Ferramenta para recuperação de informação baseado em arquivos de índices

Jony Antonio Lemes[1], Luis Fernando de Almeida[1,2]

[1]Faculdasde de Tecnologia Valdomiro May – Centro Paula Souza
12.730-010 – Cruzeiro – SP – Brasil

[2]Departamento de Informática – Universidade de Taubaté (UNITAU)
12.010-000 – Taubaté – SP – Brasil

{jony.lemes}@gmail.com, luis.almeida@unitau.br

Abstract. *In legal scope the text communication is the main way of transmitting knowledge..With advances in technology the textual production in this area goes beyond the human ability to filter the most relevant documents in a short period of time. In this paper is applied the information recovery paradigm in order to provide an efficient query tool to the users of the newspaper "Judiciário do Tribunal Regional do Trabalho" for the information extraction of its section orders.*

Resumo. *No âmbito jurídico, a comunicação textual é a principal forma de transmitir conhecimento. Com o avanço tecnológico, a produção textual nesta área ultrapassa a capacidade humana de filtrar os documentos mais relevantes em curto período de tempo. Neste trabalho é aplicado o paradigma de recuperação de informação, a fim de disponibilizar uma ferramenta eficiente de consulta aos usuários do caderno "Judiciário do Tribunal Regional do Trabalho" para a extração de informação de sua seção de despachos.*

1. Introdução

A comunicação textual é a principal forma de transmitir conhecimento. Sendo fundamental para a área jurídica, por servir como principal forma de divulgar os atos jurídicos através de publicações oficiais.

Apenas no Estado de São Paulo são diariamente publicados por meio eletrônico nove cadernos referentes aos atos oficiais no âmbito do governo estadual. Entre esses cadernos, o Judiciário do Tribunal Regional do Trabalho da 15ª Região, que será tratado nesse texto como JTRT15, que publica os atos pertinentes às varas de trabalho abrangendo 599 Municípios Paulistas, perfazendo 95% do território do Estado. Dentre os usuários do JTRT15, encontram-se advogados, indivíduos e empresas que buscam informações sobre os despachos proferidos pelas varas regionais de trabalho, os advogados encarregados e seus interessados.

Toda publicação do JTRT15 é disponibilizada gratuitamente no site Diário Eletrônico da Justiça do Trabalho, no formato eletrônico denominado *Portable Document Format* (PDF), um padrão de arquivo que mantém a formatação original da publicação, permitindo opções de interatividade com o usuário, que pode pesquisar por palavras contidas no documento.

Mesmo com esse avanço tecnológico na forma de disponibilizar o JTRT15, não existem ferramentas que auxiliem a busca de atos jurídicos de acordo com a necessidade do usuário. Ainda que o formato PDF permita a pesquisa de palavras contidas no documento, não há um valor de relevância para o usuário nesse mecanismo.

Outro fator que dificulta a pesquisa acontece quando se tem a necessidade de armazenamento histórico desse tipo de publicação, pois, para todos os dias úteis da semana, são publicadas, no mínimo, mais de 200 páginas de atos jurídicos, e nem todos os atos interessam aos usuários, tornando ainda mais penosa a tarefa de encontrar, organizar e armazenar essas informações manualmente. Para auxiliar essa tarefa de manter e recuperar informações provenientes de textos surge a recuperação de informação, que vem evoluindo com o passar do tempo de acordo com as necessidades de seus usuários.

Diante do exposto, este trabalho propõe uma ferramenta para auxiliar a tarefa de encontrar, organizar e armazenar informações da seção do JTRT15 denominada Despachos – Intimações/Notificações. Para tanto, propõe um modelo de identificação de padrões no documento que determina as necessidades de informação do usuário e o uso do paradigma da Recuperação de Informação. Para validação do modelo, foi desenvolvido um protótipo de *software* com conceitos que serão abordados nesse trabalho.

Dessa forma, esse artigo trará uma breve revisão sobre os conceitos relacionados à recuperação de informação, uma abordagem sobre as ferramentas que serão utilizadas para o desenvolvimento do protótipo, seguida da descrição da ferramenta desenvolvida para recuperar informações e da apresentação dos testes realizados com a mesma e de considerações finais acerca do que foi tratado.

2. Recuperação de informação

2.1. Conceitos

O significado do termo recuperação de informação pode ser bem vasto, não se limitando apenas a relatórios ou à procura de *website*. Manning et al. (2008) define o termo como a ação de se encontrar material (normalmente documentos) de natureza desestruturada, geralmente textos, que satisfaz uma necessidade de informação dentro de grandes coleções, geralmente armazenadas em computadores.

Esta definição elucida os componentes básicos para o uso da recuperação de informação, a necessidade do usuário e a coleção de documentos disponíveis para pesquisa. Para atingir essa meta, o sistema de Recuperação de Informação deve adotar um modelo de acordo com a estratégia adotada para a função de recuperação. São comuns entre os modelos dois momentos distintos durante a execução de um sistema de recuperação de informação: a *indexação* e a *busca* (Baeza-Yates,1999).

