inteiro como identificador do documento (doc IDs) - contendo o termo. Cada entrada na lista de postagem também armazena informações suficientes visando agilizar o processo de recuperação e ordenação de cada documento, como a frequência das ocorrências e possivelmente informações sobre a posição

dos termos dentro de cada documento. Essas informações permitem, por exemplo, que em uma determinada pesquisa seja levada em consideração a ordem das palavras (também chamada de frase) e/ou a proximidade dos termos, ou seja, a quantidade de *tokens* que separam os termos informados pelo usuário em uma consulta.

A Figura 3 mostra duas formas na qual o índice pode ser construído. O índice A (Figura 3a) têm dois campos sendo o identificador do documento e a posição do termo em cada documento. Já o índice B (Figura 3b) contém apenas o identificador do documento.

(Doc 1) Carros estão por toda parte.

(Doc 2) Parte dos carros estão na maior parte das ruas.



(a) (b)

Figura 3 - Exemplos de índices invertidos

2.3 MODELOS DE RECUPERAÇÃO DE INFORMAÇÃO

De modo geral, os modelos de recuperação de informação servem como estratégias de busca de documentos relevantes a partir de uma necessidade de consulta. Entre esses modelos considerados clássicos destacam-se o modelo booleano, o modelo vetorial e o modelo probabilístico. Neste trabalho serão detalhados os modelos booleano e vetorial.

2.3.1 Modelo booleano

O modelo booleano tipicamente aplica as funções booleanas para o processamento dos termos localizados em qualquer posição dentro dos documentos. Os operadores usados no modelo booleano são AND, OR e NOT. (SALTON; EDWARD; WU, 1983; JOON, 1995; KOWALSKI, 2002; KANA, 2000; KORFHAGE, 1997; POHL; ZOBEL; MOFFAT, 2010).

O modelo booleano é binário, ou seja, utiliza valores 0 ou 1, indicando respectivamente a ausência ou presença do termo no conteúdo textual. Nesse modelo, a

semântica da consulta é bem definida, cada documento corresponde à expressão booleana, ou não. O modelo com operadores booleanos foi muito utilizado até a década de 90, principalmente, em ferramentas de buscas comerciais, devido a semântica simples e ao cálculo direto dos resultados utilizando operações de conjunto. Na Tabela 1 é apresentado um exemplo de matriz documento-termo utilizada no modelo booleano.

Documentos	Termo 1	...	...	Termo *k*	...	...	Termo *n*
d1	1	...	...	1	...	...	0
...	...	...	...	...	...	...	...
d*j*	0	...	...	1	...	...	0
...	...	...	...	...	...	...	...
d*m*	0	...	...	1	...	...	1

Tabela 1- Matriz documento-termo utilizada pelo Modelo Booleano

A Tabela 2 apresenta um exemplo base para a utilização do modelo booleano sendo t1, t2 e t3 termos de documentos, d1 até d5 os documentos que contêm estes termos e *q* a consulta desejada pelo usuário. O valor 1 indica a presença do termo, no documento ou consulta, enquanto o valor 0 indica a ausência.

Sendo que *q* = t1 AND (t2 OR (NOT t3)).

	t1	t2	t3
d1	1	0	0
d2	1	1	0
d3	1	0	1
d4	0	1	0
d5	1	1	1
q	1	1	1

Tabela 2 - Exemplo de utilização do modelo booleano reunindo a relação documento-termo e a consulta

A Figura 4 é a representação da Tabela 2, em que são apresentadas as relações entre os documentos e seus respectivos termos.

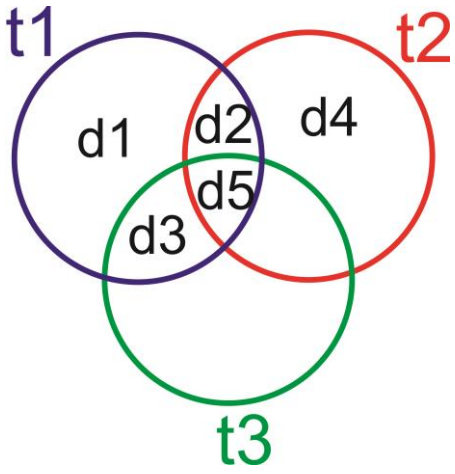


Figura 4 - Representação dos documentos para os termos

A partir da Tabela 2 - Exemplo de utilização do modelo booleano reunindo a relação documento-termo e a consultae considerando uma consulta *q* = t1 AND (t2 OR (NOT t3)) os documentos d1, d2 e d5 seriam retornados. A Tabela 3 apresenta de maneira detalhada os passos para se atingir o resultado da consulta.

					X	q
	t1	t2	t3	NOT t3	tT2 OR (NOT t3)	t1 AND x
d1	1	0	0	1	1	1
d2	1	1	0	1	1	1
d3	1	0	1	0	0	0
d4	0	1	0	1	1	0
d5	1	1	1	0	1	1
Q	1	1	1			

Tabela 3 - Exemplo de consulta do modelo booleano

O principal problema do modelo booleano se refere à formulação adequada de uma consulta. Isso ocorre principalmente se o domínio da informação não for bem conhecido, uma vez que o conjunto resultante não tem tamanho definido, isto é, pode não conter itens, mas pode também conter milhares. Outro problema é a falta de um grau de comparação, não sendo possível ordenar os resultados de acordo com a relevância ou mesmo estabelecer níveis de importância dos documentos o que possibilitaria reduzir o conjunto resultante de documentos.

2.3.2 Modelo Vetorial

O modelo vetorial é um modelo estatístico, em que um documento é representado como um vetor de termos num espaço euclidiano ***n***-dimensional (***n*** é o número de termos), no

qual cada dimensão representa um termo extraído do documento em questão. Cada termo pode ter um peso associado para representar seu grau de importância. Essa atribuição de pesos deve refletir a importância de cada termo no contexto do documento e também da coleção de documentos (ZHANG; GUAN; ODBAL, 2012; SMITH; NAPHADE; NATSEV, 2003; GAN, 2008).

A posição do documento em cada dimensão é dada pelo peso do termo associado àquela dimensão. Portanto, a distância entre dois documentos indica o grau de similaridade entre ambos, ou seja, documentos com os mesmos termos estão localizados em uma mesma região do espaço e, teoricamente, possuem conteúdos similares. (JONES; FURNAS,1987; YUBO; XING et al., 2011).

Para a obtenção do peso de um termo em um documento utiliza-se em geral a equação *tf-idf*. Esta equação está baseado em dois pressupostos, sendo:

- Quanto maior é a frequência de uma palavra em um documento, maior é sua relevância para o tópico daquele documento;
- Quanto maior é a frequência de uma palavra nos documentos de uma coleção, menor é seu fator discriminante entre os documentos dessa coleção.

A equação do *tf-idf* necessita de algumas informações pertinentes aos documentos em si, mas também referentes à coleção de documentos para que o peso (*w*) de um termo *t* em um documento particular seja calculado, sendo:

$$w = ntf \times idf$$

de forma que, *ntf* representa a frequência do termo (*tf*) em um documento específico normalizada pela máxima frequência do termo neste mesmo documento (tf_{max})

$$ntf = \frac{tf}{tf_{max}}$$

e, *idf* representa o *log* da divisão do número total de documentos que fazem parte do *corpus* (*N*) pela quantidade de documento que mencionam o termo em questão *df*.

$$idf = \log \frac{D}{df}$$

Nas Tabelas 4, 5 e 6 são apresentados exemplos dos cálculos dos pesos, tendo três documentos: d1, d2, d3 e 6 termos: t1, t2, t3, t4, t5, t6.

D1	*tf*	tfmax	df	tf-idf	w = ntf-idf
t1	20	100	1230	0.41723723	0.153489036
t2	100	100	467	2.506774378	0.92216695
t3	99	100	100000	0.174330346	0.064130895
t4	15	100	15	0.6	0.220721966
t5	50	100	10000	0.58804563	0.216324313
t6	80	100	60	2.718352007	1

Tabela 4 - Exemplo do cálculo do peso para o documento d1

D2	tf	tfmax	dt	tf-idf	w = ntf-idf
T1	50	50	1230	2.086186148	0.80516292
T2	16	50	467	0.802167801	0.309596423
T3	33	50	100000	0.116220231	0.044855163
T4	48	50	300	2.591011204	1
T5	26	50	10000	0.611567455	0.236034276
T6	2	50	60	0.1359176	0.052457357

Tabela 5 - Exemplo do cálculo do peso para o documento d2

D3	tf	Tfmax	dt	tf-idf	w = ntf-idf
T1	30	75	1600	0.788788511	1
T2	5	75	862	0.149372266	0.18936922
T3	12	75	5681	0.227466635	0.28837468
T4	75	75	90000	0.22184875	0.28125251
T5	39	75	6523	0.708054818	0.89764849
T6	8	75	96	0.340674136	0.43189541

Tabela 6 - Exemplo do cálculo do peso para o documento d3

Com os pesos definidos, pode-se realizar a similaridade através do cosseno Θ, entre o vetor dos termos pesquisados (vetor de consulta) e o vetor dos termos de determinado documento, sendo.

$$\cos\theta = \frac{\sum_{i=1}^{n}(d_i \times q_i)}{\sqrt{\sum_{k=1}^{n}(d_k)^2}\ \ x\ \sqrt{\sum_{j=1}^{n}(q_j)^2}}$$

Ao passo que, d_i e d_k representam as frequências normalizadas (peso) dos i_{th} e k_{th} termos do vetor de documentos *d*, e q_i e q_j são as frequências dos i_{th} e j_{th} termos do vetor *q*. Na equação abaixo o vetor de consulta será designado por "*q*" e terá apenas dois termos: (T1, T5)

com peso igual a 1. Os demais termos possuem peso igual a 0. A seguir é apresenta a equação do cosseno:

O cosseno é calculado pelo produto dos pontos entre o vetor *d* e *q*, dividido pela raiz quadrada do somatório de todos os termos do documento ao quadrado e multiplicando pela raiz quadrada do somatório de todos os termos da consulta. Nas Tabela 7, Tabela 8 e Tabela 9 são apresentados os cálculos entre o vetor de consulta e os três documentos do exemplo anterior.

i	d1	d2	d3	q	q x d1	q x d2	q x d3
t1	0.153489036	0.805163	1	1	0.153489036	0.805163	1
t2	0.92216695	0.309596	0.189369	0	0	0	0
T3	0.064130895	0.044855	0.288375	0	0	0	0
T4	0.220721966	1	0.281253	0	0	0	0
T5	0.216324313	0.236034	0.897648	1	0.216324313	0.236034	0.897648
T6	1	0.052457	0.431895	0	0	0	0
Soma					0.369813349	1.041197	1.897648

Tabela 7 – Produto dos pontos entre os vetores *d* e *q*

	D1	D2	D3	Q
Quadrado	0.023558884	0.648287328	1	1
Quadrado	0.850391883	0.095849945	0.035861	0
Quadrado	0.004112772	0.002011986	0.08316	0
Quadrado	0.048718186	1	0.079103	0
Quadrado	0.046796208	0.05571218	0.805773	1
Quadrado	1	0.002751774	0.186534	0
Soma	1.973577934	1.804613213	2.19043	2
Raiz	1.404840893	1.343358929	1.48001	1.414214

Tabela 8 – Somatório e raiz de cada um dos vetores

cosϴ (q,d1)=	0.186140315
cosϴ (q,d2)=	0.548057248
cosϴ (q,d3)=	0.9066425

Tabela 9 - Similaridade com o vetor de consulta

Com o resultado dos cálculos nota-se que entre os documentos comparados com o vetor de consulta *q* o documento *d3* possui a maior similaridade seguido pelos documentos d2 e d1, respectivamente.

A Figura 5 apresenta o gráfico do cosseno indicando a similaridade entre os documentos e o vetor de consulta detalhando no exemplo anterior.

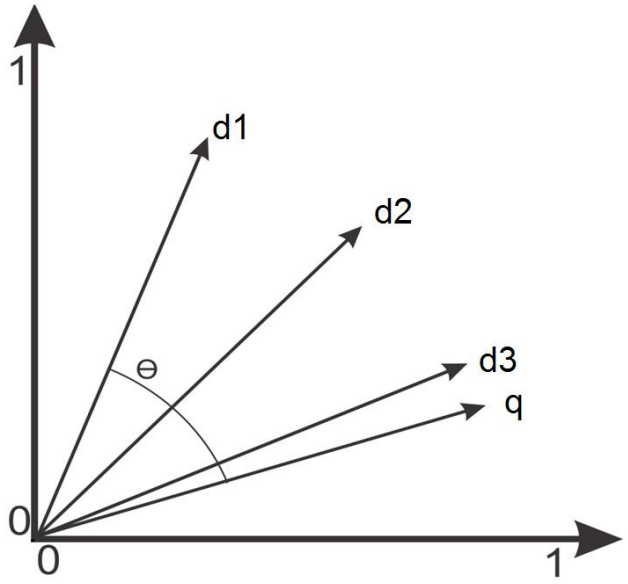


Figura 5 - Ilustração da representação vetorial do cálculo do cosseno com base no exemplo anterior

Porém, o modelo vetorial também apresenta problemas, de modo que um deles é a representação vetorial é a alta dimensionalidade que pode ocorrer em casos nos quais o número de palavras distintas (incluindo ainda variações de estilo de escrita e erros de grafia) é consideravelmente grande numa coleção de documentos.

2.4 FRAMEWORKS

Segundo Larmam (2007), um *framework* é um conjunto de classes abstratas e concretas em que as interfaces entre elas fornecem algum tipo de serviço para um subsistema lógico. *Framworks* raramente são aplicações propriamente ditas. As aplicações são geralmente constituídas pela integração de vários *frameworks*. (GAMMA, 2000).

Fayad e Schmidt (1997) citam três tipos de *frameworks*:

- *Framework*s de infra-estrutura de sistemas: Esses *frameworks* apoiam o desenvolvimento de infra-estrutura de sistemas, tais como comunicações, interfaces com o usuário e compiladores.
- *Frameworks* de integração de *middleware*: Consistem em um conjunto de padrões e classes de objetos associados que apoiam a comunicação e troca de informações entre componentes.

- *Frameworks* de aplicações empresariais: São dedicados a domínios de aplicações específicos. Eles incorporam conhecimento de domínio e apóiam o desenvolvimento de aplicações de usuários finais.

Na recuperação de informação existem alguns *frameworks* voltados à criação de aplicações de indexação e recuperação de conteúdo textual. Um dos mais utilizados atualmente é o Lucene™ e será detalhado na próxima seção.

2.4.1 Lucene

O *framework* Lucene™[6] é um *framework* de código aberto escrita em Linguagem Java™ voltado ao desenvolvimento de aplicações de recuperação de informação, ou seja, a indexação e consulta de documentos textuais. Nos últimos anos, Lucene tornou-se muito popular e, atualmente, vem sendo utilizado em vários projetos[7]. Lucene pode ser integrado com outras linguagens de programação incluindo Delphi, Perl, C #, C + +, Python, Ruby e PHP. De modo geral, pode ser intergrado com qualquer linguaguem ou aplicação através de serviços web.

Um dos principais fatores responsáveis pela popularidade do Lucene é sua simplicidade, seu alto desempenho e tambem por não se preocupar com a fonte dos dados, seu formato, ou mesmo a sua linguagem. Desse modo, o tratamento do texto para lidar com múltiplos formatos (PDF, HTML, XML, entre outros) e a origem da fonte de informação (Bancos de Dados Relacionais ou Orientados a Objetos, Web, Sistemas de Arquivo) devem ser realizados antes do processo de indexação. Vale mencionar ainda que o Lucene é baseado no modelo vetorial para a recuperação de documentos.

2.4.1.1 Classes de Indexação

No Lucene, para ser realizada a indexação existe uma sequência de passos de maneira que para cada um destes passos são necessárias classes que devem ser utilizadas. A Figura 6 ilustra uma simplificação desse processo em que é necessário primeiro o preenchimento de uma instância da classe *Document* com campos (instâncias da classe *Field*) com os respectivos conteúdos, passando pelo processo de atualização do índice (*IndexWriter*) em um base final (*Directory*).

---

[6] Projeto Lucene <http://lucene.apache.org/>

[7] Lista de aplicações e sites web que utilizam Lucene <http://wiki.apache.org/lucene-java/PoweredBy>

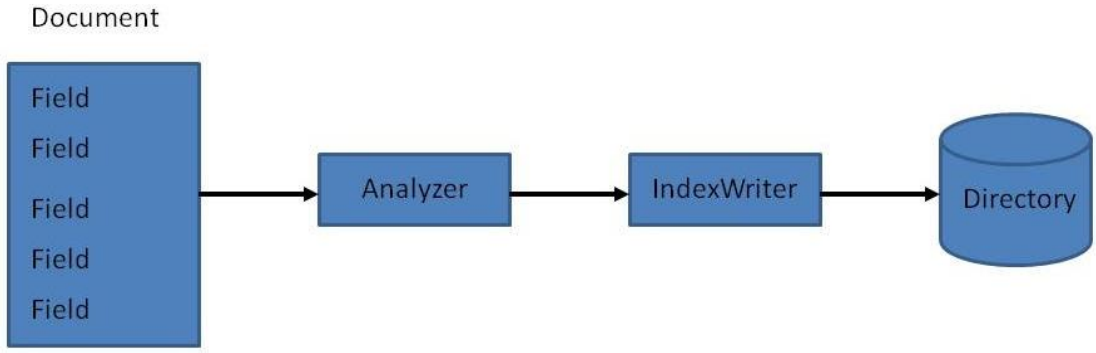


Figura 6 – Classes utilizadas no processo de indexação através de Lucene

Fonte: Hatcher (2009).

A seguir são detalhadas as principais classes necessárias do Lucene para o processo de indexação (HATCHER, 2009):

Document: uma classe *Document* é uma representação de uma coleção de campos (*fields*) em que tais campos representam a estrutura associada a este documento. Por exemplo, em um documento textual deve-se, através de um analisador, extrair as informações estruturais (autor(es), título, palavras-chave, resumo, texto completo) e atribuir cada uma dessas informações a campos separados. Desse modo, dependendo da necessidade de determinada aplicação será possível a realização de buscas completas por cada um desses campos.

Field: cada documento em um índice contém um ou mais *fields* encapsulados pela classe Field. Apesar de não ser comum, um documento pode ter mais de um *field* com o mesmo nome. Nesse caso, a indexação ocorre na ordem em que os campos (*fields*) foram adicionados. Ao pesquisar por determinado conteúdo nos textos, todos os *fields* iguais são concatenados e tratados como um único campo de texto.

Analyzer: antes do texto ser indexado ele é processado por um analisador. O trabalho do analisador consiste em: a) extrair as ‘palavras’ (tokens) que serão indexados; b) descartar determinados tokens irrelevantes para o processo (por exemplo, *stop words*); e c) realizar transformações diversas (por exemplo, conversão para letras minúsculas e a retirada de acentuação). Em geral permite realizar qualquer funcionalidade de analise de texto que seja programável.

IndexWriter: é o principal componente no processo de indexação. Através deste é realizada a criação de um novo indice ou abertura para escrita/leitura de um índice já

existente. Uma vez que o índice esteja disponível (aberto) torna-se possível a realização das operações de adição, remoção e atualização de documentos.

Directory: representa o local propiamente dito em que fica armazenado o índice. A classe *Directory* é uma classe abstrata e possui algumas implementações disponíveis no Lucene. Entre as implementações cita-se a *FSDirectory* que é uma classes que possibilita o acesso dos arquivos do índice no sistema operacional, e a *RAMDirectory* que é uma classes que mantém os dados do índice em memória RAM.

