Apache Lucene como indexador de páginas web

Christopher Renkavieski[1], **Edenilson Jonatas dos Passos**[1]

[1]Universidade do Estado de Santa Catarina (UDESC)
– Joinville – SC – Brasil

chris.renka@gmail.com, edenilson.passos@yahoo.com

Abstract. *With the current amount of data that has been dealt with on the past decades, efficient search and indexing engines became a major need on the market. Among this tools, possibly the one that stands out the most, due to its efficiency and versatility, is the Apache Lucene. In this article are presented several aspects of this tool, including it's algorithm, generated files, it's scoring process and documents removal, besides some notes about the performance and the stemming procces, which is one of the most used techniques by Lucene to improve its efficiency. At last, it is presented a practical implementation of the Apache Lucene.*

Resumo. *Com a grande quantidade de dados com que se têm lidado nas últimas décadas, ferramentas de busca e indexação eficiente se tornaram uma grande necessidade no mercado. Entre essas ferramentas, possivelmente a que mais se destaca, devido à sua eficiência e versatilidade, é o Apache Lucene. Neste artigo são apresentados vários aspectos dessa ferramenta, incluindo seu algoritmo, seus arquivos gerados, seu processo de scoring e remoção de documentos, além de algumas notas sobre seu desempenho e sobre o processo de stemming, que é uma das técnicas utilizadas pelo Lucene para aumentar sua eficiência. Por fim, é apresentada uma aplicação prática do Apache Lucene.*

1. Introdução

O Apache Lucene, também conhecido como somente Lucene, é um sofware de busca Open Source, ou mais precisamente um framework que foi criado em 2000 por Doug Cutting na linguagem Java. A principal funcionalidade dessa API é a indexação de documentos [Sigrist and Higashino 2016] [Apache 2016].

Para que uma busca por um determinado conteúdo num amplo receptáculo de conhecimento como a internet, o Lucene baseia-se nos chamados índices. Os índices são estruturas de dados que tornam a localização de termos mais rápida. Porém, a complexidade dessa estrutura é alta, portanto faz-se necessário a divisão em dois grandes grupos de funcionalidades [Sigrist and Higashino 2016].

O primeiro grupo é a construção do índice, que na primeira etapa se dá a aquisição do conteúdo. Quando obtido, eles são transformados em documentos, representações do conteúdo obtido. Documentos esses, todos em formato em comum. Em seguida, esses documentos são analisados para tornar o índice mais eficaz. Nessa etapa são realizadas conversões de letras maiúsculas para minúsculas, eliminação de conectivos. Ou em casos mais complexos as palavras podem até mesmo ser transformadas para suas raízes

morfológicas, no caso as palavras computação e computadores se tornariam apenas “computa”. O resultado desse processo, é um conjunto de tokens, isto é, as palavras que farão parte do índice [Franklin 2000] [Sigrist and Higashino 2016].

Já no segundo grupo os usuários interagem com a API através de uma interface na qual ele pode entrar com a busca desejada. Depois disso, a entrada que o usuário entrou é transformada em um formato definido pelo mecanismo de busca. Essa consulta é feita com o auxílio do índice construído no processo anterior. A figura 1 ilustra brevemente a situação descrita nesses dois grupos descritos anteriormente [Franklin 2000] [Sigrist and Higashino 2016].