A *indexação* consiste no processo que visa evitar a leitura seqüencial em uma coleção de documentos quando se procura por algum termo ou palavra, convertendo a fonte de texto em um formato que permite uma busca rápida (Gospodnetic et al., 2009). Para tanto, existem várias técnicas de construção de arquivos de *indexação,* dentre as quais podemos citar como exemplo as árvores de sufixos, os arquivos de assinatura e os arquivos invertidos (Baeza-Yates,1999).

Assim como existem várias técnicas de *indexação*, para *busca* existem modelos de recuperação conforme a técnica de *indexação* utilizada. Para a técnica de índices invertidos, podem ser citados o modelo *booleano* e o modelo de *espaço vetorial* (Gospodnetic et al., 2009).

2.2. Processamento de linguagem natural

O *Processamento de Linguagem Natural* (PLN) é o conjunto de métodos formais para analisar textos e fornecer aos computadores a capacidade de reconhecer o contexto, fazer análise sintática, semântica, lexical e morfológica, criar resumos, extrair informação e compor texto (Jackson and Moulinier, 2002).

Alguns métodos da PLN analisam os documentos e as buscas para assim aprimorar o desempenho na criação de índices e no processo de busca, o que pode servir como uma opção de recuperação de informação. A análise compreende as seguintes técnicas lingüísticas voltadas ao processamento de linguagem natural: normalização de variações lingüísticas, remoção de *stop words* e Tesauro.

3. Ferramentas utilizadas

3.1. XML

O XML é uma linguagem de marcação de texto derivada do SGML (ISO 8879). O documento XML é estruturado em forma de uma árvore rotulada, onde cada nó é identificado por dois rótulos, um rótulo de abertura e um rótulo de fechamento, e entre eles o valor do atributo. O XML permite que se possa definir o nome do rótulo desde que este fique entre os sinais “<” e “>” e que o rótulo de fechamento contenha uma barra “/” antes de seu nome.

Documentos XML devem conter um elemento chamado raiz, que é o pai de todos os outros elementos. Os documentos XML também devem conter um cabeçalho, que tem a função de qualificar o documento como um arquivo XML, e, opcionalmente, definir a codificação do texto.

3.2. Digester

Devido à dificuldade de carga do arquivo de configuração do *framework Struts*, foi desenvolvido pelo *Projeto Jakarta* a biblioteca de código aberto *Digester*. A simplicidade e a facilidade de uso da solução fizeram com que esta ferramenta fosse amplamente adotada por diversos outros projetos e sistemas, de tal forma que foi destacada do *Struts* e passou a ser distribuída como um componente separado.

Digester utiliza a biblioteca SAX como base e, sobre ela, um conjunto de classes que facilitam a sua utilização. Estas classes formam um mecanismo genérico para tratamento dos eventos gerados pelo SAX permitindo que, para cada evento gerado, seja mapeada uma ação de execução de um método ou de criação de um objeto Java.

O modo como os eventos devem ser tratados pela aplicação é fornecido para o *Digester* em forma de um conjunto de regras para processamento dos eventos. Esse conjunto é um arquivo XML onde se tem descrição do rótulo XML associado ao evento e especificação do código na aplicação a ser executada.

3.3. Biblioteca Lucene

Lucene é uma biblioteca de recuperação de informação, escrita originalmente por Doug Cutting (Gospodnetic et al., 2009), que permite a indexação e busca em aplicações JAVA. Em setembro de 2001, a biblioteca juntou-se a família de soluções JAVA de código aberto da *Apache Software Foundation's Jakarta* (Gospodnetic et al., 2009), atraindo assim mais desenvolvedores que a tornaram mais robusta e, conseqüentemente, a biblioteca gratuita mais popular de recuperação de informação (Gospodnetic et al., 2009).

Os conceitos de recuperação de informação encontrados na biblioteca *Lucene* são principalmente processos de adição de índices a documentos em um formato e depois a utilização de um algoritmo de busca para encontrar as palavras chave dentro do texto. Para cada processo, indexação e busca, existem componentes na biblioteca *Lucene* cruciais para compreensão da ferramenta.

4. Protótipo proposto

O processo de geração do arquivo de índices pode ser descrito em etapas distintas. Inicialmente, é efetuada a conversão de um documento digital no formato PDF, para o formato texto. Na seqüência, a extração de informação desse documento texto para o formato XML, rotulando cada estrutura importante para compor um ato jurídico completo. Por final a indexação do documento XML com a biblioteca *Lucene* para que se possa realizar busca dos processos por meio de uma interface gráfica. A Figura 1 ilustra todo o processo.