2.4.1.2 Classes de Consulta

Além das classes disponíveis para lidar com o processo de indexação, o Lucene fornece um conjunto básico de classes que permitem a recuperação dos documentos constantes na base de índices, entre elas, citam-se as classes *IndexSearcher*, *Term*, *Query*, *TermQuery*, *TopDocs*. (HATCHER, 2009).

IndexSearcher: é a classe responsavel por procurar o que o *IndexWriter* havia indexado. Ela é responsávem por abrir o índice em modo de leitura para que consultas sejam realizadas.

Term: é a unidade básica de pesquisa, similar ao *Field* utilizado na indexação. Consiste em um par de *strings* composto pelo nome do campo a pesquisar e o valor a ser pesquisado.

Query: é uma classe abstrata que fornece a base para outras classes de consulta especializadas, assim como, métodos que possibilitam configurar adequadamente determinada consulta.

TermQuery: é o tipo mais básico e primitivo de *Query* existente no Lucene sendo utilizado para encontrar documentos que contenham em seus campos o conteúdo indicando no termo (*Term*) que faz parte da consulta.

TopDocs: é o conjunto de resultados de sua busca em que os *N* documentos mais relevantes são retornados. O resultado será ordenado pela relevância dos documentos em relação à consulta.

Além disso, o retorno de uma consulta pode conter outras informações importantes, por exemplo, o conteúdo utilizado na busca em destaque (*highlight*) para apresentação ao usuário.

3. SOLUÇÃO PROPOSTA

3.1 INTRODUÇÃO

Neste capítulo, são analisadas e discutidas a composição do sistema proposto para recuperação de informação considerando uma visão lógica e uma visão física. A visão lógica apresenta o funcionamento geral do sistema, enquanto a visão física promove o detalhamento dos componentes tecnológicos utilizados no desenvolvimento da solução.

Tanto a visão lógica quanto a visão física são dividas em camadas com responsabilidades bem definidas, envolvendo a coleta de dados, a indexação, os serviços que permitem a escrita e a leitura da estrutura de índice e a aplicação cliente.

3.2 VISÃO LÓGICA

A visão lógica representada pela Figura 7 apresenta as funcionalidades do sistema em camadas com o objetivo de detalhar as fases e /ou componentes de coleta, indexação, serviços de consulta e indexação e a aplicação que acessa os dados.

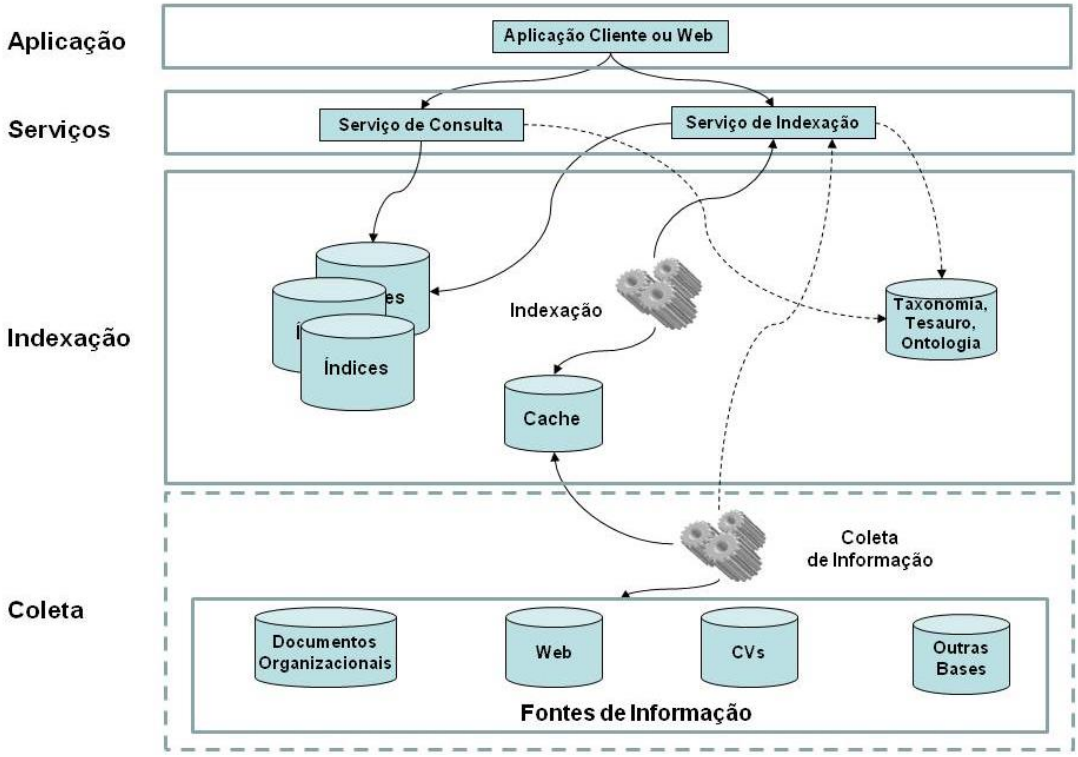


Figura 7 – Visão lógica do sistema proposto

3.2.1 Coleta de Dados

A primeira etapa em um sistema de recuperação de informação envolve a coleta de dados. Nessa etapa são selecionados os textos que irão fazer parte da coleção de documentos. No que se refere à origem dos documentos, eles podem ser obtidos das mais variadas fontes de informação, por exemplo, em documentos organizacionais, currículos vitae e na Web.

A coleta de dados é, em geral, realizada de maneira automática por meio de *crawlers*. Um *crawler* é um sistema que faz visitas em todo e qualquer documento disponível nas fontes de informação em que este pode inspecionar. Depois de coletar um *crawler* envia as informações coletadas para serem armazenadas em uma base de dados (*cache*) que depois serão inspecionadas e utilizadas na indexação por outro processo. Além disso, as informações coletadas podem ser enviadas diretamente ao serviço de indexação.

Vale mencionar que, apesar da fase de coleta estar presente na visão lógica, ela não faz parte dos objetivos do trabalho proposto. Ela consta como uma indicação de possibilidade de desenvolvimento futuro.

3.2.2 Indexação

Após a etapa de coleta de dados, a próxima fase é a indexação. Nessa etapa, o mecanismo ou processo de indexação vai até o *cache* e analisa os documentos que ainda não foram indexados, enviando estes ao serviço de indexação que destina o conteúdo para uma estrutura de índice invertido que permite a realização de consultas textuais.

3.2.3 Serviços

3.2.3.1 Serviço de Indexação

O serviço de indexação possui um papel fundamental no SRI proposto. Por meio dele é possível armazenar documentos em um formato apropriado que possibilite a rápida obtenção destes em um momento futuro, determinado por uma consulta.

Analisando a visão lógica é possível verificar mais de uma possibilidade de interação, em que os documentos a serem indexados chegam ou através do processo de indexação ou através do processo de coleta de dados.

Uma vez que o serviço tenha recebido determinado documento é necessária a conversão para um formato apropriado antes de enviá-lo ao índice. A Figura 8 demonstra o fluxo de conversão de um texto bruto até a representação no índice. De modo geral, existe um processo de conversão, chamado de *parser*, que realiza transformações sobre o texto, entre estas: a) obtenção do texto bruto utilizando um extrator compatível com o formato do arquivo, por exemplo, pdf, doc, xls, entre outros; b) a limpeza do texto, tais como, a retirada de acentuação e a eliminação de *stop words* (palavras frequentes encontradas em uma determinada língua); e c) representação para o formato adequado.

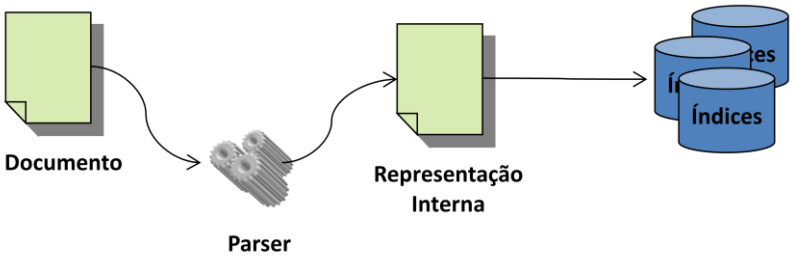


Figura 8 - Fluxo de execução do serviço de indexação

Durante o processo de indexação pode ser adequado considerar somente termos que residam em uma estrutura formal de representação do conhecimento, tal como, um dicionário

controlado, uma taxonomia, um tesauro ou uma ontologia. Este processo é chamado de indexação temática.

Uma possibilidade é a utilização de um tesauro para a indexação temática em que quatro componentes são necessários: o próprio tesauro, o termo indexador, o termo preferido e o termo não preferido. Termos indexadores são termos simples ou compostos que representam um conceito no tesauro. Para indexar os conceitos são usados os termos preferidos que estão organizados hierarquicamente e podem ou não serem os próprios termos indexadores. Os termos não preferidos são relacionados à estrutura hierárquica por meio de sua referência ao termo preferido, enquanto termos preferidos são relacionados uns aos outros por relações que definem a hierarquia. Por exemplo, pode-se citar as palavras "carro", "automóvel", "veículo", que poderiam ser agregadas em uma só palavra, ou termo preferido, "carro". Esta abordagem pode reduzir em muito o tamanho do índice. Contudo, criar, manter e evoluir estruturas formais de conhecimento se tornam um desafio a medida que aumenta a expectativa de mapeamento de um domínio em específico.

3.2.3.2 Serviço de Consulta

O serviço de consulta visa permitir a consulta à coleção de documentos indexados por meio de palavras-chave que representam determinada necessidade pela informação.