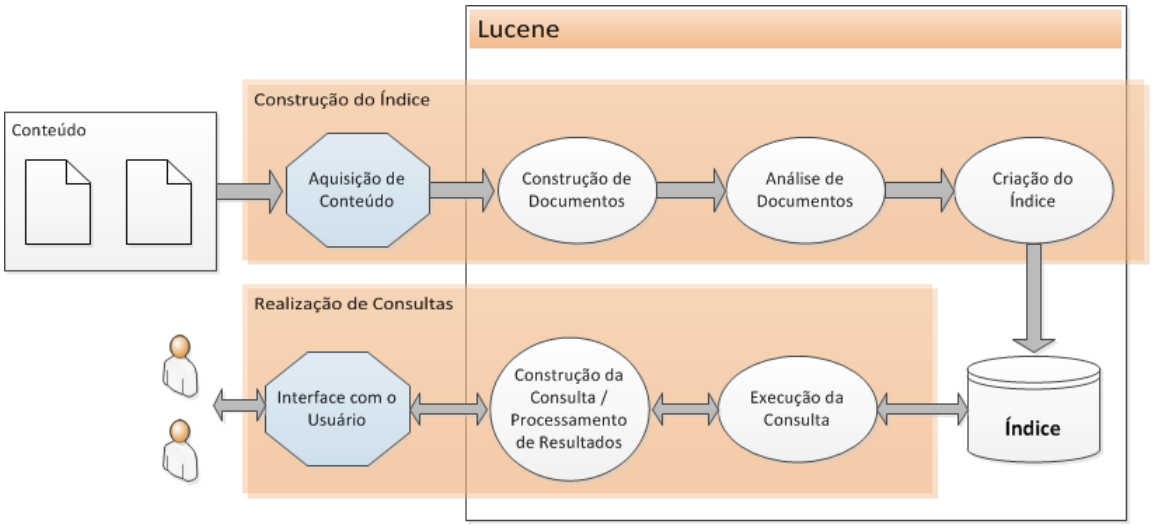

Figura 1. Componentes de uma aplicação de busca.

Neste artigo, as seções de 2 a 7 dão detalhes sobre o funcionamento interno do Apache Lucene. As seções 2 e 3 dão detalhes sobre como o Lucene gera seus índices, explicando cada um dos campos do índice e dos arquivos gerados, respectivamente.

Na seção 4 fala-se do processo de stemming, que é utilizado para se aumentar a eficiência dos índices, e na seção 5 explica-se como funciona a remoção de documentos no Lucene.

A seção 6 trata do processo de scoring do Lucene, utilizado para definir a prioridade das buscas, e na seção 7 são discutidas algumas formas de se aumentar o desempenho da ferramenta.

Por fim, a seção 8 apresenta um exemplo simples de implementação do Lucene, por meio de sua variedade Solr, onde são indexadas quatro páginas web, e a seção 9 conclui discutindo o poder e a importância dessa ferramenta de busca, citando algumas grandes empresas que a utilizam.

2. Algoritmo

O algoritmo de indexição do Apache Lucene se utiliza de um índice invertido para guardar os termos e sua frequência [Apache a]. Por termo, entende-se uma palavra do texto, que são organizados em campos, sendo que um conjunto de campos forma um arquivo a ser indexado. A forma de índice utilizado pelo Lucene é chamado de índice invertido

porque ele usa os termos como chave e lista os documentos que o contém [Apache a], o que é o oposto de uma implementação tradicional, que listaria os termos contidos em cada arquivo. Por fim, cada documento indexado recebe um número inteiro, onde o primeiro recebe o número zero, e cada documento seguinte recebe o número do anterior incrementado em um.

Cada índice criado pelo Lucine é composto de múltiplos segmentos que são, por si só, índices [Apache a]. Cada segmento é composto por:

- Nomes dos campos: conjunto de nomes usados nos índices.
- Valores armazenados nos campos: contém uma lista de pares valor-atributo para cada documento, onde o atributo é o nome do campo. É utilizado para guardar informações auxiliares sobre os documentos, como título ou URL.
- Dicionário de termos: dicionário contendo todos os termos, de todos os campos indexados de todos os documentos. Também guarda o número de documentos que contém o termo, além de apontadores para os próximos dois campos.
- Dados sobre frequência do termo: para cada termo do dicionário, guarda o número de documentos que o contém, e quantas vezes o termo aparece em cada documento.
- Dados sobre proximidade de termos: para cada termo do dicionário, guarda suas posições de ocorrência em cada arquivo.
- Fatores de normalização: para cada campo do documento, guarda um valor que é multiplicado pelo *score* daquele campo. A seção 6 explica melhor o que é o score.
- Vetores de termos: consiste do texto e da frequência de um termo, que é guardado para cada campo do documento.
- Documentos removidos: opcional, indica quais documentos foram removidos.

Tendo esses campos, o Apache Lecene consegue indexar adequadamente e de forma eficiente múltiplos arquivos.

3. Arquivos gerados

O Lucene trabalha internamente com vários arquivos, de extensões diferentes, para poder fazer sua indexação. Abaixo são explicados cada um desses arquivos.