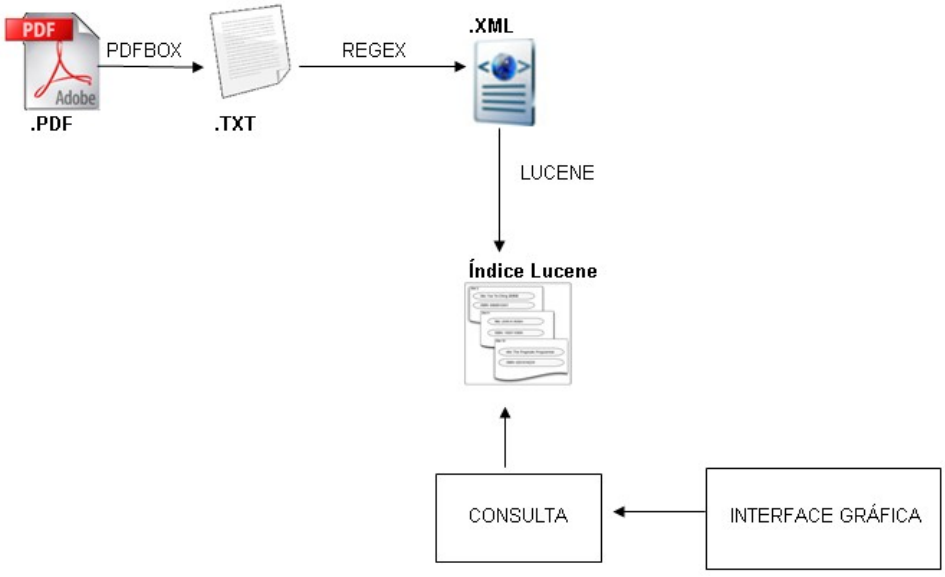

Figura 1. Descrição do processo de funcionamento do protótipo

4.1. Extração da informação

O processo de converter um texto não estruturado para um formato mais significativo computacionalmente é chamado de Extração de Informação (EI), que Marie-Francine define como a identificação e, consequentemente ou concomitantemente, a

classificação e estruturação em classes semânticas, de informações específicas encontradas em fontes de dados não estruturados, como textos em linguagem natural, tornando a informação mais adequada para a tarefa de processamento de informações. (MOENS, p. 04, 2006)

O primeiro passo para a extração de informação neste trabalho é passar o conteúdo do arquivo PDF para um formato em que se possa manipular o texto de forma mais flexível, sem precisar consumir memória de máquina. A biblioteca *PDFBOX* permite que essa mudança de formatos seja feita sem maiores esforços, necessita apenas da indicação de onde se encontra a pasta que contém os arquivos PDF, e usar uma única função de extração de texto. Entretanto, durante esse processo é importante pontuar dois tópicos relacionados ao tratamento que o texto recebe:

- Todas as ocorrências do *e* comercial (&), devem ser substituídas pela letra *e*, pois o uso deste caractere causa falhas no processamento do XML.
- No final de cada linha em que ocorrer a separação da palavra, o ultimo hífen (caso exista mais do que um hífen) e o marcador que representa a quebra de linha, devem ser apagados para remontar a palavra, evitando assim uma futura indexação incorreta da palavra.

No documento de texto gerado no passo anterior, a extração de informação tem como objetivo delimitar o documento JTRT 15 para apenas a seção Despachos - Notificações/Intimações. Para tanto, os padrões textuais identificados nesta seção do caderno serão convertidos em expressões regulares, para que, ao encontrar esses padrões, o sistema inicie o processo de rotulação e conversão para arquivo XML.

Quando encontrado o termo “DESPACHOS INTIMAÇÕES/NOTIFICAÇÕES“ iniciando uma linha, sendo que essa linha não seja parte do sumário do caderno, significa que se iniciou a seção de despachos dos juízes, e dá-se inicio ao processo de extração de informação. O aplicativo inicia o cabeçalho do arquivo XML com o padrão de codificação UTF-8, para caracteres de origem latina e abre o rótulo “<DESPACHOS>”.

Quando uma linha iniciar opcionalmente com até dois números, um dos caracteres “º” ou “ª”, espaço, e obrigatoriamente a palavra “vara” seguido de espaço, dá-se início a uma vara do trabalho. O aplicativo extrai o texto e o insere no XML entre os rótulos “<VARA>” e “<\VARA>”.

Para identificar o advogado encarregado dos processos, o padrão encontrado foi o de que, em qualquer parte do texto onde se encontrar a sigla OAB, o caractere de dois pontos (:), espaço e mais seis dígitos, o aplicativo extrairá a linha e inserirá o conteúdo entre os rótulos “<ADVOGADO>” e “</ADVOGADO>”;

Quanto aos processos, é verificado se a linha inicia com um conjunto de dois a cinco números, e em seguida obrigatoriamente um conjunto com quatro números, um conjunto com três números, dois conjuntos com dois números e finalmente um número, todos os conjuntos e o último número separados por traços, como no exemplo: 1026-2007-068-15-00-2. O aplicativo inicia uma rotina que divide a o número do processo do texto pertinente ao conteúdo do processo, extraindo em seguida o processo e inserindo-o no arquivo XML entre os rótulos “<PROCESSO><NUMERO>” e "</NUMERO>".

Na próxima etapa, é verificado se as linhas posteriores fazem parte do início de uma vara, ou se é algum advogado, ou início de processo, caso não satisfaça nenhuma

das condições, significa que o processo ainda não acabou, e as linhas são concatenadas para formarem o conteúdo do processo. Caso alguma das condições seja verdadeira, todo o texto concatenado é inserido no XML entre os rótulos "<CONTEUDO>" e "</CONTEUDO></PROCESSO>".