A Figura 9 demonstra o fluxo de interação com o serviço de consulta. A partir da necessidade do usuário o serviço é invocado recebendo um conjunto de palavras-chave de interesse. Essas palavras-chave são então utilizadas para recuperar os documentos mais relevantes de modo que estes possam ser apresentados ao usuário como retorno à requisição.

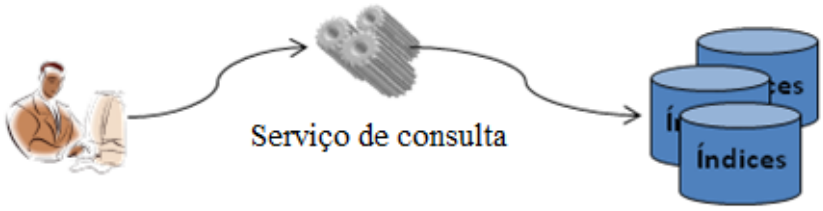


Figura 9 - Fluxo de interação com o serviço de consulta

Vale mencionar que, o serviço de consulta deverá realizar uma verificação na estrutura formal de conhecimento caso esta tenha sido utilizada durante o processo de indexação. Isso é necessário para que as mesmas transformações realizadas e que estão refletidas no índice textual sejam realizadas pelo serviço de consulta garantindo a correta recuperação dos documentos.

3.2.4 Aplicação

Por último, encontra-se a camada da aplicação que deverá interagir com os serviços de indexação e de consulta. Pensando na indexação pode-se vislumbrar um processo que constantemente analisa o *cache* e envia as novas informações (documentos) para o índice por meio do serviço de indexação.

Por outro lado, considerando o serviço de consulta pode-se pensar em uma aplicação Web que permite a interação com o usuário de modo que este possa expressar sua necessidade de consulta por meio de palavras-chave e filtros que são aplicados ao índice textual. Como resultado, a aplicação recebe um conjunto de documentos que são apresentados ao usuário, que pode então, inspecioná-los.

3.3 VISÃO FÍSICA

A visão física representada pela Figura 10, promove um detalhamento das funcionalidades do sistema e como ocorre a interconexão das mesmas por meio dos componentes tecnológicos.

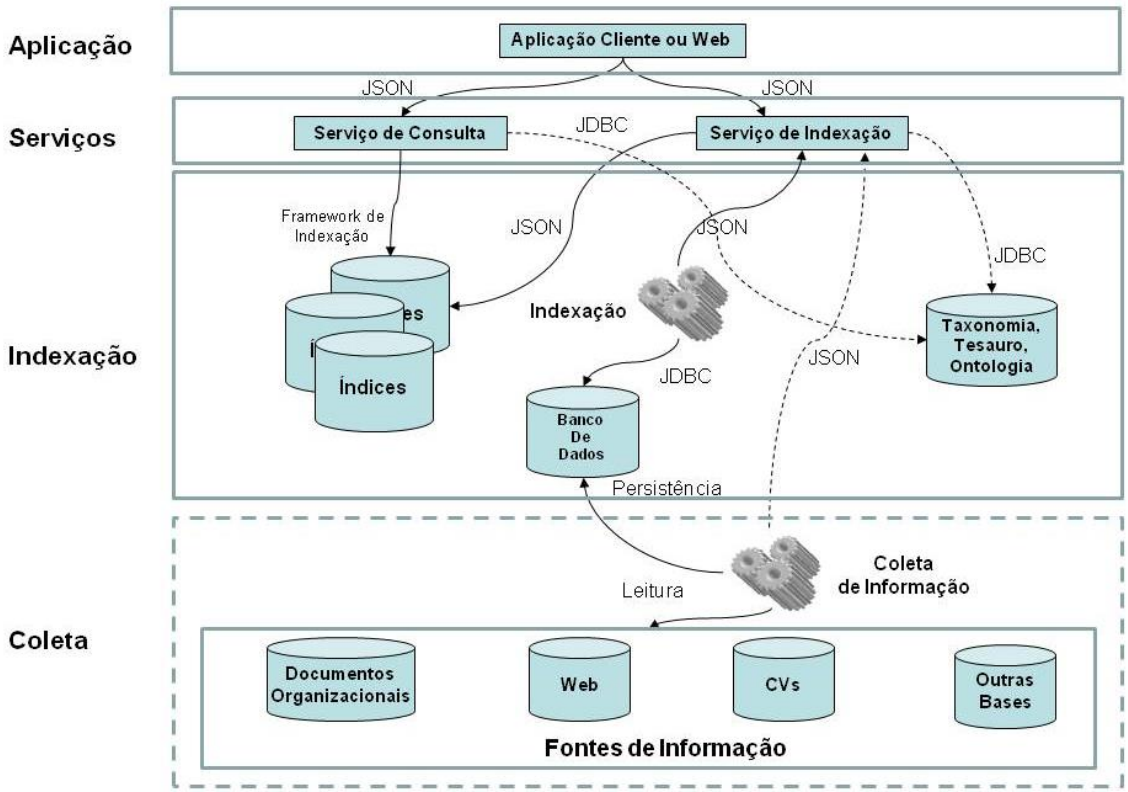


Figura 10 - Visão física do sistema proposto

3.3.1 Coleta de Dados

A coleta de dados pode ser aplicada em diferentes fontes de informação como em documentos organizacionais, currículos vitae, paginas Web e em diversas outras bases. O coletor também chamado de *crawler*, começa, primeiramente, com um conjunto de endereços para serem coletados chamados de sementes. Estas sementes podem ser URLs que indicarão o ponto de partida do coletor. Além das sementes, também é indicado ao coletor o tipo de dado de interesse, neste caso, documentos textuais. Após configurar o coletor com essas informações, o mesmo realiza leitura das fontes de informação automaticamente, e após isso promove a persistência dos documentos que foram encontrados em um banco de dados, para posterior análise, ou pode ainda, enviar diretamente ao serviço de indexação. Vale ressaltar que a etapa de coleta de dados não será implementado nesse trabalho.

3.3.2 Indexação

Logo após a coleta de dados e com esses dados persistidos em um banco de dados, a próxima etapa é a indexação, que deve capturar o conteúdo no banco de dados (*cache*). Isso é realizado através de uma conexão JDBC (*Java Database Connectivity*). JDBC é uma interface de programa de aplicação (API) escrita em Java que possibilita a conexão com bases de dados. O processo abre uma conexão com um banco de dados, e após a conexão executa instruções SQL visando recuperar seu conteúdo, ou seja, os documentos.

Após a recuperação do documento este deve ser convertido para ao formato JSON. O JSON (Java Script Object Notation) constitui-se em uma notação padronizada e leve destinada a troca de mensagens e dados entre aplicações. Ele usa um formato de texto que é completamente independente da língua. Ao contrário de outros padrões, os objetos JSON são analisados como um vetor de string, o que lhe confere rapidez e eficiência..

Para a indexação propriamente dita utilizou-se o *framework* ou API de indexação Lucene. O Lucene é uma biblioteca Java voltada ao desenvolvimento de aplicações de recuperação de informação, ou seja, a indexação e consulta de documentos textuais. De modo geral, pode ser intergrada com qualquer linguaguem ou aplicação através de serviços web. No processo de indexação o Lucene torna-se responsável por criar a ligação entre a aplicação e o índice que será armazenado diretamente no sistema de arquivos do sistema operacional. Permite ainda uma representação genérica de um documento na forma tabular, ou seja, através de um identificador para um campo e o seu conteúdo. Já no processo de consulta ele também é responsável por criar a ligação entre a aplicação e o índice que será consultado, para traduzir

a requisição em um formato adequado, de tal maneira que seja possível a recuperação dos documentos que satisfazem a consulta.

3.3.3 Serviços

Os serviços disponíveis na arquitetura utilizam a tecnologia *Servlet* que tem emergido como uma forma de estender um servidor Web com a funcionalidade de carregar páginas Web dinâmicas. Um *Servlet,* é um programa Java que é executado em um servidor Web. De modo geral, o *Servlet* intercepta uma solicitação HTTP de um navegador, gera o conteúdo dinâmico e fornece uma resposta de retorno via HTTP para o navegador. Alternativamente, o *Servlet* pode ser acessado diretamente de outro componente da aplicação ou enviar a sua saída para outro componente de um sistema mais complexo.

Mais especificamente, um *Servlet* é executado em um servidor de aplicação J2EE, ou seja, um *Servlet* é um tipo de aplicação do componente J2EE que é executado no lado do servidor. Neste sentido, os serviços de indexação e consulta são implementações considerando a tecnologia *Servlet*.

O serviço de consulta também utiliza, caso seja necessário, uma conexão JDBC para acessar os dados no tesauro propondo obter termos relacionados aos utilizados na consulta informada pelo usuário. Por exemplo, se a consulta for “Recuperação de Informação” o serviço de consulta poderia acessar o tesauro e recuperar seus sinônimos visando a expansão da consulta original. Neste sentido, não somente documentos relacionados ao termo principal seriam recuperados mas também documentos em que os sinônimos fossem encontrados. Esta possibilidade de consulta, chamada de expansão vetorial.

3.3.4 Aplicação

A última camada é representada por uma aplicação cliente, que em geral, é constituída por uma aplicação Web implementada por uma tecnologia que possibilite a carga dinâmica de conteúdo em um navegador (*browser*) Web.

A interação da aplicação com os serviços de indexação e consulta é realizada através de requisições via objetos no formato JSON. No caso do serviço de consulta o retorno ocorre através do formato XML e poderá ser apresentado conforme a necessidade da aplicação.

4. DESENVOLVIMENTO E APRESENTAÇÃO DOS RESULTADOS

4.1 INTRODUÇÃO

A apresentação dos resultados demonstrada neste capítulo visa permitir uma visão das fases de indexação e consulta de conteúdo textual e possibilidades de utilização de um sistema de recuperação de informação. Este capítulo está dividido em três partes, sendo:

- Detalhamento do Protótipo: Apresenta de maneira mais detalhada o protótipo desenvolvido através dos serviços de indexação e consulta implementados no trabalho. O projeto como um todo recebeu o nome de *Chronos*.
- Cenário de Aplicação: Apresenta o cenário de aplicação do protótipo declarando de maneira geral a origem e composição do *corpus* (coleção de documentos) utilizada.
- Exemplos de Consultas: Promove uma visão geral das possibilidades de consulta que podem ser realizadas através do protótipo, assim como o resultado em formato XML visando a integração com interfaces gráficas.