- Arquivo de segmentos: segments.gen, segments_N, informação sobre segmentos.
- Arquivo de bloqueio: write.lock, previne que um arquivo seja sobrescrito
- Arquivo de composição: .cfs, opcional, consiste dos índices dos sistemas que frequentemente não podem ser manipulados.
- Arquivo de composição da tabela de entrada: .cfe, guarda todas as entradas do arquivo .cfs correspondente.
- Campos: .fnm, guarda informações sobre os campos.
- Índice do campo: .fdx, guarda um ponteiro para cada dado.
- Dados do campo: .fdt, campos guardados para os documentos.
- Informação dos termos: .tis, parte do dicionário de termos, guarda as informações dos termos.
- Índice da informação do termo: .tii, índice para o arquivo de informação dos termos.
- Frequências: .frq, lista dos documentos que contém cada termo, junto de sua frequência.

- Posições: .prx, guarda informações da posição onde cada termo ocorre no índice.
- Normas: .nrm, codifica o tamanho e informações de otimização para documentos e campos.
- Índice do vetor de termos: .tvx, guarda desvios no documento de dados.
- Documentos do vetor de termos: .tvd, possui informações sobre cada documento que tem vetores de termos.
- Campos do vetor de termos: .tvf, informação sobre o nível do campo de vetores de termos.
- Documentos removidos: .del, informação de que arquivos foram removidos.

4. Stemming

Um dos processos utilizados pelo Lucene para se aumentar a eficiência da indexação e da busca é o chamado *stemming*, que consiste em reverter palavras para sua origem morfológica, para então indexar essa palavra de origem no lugar da palavra no texto [Porter 2006]. Isso permite que a busca retorne também palavras morfologicamente semelhantes à palavra procurada, pois nem sempre se sabe exatamente o que está no texto, além de diminuir a quantidade de termos únicos indexados.

Nativamente, o Apacehe Lucene realiza o stemming somente de palavras em inglês, e para isso ele utiliza o algoritmo de stemming de Porter. O algoritmo de Porter está disponível em sua página web [Porter 2006] e pode ser analisado em detalhes lá, portanto aqui será dada apenas uma ideia breve de seu funcionamento.

Em essência, o algoritmo analisa as sequências de vogais e consoantes de uma palavra e, com isso, a classifica dentro de certos grupos. Em seguida, é verificado se o sufixo da palavra é comum dentro de seu grupo, o que permite a remoção da maioria dos sufixos como "*s*", "*es*", "*ing*", "*ment*", entre outros. Em seguida, são feitos outros testes para casos específicos do idioma inglês, que no final resultam na raíz da palavra passada como entrada para o algoritmo.

Um algoritmo para stemming do idioma português brasileiro é apresentado por [Alvares et al.], que foi elaborado com base em um gerador léxico web com aproximadamente 130.000 palavras. A ideia do algoritmo é inicialmente verificar se a palavra pertence a alguma categoria especial, e depois tratar o seu prefixo e seu sufixo, sempre levando exeções em consideração. Com isso, consegue-se reduzir palavras em português para sua raíz morfológica.

5. Remoção de documentos

A remoção de documentos do índice do Apacehe Lucene [TutorialsPoint] pode ser realizada das seguintes formas:

- Remover um ou mais documentos em particular;
- Remover todos os documentos que contém um ou mais determinados termos.

As identificações dos documentos removidos são então gravadas no arquivo de documentos removidos, mencionado na seção 3.

6. Scoring

Uma das principais razões da popularidade do Lucene é seu sistema de *scoring* [Apache b]. Por score, entende-se uma espécie de nota ou pontuação dada a um documento, que determina sua prioridade em um processo de busca.

Um exemplo de aplicação prática de scoring vem das ferramentas de busca da web, como por exemplo o *Google*, onde certas páginas aparecem no topo da pesquisa, em destaque, por terem um alto score.

No Lucene, o score de um documento é determinado principalmente pela frequência que o termo buscado aparece no documento, e pela quantidade de documentos que possuem o termo [Apache b]. Além disso, o Lucene também tem ferramentas que permitem incrementar o score de um dado documento durante a indexação, para que, por exemplo, resultados patrocinados tenham prioridade.

7. Desempenho

Há alguns cuidados que o usuário deve ter para que não seja perdido desempenho nas operações de indexação e busca com o Apache Lucene. As referências [Lucene-JavaWiki b] [Lucene-JavaWiki c] detalham múltiplas estratégias para se aumentar o desempenho do Lucene. Algumas delas são:

- Utilizar arquivos de sistema locais;
- Usar o máximo possível de RAM;
- Utilizar um único grande campo de texto para indexação, no lugar de vários campos pequenos;
- Chamar a função IndexReader como somente leitura para a busca;
- Considerar o uso de filtros de pesquisa.