Quando o aplicativo encontra uma linha nula, ou a próxima seção do JTRT 15, significa que a seção de despachos acabou e finalmente fecha o rótulo "</DESPACHOS>" iniciado no começo do processo de extração de informação.

3.2. Indexação e busca

Com a extração de informação concluída, o próximo passo é a indexação do arquivo XML. Neste momento que a Biblioteca *Digester* atua em paralelo com a biblioteca *Lucene*.

A regra *Digester*, criada para este trabalho e ilustrada pela Figura 2, define a hierarquia do documento XML, criado no processo de extração de informação. E para cada rótulo identificado pela regra *Digester*, um evento é disparado para a aplicação Java com o valor do rótulo, e a aplicação cria um documento *Lucene*, que passa a preencher os campos com os valores pertinentes a cada rótulo identificado.

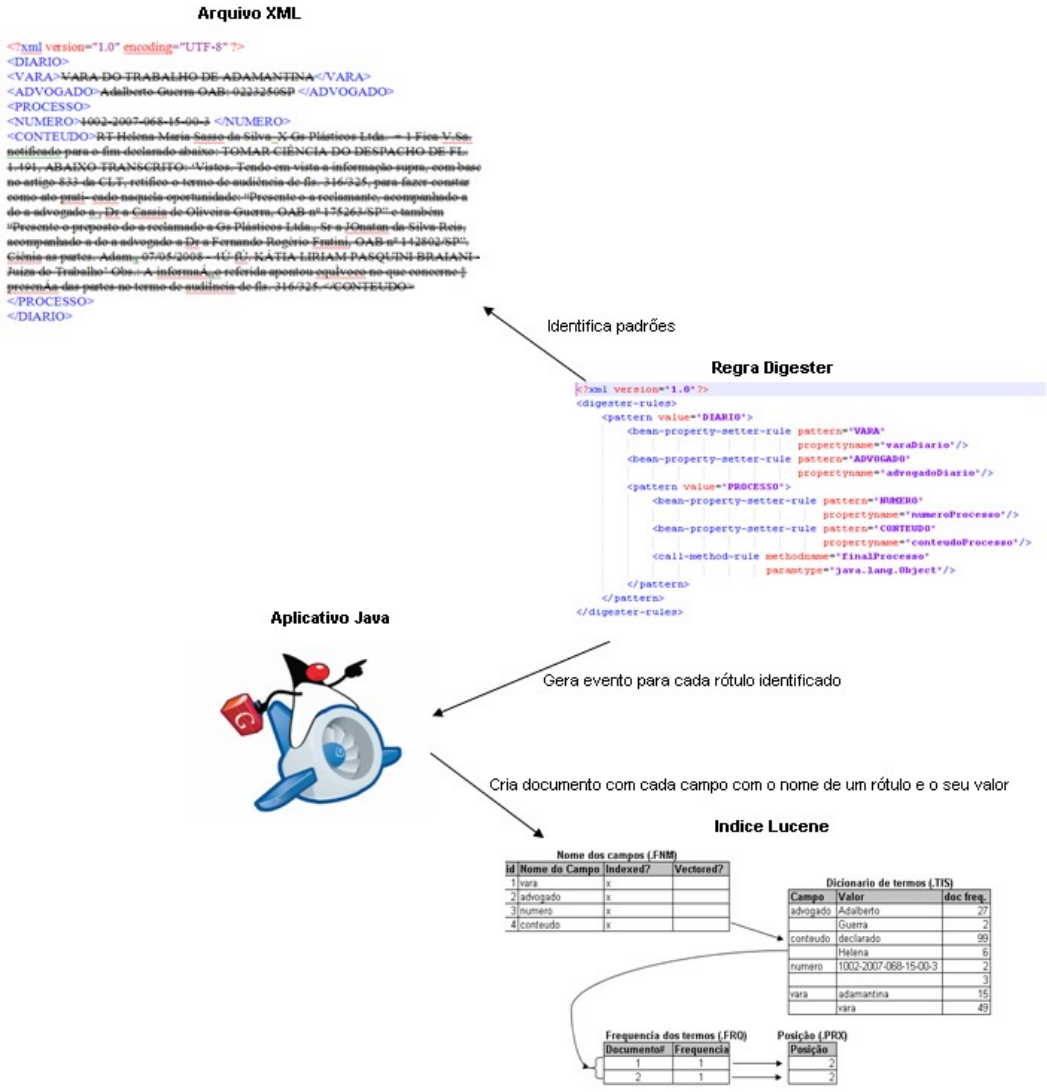

Figura 2. Interações Digester, aplicativo e Lucene

Para a construção do documento *Lucene*, foi escolhido como *Analyzer*, a classe *SimpleAnalyzer*, que não considera *stopwords*, mas que, por outro lado, não considera números como parte do índice. Também foi decidido que todos os termos analisados iriam compor o índice do documento, para tanto o parâmetro MaxFieldLength.UNLIMITED.