4.2 DETALHAMENTO DO PROTÓTIPO

Para um melhor entendimento do protótipo desenvolvido neste trabalho, as seções a seguir detalham os serviços de indexação e consulta de conteúdo textual.

4.2.1 Indexação

Uma característica importante do sistema de recuperação de informação proposto é que ele é genérico, ou seja, ele pode receber qualquer tipo de documento para ser indexado. As Tabela 10 e Tabela 11 apresentam exemplos. A Tabela 10 representa um documento qualquer que no índice terá alguns campos, sendo uma chave de identificação do documento, um título, um autor, uma data e um tipo. Já a Tabela 11, representa o cadastro de um cliente,

que no índice terá também uma chave de identificação do cliente que será o CPF, um nome, uma data de nascimento, um e-mail, um endereço e um tipo. Nota-se que nos dois exemplos existe o campo **tipo**. Este campo tem o objetivo de ser um artifício que permite a distinção de quais documentos serão recuperados conforme a necessidade do usuário.

Campo	**Descrição**	**Conteúdo**
Chave	Identificação do documento	URL-1
Titulo	Título do documento	Chronos1
Autores	Autores do documento	Autor1; Autor2;
Data	Data do documento	12/11/2012
Tipo	Tipo do documento	Geral

Tabela 10: Modelo de um registro de índice representando um documento geral

Campo	**Descrição**	**Conteúdo**
Chave	Identificação do cliente	744.587.574-45
Nome	Nome do cliente	Cliente1
Data_de_nascimento	Data de nascimento do cliente	23/08/1990
E-mail	E-mail do cliente	Cliente1@cliente.com
Endereco	Endereço do cliente	Endereço1
Tipo	Tipo do documento	Cliente

Tabela 11: Modelo de um registro de índice representando um cliente

De modo geral cada documento enviado ao serviço de indexação deve seguir um padrão, que no caso desse trabalho, é representado através de uma requisição JSON. A Figura 11 apresenta um exemplo do objeto JSON proposto. Nele deve conter a operação que será realizada, neste caso é a indexação, e os demais campos necessários que representam um documento, tais como texto, título, tipo e chave. Em todos os campos existem 4 atributos, sendo *analyzed*, *content*, *field* e *stored*. O campo *analyzed* indica que o texto sofrerá a aplicação de um analisador para a separação das palavras do texto permitindo a busca completa (tokenização). Um analisador comum é a separação por espaços, mas isto depende da necessidade. Vários outros analisadores existem disponíveis, por exemplo, analisadores específicos para determinadas línguas, entre elas o português, em que ao processar cada palavra os acentos são retirados e caso seja possível esta a mesma palavra é trazida para o seu radical. O campo *stored* indica se o campo deve ser armazenado ou não no índice. Isto é útil por questões de desempenho, visto que ao recuperar o documento o conteúdo já pode ser retornado evitando a obtenção desse conteúdo em outras bases. Por outro lado, isso implica em um aumento do tamanho do índice. Já o campo *content* indica o conteúdo que deverá ou não ser analisado e que deverá ou não ser armazenado dependendo da configuração dos

atributos *analized* e *stored*. Por fim, *field* indica o nome que o campo irá possuir na estrutura do índice.

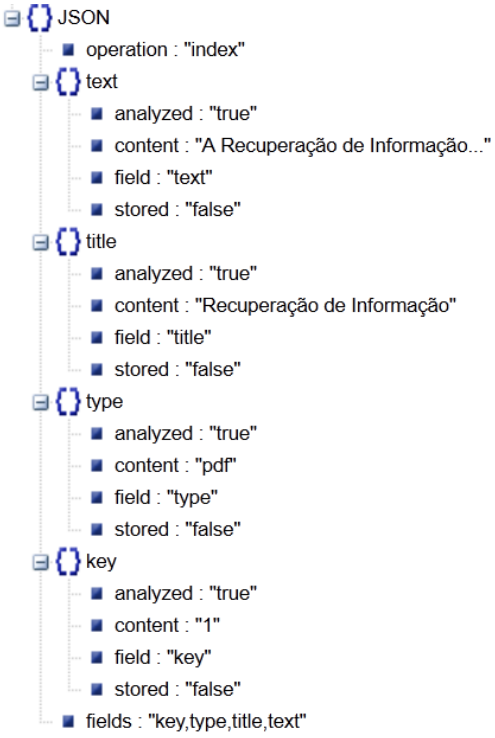


Figura 11 - Exemplo de mensagem JSON utilizada na indexação

Para descrever o comportamento geral o serviço de indexação é utilizado um diagrama de sequência (Figura 12), que serve para mostrar os objetos do sistema e seus comportamentos tempo de execução.

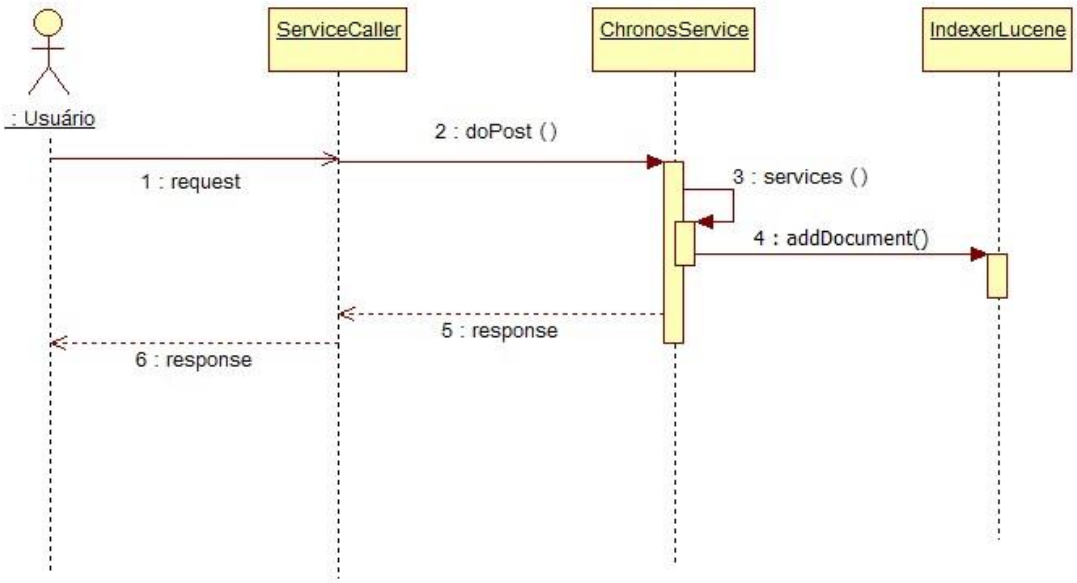


Figura 12 - Diagrama de sequência do serviço de indexação

Logo após a requisição ter sido enviada pelo usuário através da *ServiceCaller* é criado um canal de conexão convertendo a requisição em uma sequência de *bytes* para que ela possa ser serializada e enviada via HTTP à classe *ChronosService*, que recebe a mensagem através do método *doPost*(). Este método possui como comportamento básico, a interceptação de determinada requisição realizada através de uma URL. De posse da mensagem que trará as instruções de qual serviço e dos parâmetros desse serviço é realizado o processamento. Após o processamento a resposta é configurada e devolvida através do canal que foi previamente estabelecido.

O método *doPost*() responsável por interceptar a mensagem de requisição, realiza o processamento do serviço através do método *services*() e configura o retorno na forma de uma resposta.

O método *services*() recebe uma mensagem do tipo *InputStream* converte para *string*, e a partir da *string* é criado o objeto JSON. Com isso o método *services*() analisa qual a operação foi enviada (nesse caso *index*) e invoca o método *addDocument*() para indexar o conteúdo enviado.

O método *addDocument*() recebe o objeto JSON e analisa a análise do seu conteúdo convertendo-o para uma lista, em que cada posição da lista se refere a um campo que será adicionado ao índice na forma de um documento. Um documento pode ser visto como uma representação genérica em formato tabular, ou seja, uma constituição de múltiplos campos onde cada campo possui um nome que o descreve e um valor (conteúdo). Para o índice um documento e constituído de uma instância da classe *Document* em que para cada campo (*field*) existem três propriedades, sendo, *content*, *stored*, *analyzed*.

Ao final, o método *service*() devolve ao usuário se o documento foi indexado com sucesso ou não.

4.2.2 Consulta

O serviço de consulta realiza uma requisição através da classe *ServiceCaller* por meio de objeto JSON criando um canal de comunicação via protocolo HTTP com a classe *ChronosService*. A classe *ChronosService* recebe a requisição através do método *doPost*() que após isso invoca o método *services*() da própria classe *ChronosService*. Esse método realiza a conversão da *stream* recebida via HTTP para um objeto no formato JSON e em seguida chama o método *search*() da classe *SearcherLucene*. O método *search*() abre o

diretório onde está localizado o índice, traduz a requisição em um formato adequado em formato Lucene e realiza a consulta. O resultado da consulta é, então enviada ao usuário respondendo a requisição.

Para descrever de maneira geral o serviço de consulta, é utilizado um diagrama de sequência (Figura 13), objetivado em apresentar os objetos e seus comportamentos em tempo de execução.

