

1. Sistema “Lucene” (<http://lucene.apache.org/>) – “nada para instalar”

Apenas na questão 6 poderá ter necessidade de instalar (em `distribution_Lucene*.zip`) para analisar algum código fonte do Lucene. As bibliotecas a utilizar são as `lucene-*-7.5.0.jar` que já estão na pasta “_sistemaCompleto”. De qualquer modo segue abaixo uma breve explicação de como obter e instalar a atual versão do Lucene. Caso realize a instalação deve ter cuidado para não originar conflitos com a estrutura de pastas desta aula prática.

Em <https://lucene.apache.org/core/downloads> tem as últimas versões do sistema; os `*.tgz` têm as bibliotecas e documentação; os `*-src.tgz` código fonte, exemplos e “demos”; cf., `moodle distribution_Lucene*.zip`. **Atenção:** se optar instalar faça-o em pastas diferentes das usadas na aula. Deve também instalar a versão executável em pasta diferente da versão com código fonte (ambas expandem para a mesma pasta). Tudo foi testado com “Java (32bit)”.

2. Indexar e Interrogar – usando o exemplo fornecido no Lucene

Leia o ficheiro “overview.html” que está em “. \src_luceneDemo”, apenas é importante ler as secções “IndexFiles” e “SearchingFiles” (tudo o resto já está preparado nesta aula prática).

- Construa os índices para a coleção que está na pasta “z_coleccao_demo” cujos documentos são o próprio código fonte da demo Lucene. Usar/adaptar o “script” “01_DEMO_indexar.bat”.
- Interroque a coleção indexada na alínea anterior. Usar/adaptar “02_DEMO_interrogar.bat”.

3. Criar e Atualizar Índices – com o exemplo fornecido no Lucene

- Elimine os índices apagando conteúdo da pasta “z02_coleccao_indexada”. Crie novos índices (“01_DEMO_indexar.bat”). Interroque a palavra “class” e note o resultado. Agora, altere o nome de um dos documentos da coleção (na pasta “z02_coleccao_paraIndexar”). Volte a indexar e interrogar a palavra “class”. E o que aconteceu ao documento a que alterou o nome?
- Analise o código de “IndexFiles.java” (. \src_luceneDemo \org \apache \lucene \demo). Altere o script em “01_DEMO_indexar.bat” para conter a opção de execução “-update”. Repita toda a alínea anterior (a) e note o que se passa agora ao documento ao qual altera o nome.

4. Alterar o exemplo fornecido no Lucene

Em “. _src \sistemaCompleto \src_luceneDemo \org \apache \lucene \demo” analise o código: “IndexFiles.java” e de “SearchFiles.java”.

- Altere o que achar necessário de modo a incluir um novo campo (“field”) (onde armazena, por exemplo, o seu nome).
- Use “03_DEMO_alterarLibDemo.bat” e reconstrua a biblioteca “lucene-demo-7.5.0.jar”.
- Execute novamente os exercícios anteriores: 2 e 3. Use a sintaxe “campo:termo” para interrogar o campo que construiu.

5. Indexar e interrogar uma coleção – modelo booleano

Considere um sistema para submeter artigos. Cada artigo submetido será avaliado de modo cego, i.e. sem fornecer aos avaliadores qualquer informação sobre os autores. Para isso, o processo de submissão solicita separadamente o título, a lista de autores, os temas e o conteúdo propriamente dito. Depois, apenas o título e conteúdo são enviados aos avaliadores.

Recorde a aula teórica; pode **usar e adaptar** o código fornecido em “src_myRI\src”.