Quanto ao conteúdo dos campos, todo ele deve ser armazenado no documento *Lucene*, e apenas não deve ser analisado o campo número, referente ao número do processo, pois o *analyzer* escolhido não considera números para compor o índice. É interessante que esse seja indexado na íntegra.

O processo de busca da aplicação utiliza tanto do modelo *booleano* quanto do modelo de *espaço vetorial*, mas não faz uso de nenhuma forma de *boost* de termos para compor o *scoring*.

3.3. Interface gráfica

O objetivo de qualquer interface gráfica é auxiliar o usuário na utilização de um sistema computacional. Para tanto a interface gráfica, preferivelmente, deve ser simples e intuitiva.

No escopo deste trabalho, a interface gráfica além de ser simples, também usa práticas que auxiliam o usuário no ato da busca de documentos, como o realce de termos pesquisados e lista de sugestão de termos. Para ambos a biblioteca *Lucene* disponibiliza classes prontas, que facilitam a ação.

Para exibir os documentos com o realce nos termos, a busca utilizada é reescrita, e realizada novamente sobre o resultado dos documentos recuperados, mas desta vez novos componentes são usados para fragmentar o texto e adicionar rótulos HTML para formatação.

O objeto *Fragmenter* quebra o texto escolhido em pequenos pedaços, que são passados para um objeto *Scorer* que qualifica os fragmentos que devem ser realçados. A formatação do realce é definida no objeto *Formatter*, que aceita como formação rótulos em HTML, e no objeto *Highlighter* os rótulos são colocados entre os fragmentos qualificados. Para exibir o resultado gerado pelo objeto *Highlighter* foi utilizado o componente da biblioteca *Swing* do *Java* chamada JTEXTPANE, que permite a exibição de texto com formatação HTML definindo seu parâmetro *contentType* para o valor "text/html". A Figura 3 ilustra o resultado da técnica de realce neste trabalho.

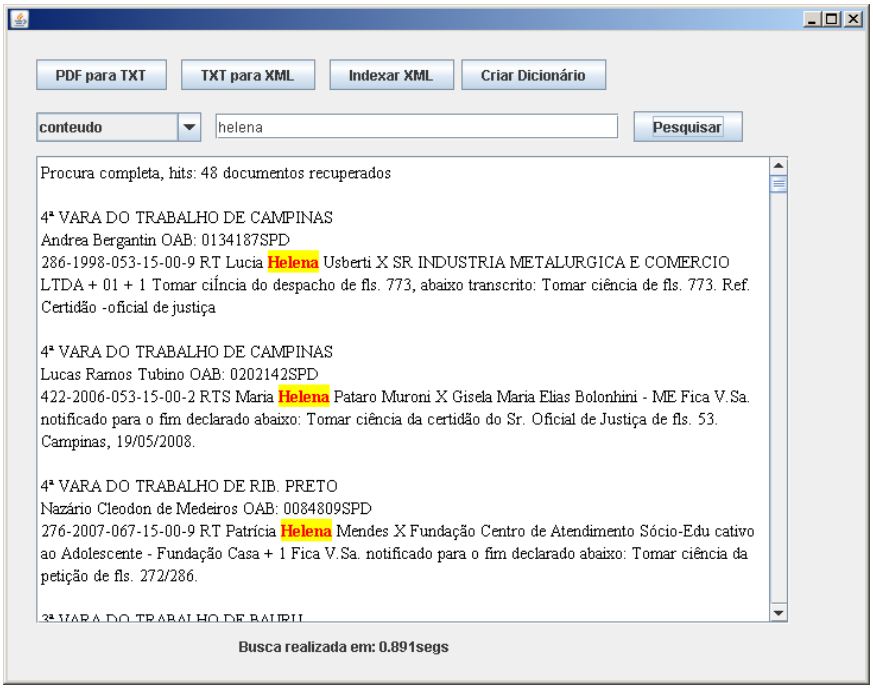

Figura 3. Interface gráfica com uso de realce no termo de busca

Quando, ao se realizar uma busca, não houver o retorno de nenhum resultado, existe a possibilidade de que o usuário tenha digitado o termo de busca errado, ou mesmo que este realmente não conste nos documentos. Para auxiliar o usuário, o sistema de RI pode retornar uma lista com sugestões de termos de busca.

Para tanto, é necessário ter uma fonte de palavras válidas para os documentos indexados até o momento. Tal necessidade se resolve criando um índice em separado que servirá de dicionário apenas para uso do verificador de termos, usando como base todos os termos únicos encontrados em um campo – como no caso deste trabalho o campo conteúdo – durante a indexação dos documentos.

A classe *SpellChecker* da biblioteca *Lucene* é responsável pela criação do índice que serve como dicionário de sugestões, que por padrão já tem seus campos definidos para a utilização de métricas de similaridade. A métrica utilizada por padrão da biblioteca *Lucene* é a distância de *Levenshtein*, que calcula um vetor com o termo de busca e os termos do índice dicionário, considerando o menor custo de modificação do termo - substituição, inserção e eliminação de letras ou conjuntos de letras – para posicionar as sugestões mais relevantes (Stephen, 1994). A Figura 4 ilustra o uso de lista de sugestões neste trabalho.