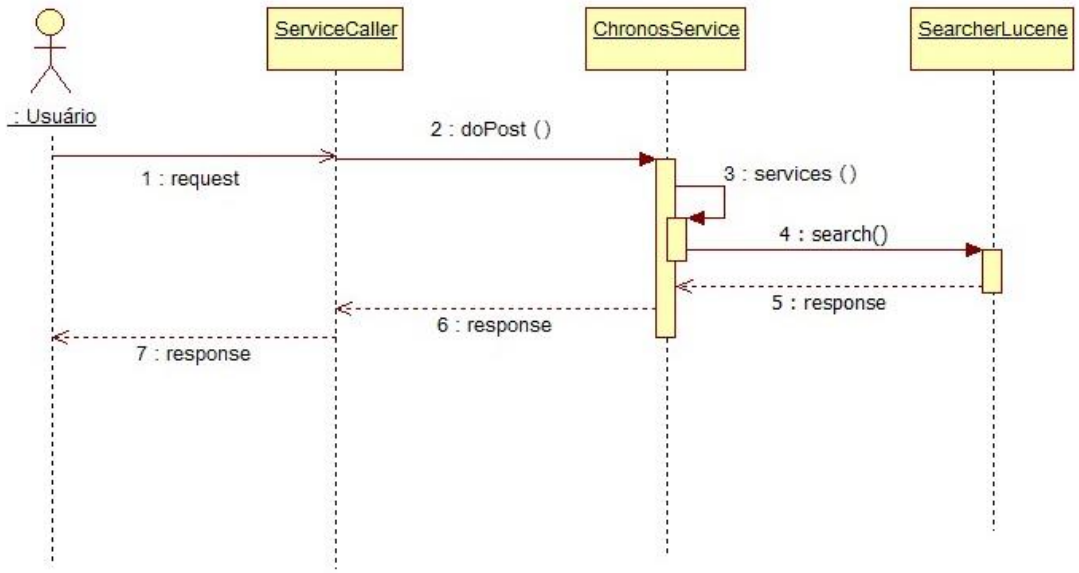


Figura 13 - Diagrama de Sequência do serviço de consulta

A Figura 14 apresenta a mensagem de requisição JSON que é enviada ao servidor de consulta, sendo que o objeto JSON contém a operação que ele deverá realizar, neste caso, a operação de consulta "*search*". Além disso, a mensagem contém mais alguns campos, sendo: a) *query*, que indica o conteúdo a ser pesquisado; b) *sort*, que indica por qual campo o resultado da consulta será ordenado; c) *hits*, que informa quantos documentos devem no máximo ser retornados, por exemplo, 10 documentos; d) *offset*, que indica o deslocamento no número de documento cada vez que a consulta é executada permitindo a realização de paginação na aplicação cliente; e e) *project*, que indica os campos que devem retornar como resultado na consulta, por exemplo, *title*, *url* e *date*.

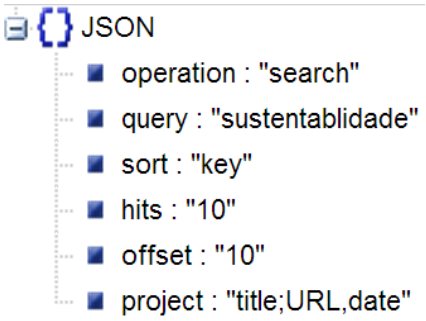


Figura 14 - Exemplo de configuração de uma requisição de consulta em formato JSON

Como mencionado, o campo *project* indica quais campos do índice serão retornados como resultado da consulta. Contudo, determinado campo somente pode ser retornado caso este tenha sido armazenado no índice textual através do atributo *stored*. A Figura 16 apresenta um exemplo do retorno esperado de um consulta em formato XML. Basicamente, o retorno contém informações que se referem ao tempo gasto na consulta, ao próprio conteúdo de busca (*query*) e o total de documentos que satisfazem a consulta. Além disso, é retornada a lista de documentos que satisfazem a consulta com o conteúdo indicado no campo *project* limitados sempre pelo valor do campo *hits*, ambos presentes na requisição JSON. Cada documento retornado (elemento item) possui também uma indicação de sua relevância (propriedade *score*) calculada considerando os termo(s) da consulta e os termos que compõem o documento através da equação do cosseno.

```
<search elapsed_time="" search_content="" total="">
        <item title="" URL="" date="" score="">
        …..
</search>
```

Figura 15 – Exemplo do formato de retorno das consultas em XML

4.3 CENÁRIO DE APLICAÇÃO

Para essa etapa utilizou-se como fonte de informação um conjunto de artigos a partir do Jornal da Ciência[8] que são disponibilizados publicamente em seu site. Ao todo foram coletados, manualmente, 100 artigos sendo estes armazenados em um Banco de Dados Relacional.

[8] http://www.jornaldaciencia.org.br/index2.jsp

A partir disso, desenvolveu-se uma classe que conecta no banco de dados, coleta cada registro e envia ao servidor de indexação. Cada registro possui informações sobre o título, a data de publicação e o texto completo do artigo.

Todos as consultas discutidas na próxima seção são executas sobre os 100 artigos armazenados em um índice invertido e o resultado dessas consultas é apresentado através de um XML, em que a ordem dos documentos é determinada pela relevância (*score*) destes em relação a consulta. O retorno segue o padrão apresentado na Figura 15. Além disso, cada consulta é configurada por meio do padrão JSON, conforme apresentado na Figura 14.

4.4 EXEMPLOS DE CONSULTAS

Neste trabalho são realizados quatro exemplos de consultas, sendo que para a primeira consulta é utilizado o termo “Ciência”, a segunda consulta é realizada com o termo “Tecnologia”, e a terceira consulta com os termos “Ciência” e “Tecnologia” utilizando-se o operador **AND**. Para a quarta, e última consulta, é utilizada uma expressão “Ciência e Tecnologia”.

Nesta primeira consulta o termo utilizado foi “Ciência” em que são retornados 47 documentos que possuem no texto completo a palavra **Ciência**. Vale ressaltar que a consulta foi alterada para “cienc”, pois sofreu o processamento do analisador. No caso do trabalho utilizou-se o analisador português do Brasil que elimina acentos e obtêm a raiz de cada palavra. A Figura 16 apresenta a requisição JSON para a consulta.

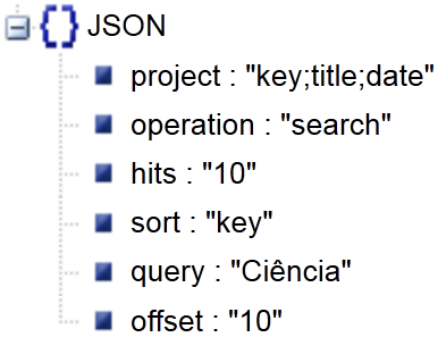


Figura 16 – Requisição JSON para a consulta com o termo “Ciência”

A Figura 17 apresentada o retorno em XML da consulta para a palavra Ciência. Dos 47 documentos encontrados somente os 10 mais relevantes são apresentados, pois considera o parâmetro *hits* da requisição da consulta. Vale resaltar que o parâmetro *hits* é usado em todos os exemplos de consultas. Outra informação importante no retorno da consulta é o tempo

gasto na execução em milissegundos através do atributo *elapsed_time*. Com base no retorno uma aplicação cliente pode realizar a apresentação.

```
<search elapsed_time="55" search_content="ciência" total="47">
    <item key="76093" title="Secretaria de C&T de Goiás busca parceria com governo
        federal e municípios" date="28/01/2011" score="0.2709"/>
    <item key="76151" title="Departamento de ciência e tecnologia(decit) do ministério
        da saúde publica boletim especial sobre comemoração de seus 10 anos"
        date="31/01/2011" score="0.2709"/>
    <item key="76121" title="Luiz Antonio Antoniazzi é o novo presidente da Cientec"
        date="28/01/2011" score="0.2682"/>
    <item key="76162" title="Espaço Ciência inagura exposição sobre evolução dos
        animais" date="31/01/2011" score="0.2655"/>
    <item key="76110" title="O museu de astronomia e ciência afins,"
        date="28/01/2011" score="0.2423"/>
    <item key="76119" title="Nova diretora da FAP do Mato Grosso do Sul é nomeada"
        date="28/01/2011" score="0.2323"/>
    <item key="76088" title="Prorrogadas inscrições para a Cátedra Dra. Ruth Cardoso,
        na Universidade de Columbia" date="28/01/2011" score="0.2299"/>
    <item key="76069" title="Inpa integrará rede brasileira de pesquisa sobre mudanças
        climáticas" date="28/01/2011" score="0.2167"/>
    <item key="76072" title="Pesquisadores do Tocantins reúnem-se com secretário"
        date="28/01/2011" score="0.2167"/>
    <item key="76094" title="Secretário de C&T do Paraná discute parcerias com a
        secretaria de Indústria e Comércio" date="28/01/2011" score="0.2120"/>
</search>
```

Figura 17 – Retorno XML para a consulta com o termo "Ciência"

Já para a segunda consulta, o termo utilizado foi "Tecnologia" em que são retornados 53 documentos no total encontrados, mas somente os 10 mais relevantes são apresentados, pois considera o parâmetro *hits* da requisição da consulta. A Figura 18 representa a requisição JSON para a consulta e a Figura 19 o retorno em XML da consulta. Neste caso a consulta sofreu o processamento do analisador sendo alterada para "tecnolog".