- Construa uma coleção de artigos. Para iniciar o trabalho pode considerar a coleção composta por oArtigoA.txt, oArtigoA.txt e oArtigoA.txt em “src_myRI”
- Desenvolva uma aplicação para indexar, usando o modelo **booleano**, a coleção que construiu. *Sugestão*: analise o código em “MeuIndexador.java”; compile com “01_compilar.bat” e execute com “02_indexar.bat”.
- Desenvolva uma aplicação para listar o conteúdo dos índices construídos após indexar coleção. *Sugestão*: analise o código em “MeuInfo.java”; compile com “01_compilar.bat” e execute com “03_obterInfo.bat”.
- Desenvolva uma aplicação para interrogar a coleção que indexou na alínea anterior. *Sugestão*: analise o código em “MeuInterrogador.java”; compile com “01_compilar.bat” e execute com “04_interrogar.bat”.

6. Indexar e interrogar uma coleção – modelo vectorial

Recorde a aula teórica; pode **usar e adaptar** o código fornecido em “src_myRI\src”.

- Desenvolva uma aplicação para indexar, com o modelo **vectorial**, a coleção da questão anterior. *Sugestão*: considere o código em “MeuIndexadorVectorial.java”.
- Elimine os índices que construiu com o modelo booleano (da última alínea) bastando para isso remover a pasta “_osMeusIndices”. Construa agora novos índices usando o modelo vectorial. Pode recorrer a “05_indexarVectorial.bat”.
- Analise o conteúdo dos índices (e.g., após executar “03_obterInfo.bat”) e note os valores do “tf” (*term frequency*) de cada termo em cada documento assim como a posição em que cada termo ocorre em cada documento.
- Interroque a coleção usando a aplicação que construiu na questão anterior.

7. Construir um novo analisador – usar e construir (novos) filtros

- a) Construa um novo analisador que aceite um ficheiro com uma lista de “*stop-words*” que não devem ser consideradas na construção do índice. *Sugestão*: analise o código disponível em “MeuAnalisador.java” com o essencial para definir um (novo) “*pipeline*” de transformações sobre os termos de um documento (ou, na nomenclatura Lucene, do *field* de um *document*).
- b) Teste o analisador que construiu na alínea anterior. *Muito importante*: “o analisador usado para indexar deve ser consistente com o usado para interrogar”; ou seja, precisa de indexar e interrogar usando sempre o mesmo (este novo) analisador. Para isso precisa substituir a linha de código (em “MeuIndexador.java” ou em “MeuIndexadorVectorial.java”) onde indica usar o “StandardAnalyzer” por uma linha onde indique usar o “MeuAnalisador” (desenvolvido alínea anterior); alteração idêntica em “MeuInterrogador.java” para indexador o interrogador usarem mesmo analisador na construção de termos. Note que usando o código fornecido nesta aula, as alterações se resumem a (des)comentar 1 única linha de código! *Atenção*: depois das alterações precisa de compilar, eliminar eventuais índices anteriores e a reconstruir os índices usando o novo analisador.
- c) Altere o analisador da alínea anterior de modo a substituir cada carácter acentuado pelo mesmo carácter mas sem acentuação e a transformar os termos em maiúsculas (em vez de minúsculas). *Sugestão*: em “MeuAnalisador.java” identifique e ative o “ASCIIFoldingFilter” e analise o código fonte de “LowerCaseFilter.java” disponível em “lucene-7.5.0\core\src\java\org\apache\lucene\analysis”. Indexe e interogue usando este novo analisador; recorde que “o analisador usado para indexar deve ser consistente com o usado para interrogar”.
- d) Altere o analisador de modo eliminar qualquer termo que tenha menos de 2 caracteres. *Sugestão*: em “lucene-7.5.0\analysis\common\src\java\org\apache\lucene\analysis\miscellaneous” analise o código fonte de “LengthFilter.java”. Indexe e interogue usando este novo analisador; recorde que “o analisador usado para indexar deve ser consistente com o usado para interrogar”.
- e) Altere o analisador de modo a incluir uma fase de “stemming” adequada ao Português. *Sugestão*: em “lucene-7.5.0\analysis\common\src\java\org\apache\lucene\analysis\pt” analise ambos “PortugueseAnalyzer.java” e “PortugueseStemFilter.java”. Indexe e interogue usando este novo analisador; recorde que “o analisador usado para indexar deve ser consistente com o usado para interrogar”.