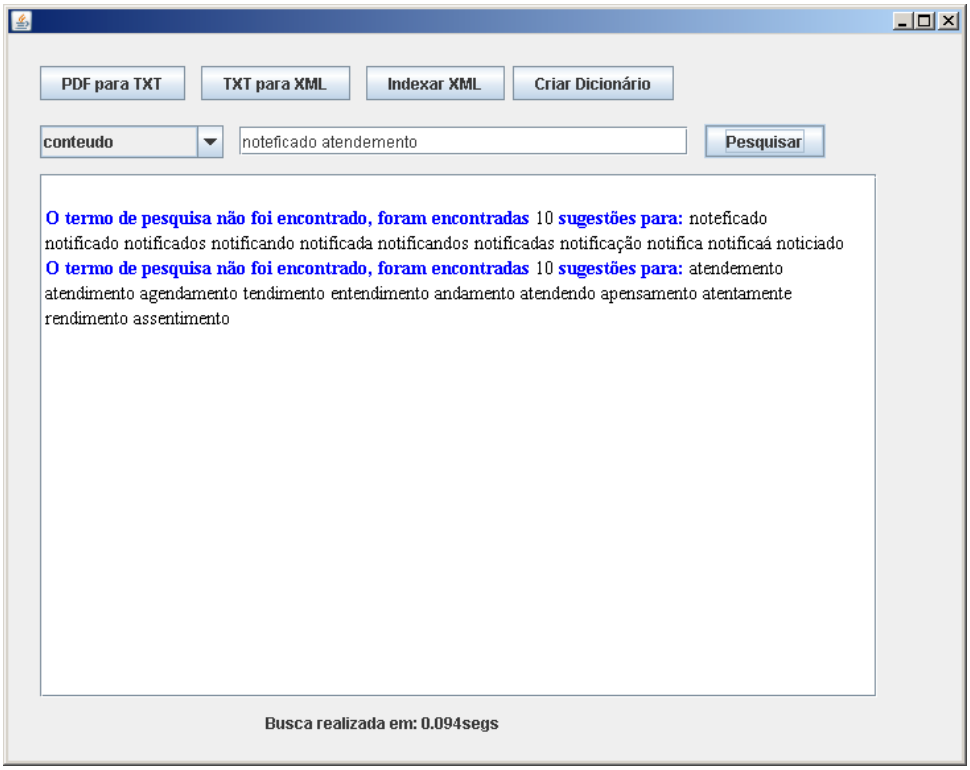

Figura 4. Sugestões de termos de busca

A classe *SpellChecker*, por padrão, permite apenas a verificação de um termo por vez. Para permitir a verificação de mais de um termo, foi necessário neste trabalho criar um método que dividisse os termos de busca, passando um a um para verificação do *SpellChecker*, para depois retornar a lista de sugestões separadamente, sem considerar um valor semântico para o conjunto de termos.

4. Testes

Os testes realizados objetivaram a verificação dos limites de busca do motor de busca do *Lucene* e a usabilidade da interface gráfica.

Em uma primeira simulação foram utilizados dois termos como critérios de busca no campo conteúdo, o motor de busca considerou a procura de documentos que contivessem a ocorrência dos termos em separado, como se estivesse utilizando o operador lógico OR, esta operação é demasiadamente custosa, podendo demorar minutos caso a incidência de um dos termos seja muito alta (o que pode significar uma reavaliação da criação do índice com uso de uma lista de *stopwords*). Na tentativa de fazer com que retornem documentos com os dois termos, foi utilizado o operador AND, o que pode resultar em nenhum retorno, conforme ilustrado pela Figura 5, pois ambos os termos devem pertencer ao documento.