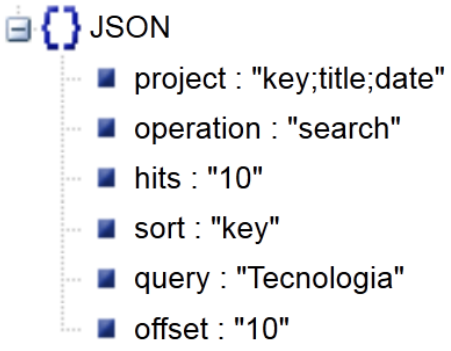


Figura 18 – Requisição JSON para a consulta com o termo "Tecnologia"

<search elapsed_time="57" search_content=" tecnologia" total="53">
<item key="76128" title="Falece Adriano Caranassios, pesquisador do Cetem" date="28/01/2011" score="0.3388"/>
<item key="76093" title="Secretaria de C&T de Goiás busca parceria com governo federal e municípios" date="28/01/2011" score="0.3341"/>
<item key="76122" title="A desindustrialização e o aumento da inovação" date="28/01/2011" score="0.2603"/>
<item key="76151" title="Departamento de ciência e tecnologia(decit) do ministério da saúde publica boletim especial sobre comemoração de seus 10 anos" date="31/01/2011" score="0.2525"/>
<item key="76121" title="Luiz Antonio Antoniazzi é o novo presidente da Cientec" date="28/01/2011" score="0.2500"/>
<item key="76057" title="Publicadas nomeações para o ministério da Ciência e Tecnologia" date="28/01/2011" score="0.2500"/>
<item key="76094" title="Secretário de C&T do Paraná discute parcerias com a secretaria de Indústria e Comércio" date="28/01/2011" score="0.2338"/>
<item key="76072" title="Pesquisadores do Tocantins reúnem-se com secretário" date="28/01/2011" score="0.2259"/>
<item key="76119" title="Nova diretora da FAP do Mato Grosso do Sul é nomeada" date="28/01/2011" score="0.2165"/>
<item key="76073" title="Jorge Audy assume diretoria da Anprotec" date="28/01/2011" score="0.2143"/>
</search>

Figura 19 – Retorno XML para a consulta com o termo "Tecnologia"

Para a terceira consulta, os termos consultados foram "Ciência" e "Tecnologia" conectados pelo operador **AND** em que são encontrados 36 documentos e retornados somente os 10 mais relevantes em função do parâmetro *hits*. A utilização do operador **AND** na consulta indica ao serviço de consulta que todos os documentos que possuam Ciência e Tecnologia, em qualquer parte do documento, devem ser recuperados. A Figura 20 representa a requisição JSON para a consulta e a Figura 21 o retorno em XML da consulta.

Pode-se notar nesta consulta que os *scores* são relativamente maiores. Isto se justifica pelo cálculo do cosseno, uma vez que quanto maior for a quantidade de termos na consulta e a relação direta destes com o documento (produto dos pontos) maior serão a semelhanças entre os vetores (consulta e documento).

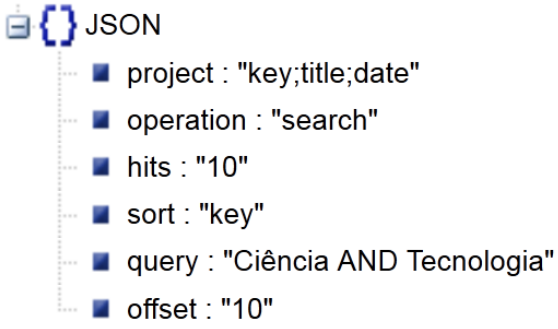


Figura 20 – Requisição JSON para a consulta com os termos "Ciência AND Tecnologia"

<search elapsed_time="58" search_content="ciência AND tecnologia" total="36">
<item key="76093" title="Secretaria de C&T de Goiás busca parceria com governo federal e municípios" date="28/01/2011" score="0.4260"/>
<item key="76151" title="Departamento de ciência e tecnologia(decit) do ministério da saúde publica boletim especial sobre comemoração de seus 10 anos" date="31/01/2011" score="0.3704"/>
<item key="76121" title="Luiz Antonio Antoniazzi é o novo presidente da Cientec" date="28/01/2011" score="0.3666"/>
<item key="76128" title="Falece Adriano Caranassios, pesquisador do Cetem" date="28/01/2011" score="0.3499"/>
<item key="76119" title="Nova diretora da FAP do Mato Grosso do Sul é nomeada" date="28/01/2011" score="0.3175"/>
<item key="76094" title="Secretário de C&T do Paraná discute parcerias com a secretaria de Indústria e Comércio" date="28/01/2011" score="0.3146"/>
<item key="76072" title="Pesquisadores do Tocantins reunem-se com secretário" date="28/01/2011" score="0.3126"/>
<item key="76067" title="Publicadas nomeações para o Ministério da Ciência e Tecnologia" date="28/01/2011" score="0.3092"/>
<item key="76110" title="O Museu de Astronomia e Ciência Afins" date="28/01/2011" score="0.2747"/>
<item key="76118" title="Deputados apontam prioridades para área de ciência e tecnologia" date="28/01/2011" score="0.2696"/>
</search>

Figura 21 – Retorno XML para a consulta com os termos "Ciência AND Tecnologia"

Neste quarto exemplo, os termos consultados foram "Ciência e Tecnologia" em que são retornados apenas 10 documentos no total, número igual ao valor do parâmetro *hits*. A Figura 22 representa a requisição JSON para a consulta e a Figura 23 é o retorno em XML da consulta. Nessa consulta o número de documentos é menor devido ao uso das aspas indicando ao serviço de consulta que somente devem ser recuperado os documentos em que os termos **Ciência** e **Tecnologia** ocorram consecutivamente.

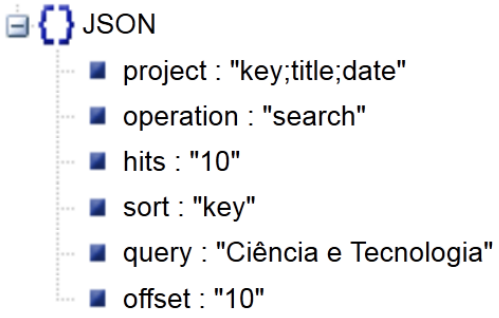


Figura 22 – Requisição JSON para a consulta com os termos "Ciência e Tecnologia"

<**search** elapsed_time=”61” search_content=”" ciência e tecnologia "” total=”10”>
<**item** key=”76151” title=” Departamento de ciência e tecnologia(decit) do ministério da saúde publica boletim especial sobre comemoração de seus 10 anos” date=”31/01/2011” score=”0.4533”/>
<**item** key=”76094” title=”Secretário de C&T do Paraná discute parcerias com a secretaria de Indústria ,e Comércio” date=”28/01/2011” score=”0.3664”/>
<**item** key=”76093” title=”Secretaria de C&T de Goiás busca parceria com governo federal e municípios” date=”28/01/2011” score=”0.2617\”/>
<**item** key=”76117” title=”Glauco Arbix assume presidência da Finep nesta sexta-feira” date=”27/01/2011” score=”0.2094\”/>
<**item** key=”76120” title=”Secti e UFBA buscam aprimorar políticas de C&T” date=”28/01/2011” score=”0.2094\”/>
<**item** key=”76119” title=”Nova diretora da FAP do Mato Grosso do Sul é nomeada” date=”28/01/2011” score=”0.1832\”/>
<**item** key=”76068” title=”Em Manaus, Mercadante ressalta necessidade investir mais em recursos humanos” date=”28/01/2011” score=”0.1832\”/>
<**item** key=”76123” title=”A articulação entre educação e ciência e tecnologia” date=”28/01/2011” score=”0.1570\”/>
<**item** key=”76143” title=”Glaucius Oliva assume o CNPq” date=”31/01/2011” score=”0.1570\”/>
<**item** key=”76081” title=”Sputnik Redux” date=”28/01/2011” score=”0.1047\”/>
</**search**>

Figura 23 – Requisição JSON para a consulta com os termos “Ciência e Tecnologia”

5. CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo geral o desenvolvimento de um sistema computacional que possibilita o armazenamento de informação estruturada e não estruturada de diferentes domínios. Esse objetivo foi viabilizado com o estudo de *frameworks* e soluções para recuperação de informação, e toda a pesquisa bibliográfica que abrange os conceitos que guiaram a implementação de um sistema de recuperação de informação.

Com o uso dos conceitos obtidos por meio da revisão bibliográfica, foi proposto um sistema de recuperação de informação com o armazenamento flexível de conteúdo textual. Sendo assim, o sistema proposto foi dividido em duas partes, sendo a primeira, a visão lógica e a segunda, a visão física.

A visão lógica promove uma explicação geral das funcionalidades do sistema explicando o funcionamento dos processos de coleta de dados, os tipos de indexação, os serviços de consulta e indexação e a aplicação cliente. Por outro lado, a visão física promove a explicação das funcionalidades do sistema com ênfase no detalhamento dos componentes tecnológicos utilizados nos serviços de consulta e indexação com a aplicação.

Com o intuito de validar os conceitos utilizados no trabalho foi desenvolvido um protótipo tendo como componente principal o *framework* Lucene. O protótipo possui basicamente dois serviços, sendo um voltado à indexação de documentos (conteúdo textual) e outro voltado à consulta. A interação com os serviços, indexação e consulta, acontece por meio de mensagens configuradas utilizando-se o padrão JSON. No que tange ao serviço de consulta este exige, além da requisição, um retorno que é configurado através do padrão XML. O retorno de uma consulta devolve todos os documentos que satisfazem determinado interesse de busca informado por um usuário, bem como, algumas informações adicionais que podem ser utilizadas em uma camada de apresentação.

Por meio da implementação do protótipo, foi possível verificar a flexibilidade e do *framework* Lucene na constituição de soluções para a área de recuperação de informação. Isto vale tanto para a indexação, de modo que cada documento (vetor) armazenado no índice não necessita possuir a mesma estrutura, quanto para a consulta, em que múltiplas possibilidades estão disponíveis.

No Capítulo 4 foram apresentadas quatro possibilidades de consultas, mas são possíveis várias outras, entre elas, a busca de conteúdo aproximado e a indicação de pesos em cada um dos termos da consulta. A primeira consulta procura encontrar, a partir dos termos indicados, quais documentos possuem termos iguais ou similares. Isto é interessante, pois a escrita de um documento pode conter pequenos erros e, considerando esta abordagem, o mesmo seria recuperado. Já a indicação de pesos é interessante quando se deseja aumentar a relevância de determinado termo da consulta, uma vez que por padrão todo termo possui peso 1. Para tal, basta informar para cada termo um peso maior do 0 e menor ou igual a 1. Quanto menor for o peso menor será a contribuição do termo no processo de similaridade vetorial com determinado documento.