Além disso, também é importante o usuário se manter atento ao fato de que as buscas tendem a perder eficiência com o tempo, caso hajam muitas remoções ou inserções de documentos indexados. Se o desempenho cair demais devido a esses fatores, pode-se considerar fazer uma reindexação completa dos documentos.

8. Implementação

Para a implementação da atividade de indexação proposta, foi possível realizar o download das páginas e sub páginas exigidas conforme o orientado, com o auxílio da linguagem de programação Ruby. O programa é simples, é solicitado a URL ao usuário, e esse digita ou cola o endereço desejado, então o algoritmo, downloadPaginasHtml, salva o conteúdo dessa página conforme solicitado. Caso o usuário digite a página raiz como por exemplo, www.sympla.com.br, ela salva essa página por inteiro como www.sympla.com.br.html. Porém, caso ele entre alguma sub página desse site, o algoritmo salvará somente a sub página. Por exemplo, https://www.sympla.com.br/maya-e-mash-para-motion-graphics__71947, o algoritmo irá salvar um arquivo com o nome de maya-e-mash-para-motion-graphics__71947.html. Feito isso é necessário fazer a indexação desses arquivos na ferramenta de busca e indexação Solr, que é construída com base no Lucene e possui suporte à linguagem Ruby, entre outras.

Os primeiros passos são criar um server standalone do Solr na máquina local. Feito isso o *core* necessita ser criado. O core é uma coleção que um esquema orientado a dados que vai associar o tipo correto do campo dos documentos que vão ser indexados. Tudo isso pode ser feito através do gerenciamento por interface que é disponibilizado quando é criado o server criado no primeiro passo.

Para realizar a indexação dos arquivos o Solr utiliza de comandos específicos que são disponibilizados apenas para sistemas Unix, por isso usuários de Windows devem

instalar uma versão alternativa para fazerem a indexação de forma correta. A indexação possui diversos tipos de aperfeiçoamentos, como tirar as tags do código html, o método de stemming, remoção de palavras de ligação. Cabe ao usuário decidir qual a melhor opção usar. Porém, é preferível um índice mais enxuto, porém eficaz. O desempenho do índice utilizado nesta implementação foi satisfatório no modo default do Solr, deste modo, não foi necessário fazer nenhuma alteração no mesmo. O método aqui utilizado deixou as tags do código html, pois assim facilitou o algoritmo de procura.

O algoritmo de busca, procuraNosIndex, utiliza da busca DisMax. O analisador de consulta DisMax é um instrumento para processar frases simples (sem sintaxe complexa) inseridos pelos usuários para procurar termos individuais em vários domínios. O algoritmo implementado solicita da entrada do usuário e o resultado é número de termos encontrados iguais ao que o usuário solicitou dentro dos índices, assim como todas as páginas que possuem tal termo.

9. Conclusão

Neste artigo, foi apresentada e explanada a ferramente de indexação Apache Lucene. Essa ferramenta elabora seus índices de forma bastante eficiente e versátil, o que, em conjunto com o fato de ser um software Open Source, explica sua grande popularidade no mercado.

Entre as grandes empresas que utilizam o Apache Lucene [Lucene-JavaWiki a], podem-se destacar: AOL, IBM, Apple, Eclipse, Twitter, entre outras.

O uso da ferramenta Solr, baseada no Lucene, ocorreu sem grandes dificuldades e com grande eficiência na indexação das páginas web mencionadas na seção 8, o que é mais um exemplo da usabilidade da ferramenta.

Referências

Alvares, R. V., Garcia, A. C. B., and Ferraz, I. Stembr: A stemming algorithm for the brazilian portuguese language.

Apache. Apache lucene - index file formats.

Apache. Apache lucene - scoring.

Apache (2016). Apache lucene.

Franklin, C. (2000). How internet search engines work.

Lucene-JavaWiki. Applications and web applications using lucene.

Lucene-JavaWiki. How to make indexing faster.

Lucene-JavaWiki. How to make searching faster.

Porter, M. (2006). The porter stemming algorithm.

Sigrist, P. and Higashino, W. A. (2016). Conhecendo o lucene - revista java magazine 104.

TutorialsPoint. Lucene - delete document operation.