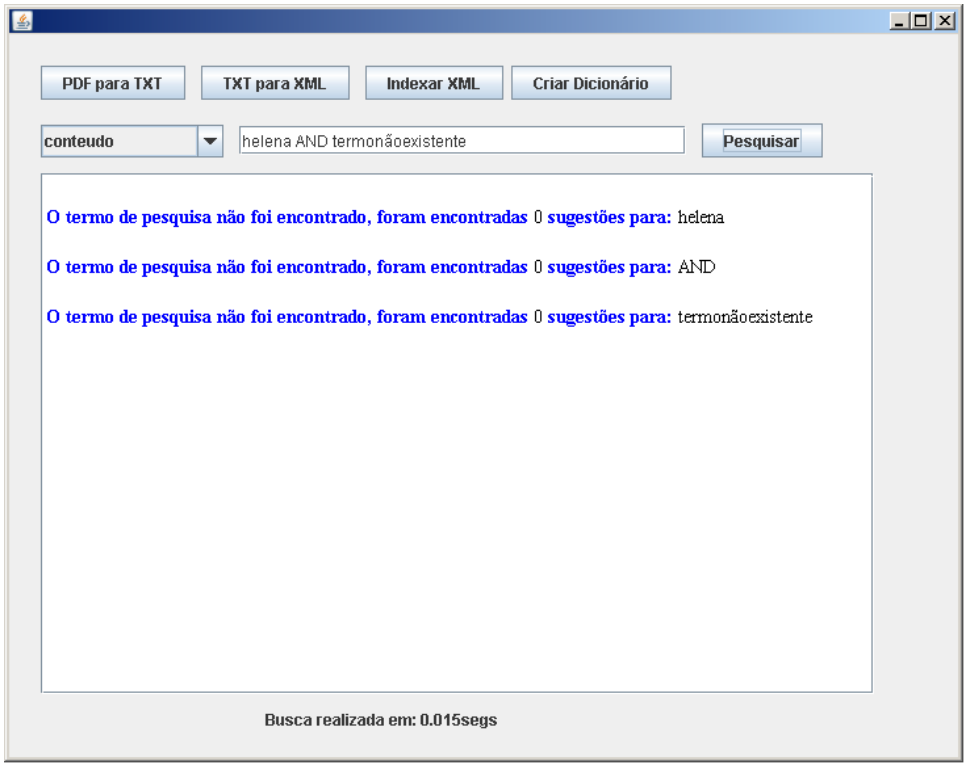

Figura 5. Busca com mais de um termo com operador AND

Um segundo teste foi a utilização dos caracteres coringas *asterisco* (*) e a *interrogação* (?), usados em mais de um termo de busca. Tanto a busca como o realce dos termos funcionaram com método desenvolvido neste trabalho como pode ser visto na Figura 6.

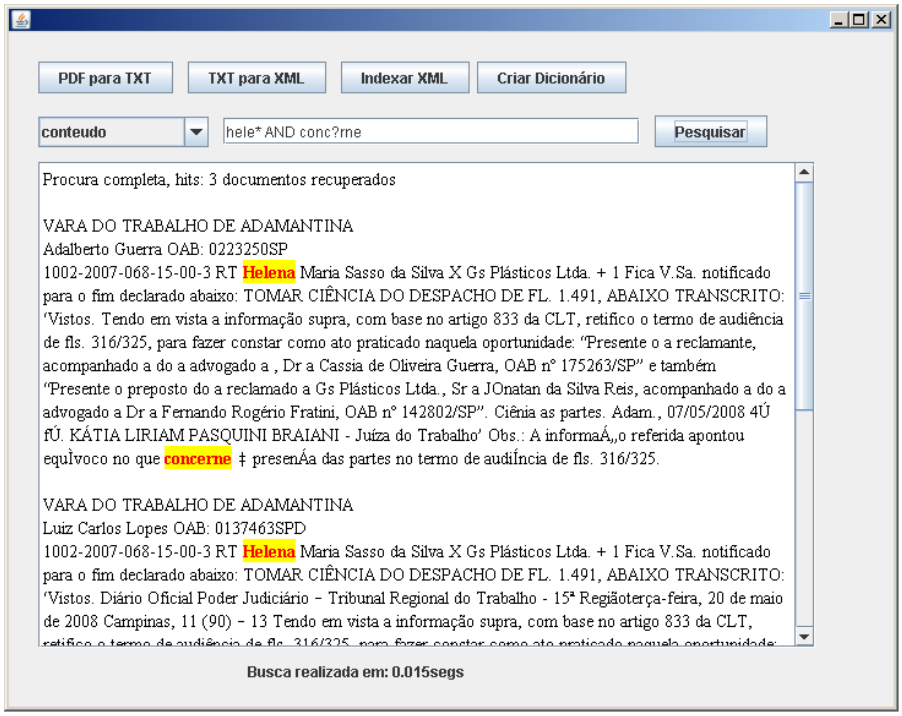

Figura 6. Busca com mais de um termo com operador AND

Em simulação, o mecanismo de busca também suporta busca textual utilizando o caractere *til* (~) que traz resultados baseados no algoritmo *Levenshtein Distance* para palavras que são próximas e não que não se tem certeza da grafia, como podemos ver na Figura 7.