Com o desenvolvimento deste trabalho surgiram outras possibilidades que não foram consideradas, de maneira que tornariam o trabalho muito extenso. Como trabalhos futuros pode-se vislumbrar a utilização da computação distribuída tanto no processamento da coleta e *parser* dos documentos (existem múltiplos formatos de documentos), quanto no armazenamento do índice textual. (SHUANG, 2011). No primeiro caso, uma solução utilizando algum *framework* de computação distribuída que facilitasse o desenvolvimento seria adequado. Quanto ao armazenamento, torna-se necessário uma solução mais robusta uma vez que para garantir alta disponibilidade, característica esta essencial para grandes índices, é requerida a funcionalidade de replicação do índice. Atualmente, existe um *framework* de distribuição livre chamado ElasticSearch[9] que possibilita o desenvolvimento de sistemas de recuperação de informação com esta característica.

Além das possibilidades estruturais e de alta disponibilidade cada vez mais se tornar necessário recuperar a informação de maneira adequada e precisa. Para tal, o uso de semântica que auxilie na interconexão entre os elementos formais de um texto (entidades e relacionamentos) pode se entendida como fundamental para promover um melhor entendimento de determinado domínio de conhecimento que é mapeado por uma coleção de

[9] http://www.elasticsearch.org.

documentos (*corpus*). Nesse sentido, as Ontologias podem fornecer o arcabouço teórico e tecnológico para a criação de sistemas de recuperação de informação que levem em consideração a semântica de um domínio em particular.

Outras possibilidades podem ser vislumbradas, como por exemplo, o desenvolvimento de uma camada de serviços com diversas funcionalidades visando facilitar a concepção de sistemas de recuperação de informação. Contudo, os exemplos apresentados anteriormente podem ser vistos como os principais, pois tratam das demandas mais atuais para esta classe de aplicação, ou seja, robustez, escalabilidade e maior entendimento de determinado domínio.

REFERÊNCIAS

BARCALA, F.M.; VILARES, J.; ALONSO, M.A.; GRANA, J.; VILARES, M. **Tokenization and proper noun recognition for information retrieval.** Proceedings of the 13th International Workshop on Database and Expert Systems Applications, v.10, n.9, p. 246- 250, 2-6 Sept. 2002.

BEPPLER, Fabiano Duarte. UNIVERSIDADE FEDERAL DE SANTA CATARINA Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento. **Um modelo para recuperação e busca de informação baseado em ontologia e no círculo hermenêutico.** Florianópolis, SC, 123 f, 2008.

BOVO, Alessandro Botelho. UNIVERSIDADE FEDERAL DE SANTA CATARINA Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento. **Um modelo de descoberta de conhecimento inerente à evolução temporal dos relacionamentos entre elementos textuais.** Florianópolis, SC, 155 f, 2011.

CARDOSO, Olinda Nogueira Paes. Recuperação de Informação. INFCOMP - **Journal of Computer Science.** v. 2, p. 33-38, 2000.

DEMARTINI, Gianluca. **Leveraging semantic technologies for enterprise search**. Proceedings Of The ACM First Ph.d. Workshop in Cikm**,** New York, n. 15, p. 25-32, 2007.

FRAKES, William B., BAEZA-YATES, Ricardo. **Information Retrieval: Data Structures and Algorithms**. Prentice Hall. 1992.

FAYAD, Mohamed; SCHIMIDT, Douglas C. **Object-oriented application frameworks**. Comm. ACM. 1997.

GAMMA, Erich. **Padrões de projeto: soluções reutilizáveis de software orientado a objetos**. Porto Alegre: Bookman, 364 p, 2000.

GAN, Lixin et al. **Query Expansion based on Concept Clique for Markov Network Information Retrieval Model**. Proceedings of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), v. 5, p. 29-33, 2008.

GREENGRASS, E. **Information Retrieval**: A Survey. 224 p, 2000.

GUAN, H,Z; ODBAL, O. **Sentence alignment for web page text based on vector space model**. International Conference on Computer Science and Information Processing (CSIP), v.12, n.14, p.167-170, 2012

HAN, J.J.; CHOI, J.H.; PARK, J.J.; YANG, J.D.; LEE, J.K. **An Object-Based Information Retrieval Model : Toward the Structural Construction of Thesauri**. Proceedings of the IEEE International Forum on Research and Technology Advances in Digital Libraries (ADL), p. 117-125, 1998.

HARMAN, D. Et al. **Inverted Files, in Information Retrieval - Data Structures & Algorithms.** Prentice Hall, 1992.

HATCHER, Erik. **Lucene in Action.** 2. ed. Maxmind Geolite: Meap. 486 p, 2009.

HILBERT, Martin; LÓPEZ, Priscila. The World's Technological Capacity to Store, Communicate, and Compute Information. **Science**, n. 2, p. 1-7, 2011.

HIMMA, K. **The concept of information overload**: A preliminary step in understanding the nature of a harmful information-related condition. Ethics and Information Technology, v.9, n. 4, p. 259-272, 2007.

HUA, Cheng et al. Text categorization algorithms using semantic approaches, corpus-based thesaurus and WordNet. **Expert Systems With Applications**, n. 5, p. 765-772, 2011.

JOON, Ho Lee. **Analyzing the effectiveness of extended Boolean models in Information Retrieval**. Technical Report TR95-1501, Cornell University, 1995.

JONES, William P.; FURNAS, George W. Pictures of Relevance: A geometric Analysis of Similarity Measures. **Journal of the American Society for Information Science,** Maryland, v. 36, n. 6, p. 420-442. 1987.

KANA, Takashi; JIAN, Li; KUNIFUJI, Susumu. **Related Document-based Information Filtering Applied to the Association Model Information Retrieval System**. Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, v. 1, p. 225-228, 2000.

KOBAYASHI, M.; TAKEDA, K. Information retrieval on the web. **ACM Computing Surveys**, v. 32, n. 2, p. 144-173. 2000.

KORFHAGE, Robert R. **Information Storage and Retrieval**. New York: John Wiley & Sons, Inc, 1997.

KOWALSKI, Gerald J.. **Information storage and retrieval systems theory and implementation.** 2. ed. Massachussetts: Kluwer Academic Publishers, 2002.

LARMAN, Craig. **Utilizando UML e padrões:** uma introdução à análise e ao projeto orientado a objetos e ao desenvolvimento iterativo. Porto Alegre: Bookman, 695 p, 2007.

LIAW, Shu-sheng; HUANG, Hsiu-mei. Information retrieval from the World Wide Web: a user-focused approach based on individual experience with search engines. **Computers in Human Behavior**, v. 22, n. 3, p. 501-517, 2004.

LYMAN, P. **How Much Information?** USA: University of California at Berkeley, 2003.

MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. **An Introduction to Information Retrieval.** Cambridge: Cambridge University Press, 569p, 2009.

MCCREADIE, Richard; MACDONALD, Craig; OUNIS, Iadh. MapReduce indexing strategies: Studying scalability and efficiency. **Information Processing and Management**, Glasgow, n. 25, p. 1-16, 2010.

MILLER, George et al. Introduction to WordNet: An On-line Lexical Database. **International Journal of Lexicography**, v. 3, n. 4, p. 235-244, 2000.

NASCIMENTO, M. A.; DA CUNHA, A. C. R. **An experiment stemming non-traditional text**. Proceedings of the String Processing and Information Retrieval: A South American Symposium, p. 75-80, 1998.

POHL, Stefan; ZOBEL, Justin; MOFFAT, Alistair. **Extended Boolean Retrieval for Systematic Biomedical Reviews**. Proceedings of the 33rd Australasian Computer Science Conference, Brisbane. 2010.

PORTER, M.F. **An algorithm for suffix stripping.** Cambridge: Program, 130-137 p, 1980.

REN, Fuji; BRACEWELL, David B. Advanced Information Retrieval. **Electronic Notes in Theoretical Computer Science**, p. 303-317, 2009.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência artificial**. São Paulo: Elsevier, 2004.

SALTON, Gerard. **Automatic information organization and retrieval**. New York: McGraw-Hill, 1968

SALTON, Gerard; EDWARD, A F; WU, H. Extended Boolean Information Retrieval. **Communication of the ACM**, v. 26, n. 11, 1983.

SHUANG, Linping; ZHU, Hongjun. **Analysis of Distributed Information Retrieval.** Proceedings of the International Conference on Multimedia Technology (ICMT)**,** p. 5297-5300, 2011.

SMITH, J.R.; NAPHADE, M.; NATSEV, A. **Multimedia semantic indexing using model vectors**. Proceedings of the International Conference on Multimedia and Expo (ICME), v. 2, p. 445-448, 2003.

THROOP, David R. **Enterprise Search Tasks in IVHM Practice**. Proceedings of the IEEE Aerospace Conference**,** 2006.

WAN, Jian; PAN, Shengyi. **Performance Evaluation of Compressed Inverted Index in Lucene**. Proceedings of the International Conference on Research Challenges in Computer Science (ICRCCS), p. 178-181, 2009.

WU, Chen. WSDL term tokenization methods for IR-style Web services discovery. **Science of Computer Programming**, n. 77, v. 3, p. 355-374, 2011.

YUBO, Jia.; XING, Dong.; YI, Wang.; HONGDAN, Fan. **A Document-Based Information Retrieval Model Vector Space**. Proceedings of the Second International Conference on Networking and Distributed Computing (ICNDC), v. 39, n. 13, p. 65-68, 2011.

ZHOU, Yao; CAO, Ze-wen. **Research on the Construction and Filter Method of Stop-word List in Text Preprocessing.** Proceedings of the International Conference on Intelligent Computation Technology and Automation (ICICTA), v.1, n.10, p. 217-221, 2011.