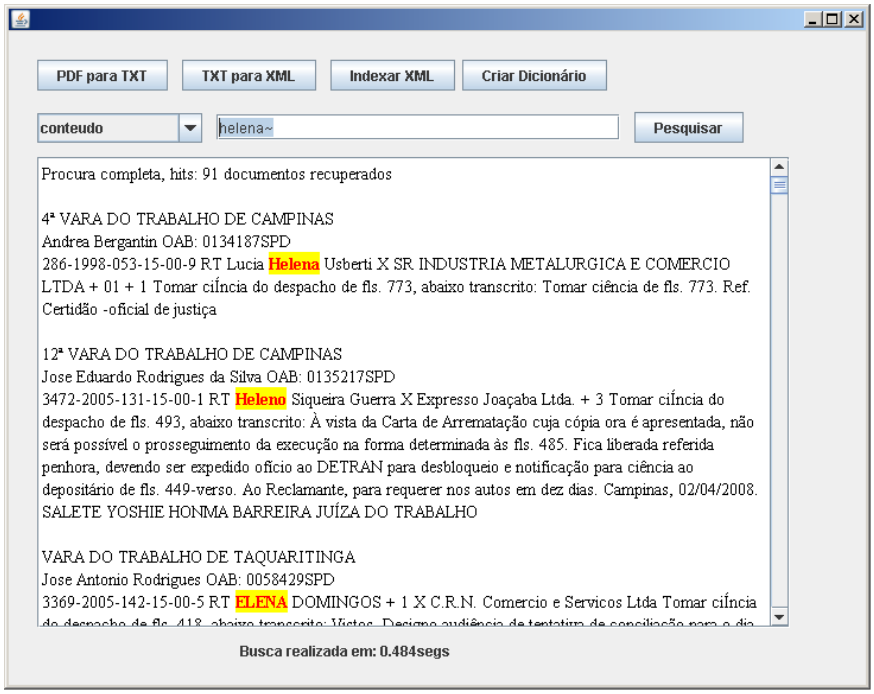

Figura 7. Uso de proximidade de termos na busca

4.1. Análise de desempenho

O protótipo desenvolvido para este trabalho foi submetido a testes de desempenho referentes ao tempo de processo, tamanho de arquivos gerados, exatidão da extração de informação e relevância da busca. Deve ser levado em consideração que para os testes realizados foi utilizado um computador pessoal não dedicado à atividade de recuperação de informação em textos.

Como objetos de teste dispuseram-se quatro publicações do JTRT15 do ano de 2008, cada publicação tendo em média 7000 despachos proferidos em arquivos com aproximadamente 6,5 megabytes.

Tomando como exemplo a publicação do dia 20 de Maio de 2008, com 152 páginas, 7115 despachos proferidos em um arquivo de 6,52 megabytes, o processo de conversão para arquivo texto durou 30 segundos e gerou um arquivo de 5,78 megabytes. O processo de extração de informação do arquivo texto gerado, durou 25 minutos, e retornou um arquivo XML de 5.55 megabytes. Para a criação de índice, o processo durou 2.23 segundos para estrutura de arquivo composto, já na estrutura de arquivos múltiplos a geração durou 2.28 segundos, ambos com tamanho total de 7,3 megabytes no diretório de índice.

O algoritmo de extração de informação utilizado neste trabalho provou ser efetivo no espaço amostral de teste deste trabalho, identificando todos os elementos de seu escopo, recuperando todos os despachos. Porém, as técnicas de programação

utilizadas para executar o algoritmo deste trabalho, precisam ser revistas para diminuir o tempo do processo.

Como o processo de extração de informação delimitou os despachos em pequenos documentos com campos estruturados e bem definidos no índice *Lucene*, o processo de busca supre as necessidades do usuário em relação às varas regionais de trabalho e aos advogados com precisão, pois estes não oferecem complexidade para seu retorno. Quanto a buscas realizadas no conteúdo dos processos, a necessidade comum dos usuários neste tipo de ato jurídico é consultar se consta em algum despacho o nome do interessado, seja este um individuo ou uma empresa ou órgão público, o que o protótipo retorna com exatidão caso a digitação esteja correta.

5. Conclusão

Para os usuários das publicações oficiais, o protótipo proposto traz as vantagens de acessibilidade e velocidade de informação, o que torna a tarefa de procurar despachos no JTRT15 menos penosa. Entretanto, as idéias e ferramentas aplicadas aqui podem ser estendidas a outras seções do mesmo caderno bem como a outras publicações oficiais e mesmo a qualquer outro documento que apresente uma estrutura aparentemente clara, passível de extração de informação.

O trabalho de forma alguma esgota o tema abordado, limitando-se apenas a extração e recuperação de informação não aprofundando em uso de técnicas de PLN como o uso de lista *stopwords* ou *stemming* que aumentariam o desempenho do sistema, e, também, o uso de Tesauro para uma melhor qualidade de resultados de busca. O protótipo de *software* apresenta algumas limitações, como a busca em apenas um campo, demora no processo de extração de informação e a interface gráfica, que ainda não apresenta todas as facilidades de busca de forma intuitiva ao usuário.

Ainda como trabalhos futuros, este trabalho pode servir de camada intermediária para armazenamento em banco de dados, ou também como camada intermediária para uma aplicação que utiliza Raciocínio Baseado em Casos. Um importante avanço seria a criação de uma versão para teste em um ambiente real para análise de seu desempenho.

References

Baeza-Yates, R. and Ribeiro Neto, B. (1999). Modern Information Retrieval, 1st edition. Harlow: Addison-Wesley Longman.

Gospodnetic, O., Hatcher, E. and McCandless, M. (2009). Lucene in Action, 2.ed., Greenwich: Manning Publications.

Jackson, P. and Moulinier, I. (2002) Natural processing for online applications: text retrieval, Extraction and Categorization. Amsterdam: John Benjamins Publishing Co.

Manning, C. D., Raghavan, P. and Schütze, H.. (2008). An Introduction to Information Retrieval, 1st edition. Cambridge: Cambridge University Press.

Moens, M. F. (2006). Information extraction: algorithms and prospects in a retrieval context, 1. ed., Dordrecht: Springer.

STEPHEN, G. A. (1994). String Searching Algorithms. London: World Scientific Publishments Co. Pte. Ltd.