MG4J: Managing Gigabytes for Java

Exercise

Adriano Fazzone

DIAG Department

Sapienza University of Rome

Important Software Modules

- *Document*.
- *FileSetDocumentCollection*.
- *DocumentFactory*.
- *IndexBuilder*.
- *Query*.

Document

- Indexing in MG4J is centered around documents
- Package: *it.unimi.di.big.mg4j.document*
- The object document, which is the instance of the class *Document*, represents a single document that can be indexed
- Different documents have different number and type of fields.
- For example,
  - E-mail: from, to, date, subject, body
  - HTML page: title, URL, body

FileSetDocumentCollection

- Package: *it.unimi.di.big.mg4j.document*

- The main method of *FileSetDocumentCollection* allows to build and serialize a set of documents specified by their filenames

Document Factory

- Package: *it.unimi.di.big.mg4j.document*

- The factory turns a pure stream of bytes (file) into a document made by several fields (title and text, for example)

Standard MG4J Document Factories

- **HtmlDocumentFactory**
- CompositeDocumentFactory
- IdentityDocumentFactory
- MailDocumentFactory
- PdfDocumentFactory
- ReplicatedDocumentFactory
- PropertyBasedDocumentFactory
- TRECHeaderDocumentFactory
- ZipDocumentCollection.ZipFactory

Query

- Package: *it.unimi.di.big.mg4j.query*
- To query the index we can use the main method of the class *Query*
- We can submit queries by using:
  - command line
  - web browser
- *QueryEngine*: The query engine receives the query and returns the ranked list of results
- *HttpQueryServer*: A simple web server for query processing

First Exercise: your turn :)

1. Create an Inverted Index on an a set of html pages of DIAG department: **htmlDIS.tar.gz**

2. Follow step by step the instructions for the exercise and try different settings/queries.

3. For any problem, have a look at MG4J manual: http://mg4j.di.unimi.it/man/manual.pdf

Indexing and querying: exercise

- TECHNICAL REQUIREMENTS:
  - UNIX Operating System
  - Java (>=6)
- Document collection and libraries are available at:

  http://www.diag.uniroma1.it/~fazzone

Set the classpath

- Download and extract **htmlDIS.tar.gz**
- Download and extract **webir_lab_lib.zip**
- Download the file **set-my-classpath.sh**
- Edit the first line of the file **set-my-classpath.sh**: replace ***your_directory*** with the path of the folder containing all the ***.jar*** files (**lib** folder)
- Set the CLASSPATH: **source set-my-classpath.sh**

Building the collection of documents (1)

- Help:

  java it.unimi.di.big.mg4j.document.FileSetDocumentCollection --help

- Create the collection:

  find htmlDIS -iname *.html | java it.unimi.di.big.mg4j.document.FileSetDocumentCollection -f HtmlDocumentFactory -p encoding=UTF-8 dis.collection

  find returns the list of files, one per line. This list is provided as input to the main method of the *FileSetDocumentCollection*

Building the collection of documents (2)

- We need also to specify a factory (the -f option) and the encoding as a property
- The name of the collection is **dis.collection**
- The collection does not contain the files, but only their names
- Deleting or modifying files of **htmlDIS** directory may cause inconsistence in the collection

Building the index

- Help: **java it.unimi.di.big.mg4j.tool.IndexBuilder --help**
- Create the index:

  java it.unimi.di.big.mg4j.tool.IndexBuilder --downcase -S dis.collection dis

  - **--downcase**: this option forces all the terms to be downcased
  - **-S** : specifies that we are producing an index for the specified collection. If the option is omitted, Index expects to index a document sequence read from standard input :o
  - **dis**: basename of the index
- If you have memory problem, you can use **-Xmx** for allocating more memory to Java:

  java –Xmx3G it.unimi.di.big.mg4j.tool.IndexBuilder --downcase -S dis.collection dis

Index files (1)

- *dis-{text,title}.terms*: contain the terms of the dictionary. One term per line

  more dis-text.terms

- *dis-{text,title}.stats*: contain statistics

  more dis-text.stats

- *dis-{text,title}.properties*: contain global information

  more dis-text.properties

Index files (2)

- *dis-{text,title}.frequencies*: for each term, there is the number of documents with the term (γ-code)
- *dis-{text,title}.globcounts*: for each term, there is the number of occurrence of the term (γ-code)
- *dis-{text,title}.offset*: for each term, there is the offset (γ-code)

Index files (3)

- *dis-{title,text}.sizes*: contain the list of the document sizes. The document size is the number of words contained in each document (γ- code)
- *dis-{text,title}.batch<i>*: temporary files with sub-indices (γ-code). Use the option **--keep-batches** to not delete temporary files
- *dis-{text,title}.index*: contain the index (γ-code)

Querying the index

- Help:
  java it.unimi.di.big.mg4j.query.Query --help

- Querying the index:
  java it.unimi.di.big.mg4j.query.Query -h -i FileSystemItem -c dis.collection dis-text dis-title

- Command line: **{text, title} > computer**
- Web browser: **http://localhost:4242/Query**

Query (1)

- **Search one word**: The result is the set of documents that contain the specified word
  - Example: **computer**

- **AND**: more than one term separated by whitespace or by AND or &. The result is the set of documents that contain all the specified words
  - Example: **computer science**
  - Example: **computer AND science**
  - Example: **computer & science**

Query (2)

- **OR**: more than one term separated by OR or |. The result is the set of documents that contain any of the given words
  - Example: **conference | workshop**

- **NOT**: the operator NOT or ! is used for negation
  - Example: **conference & ! workshop**

- **Parentheses**: the parentheses are used to enforce priority in complex queries
  - Example: **university & (rome | california)**

Query (3)

- **Proximity restriction**: the words must appear within a limited portion of the document
  - Example: **(university rome)~6**

- **Phrase**: using **“ ”** we can look for documents that contain the exact phrase
  - Example: **“university of rome la sapienza”**

- **Ordered AND**: more than one term separated by **<**
  - Example: **computer < science < department**

Query (4)

- **Wildcard (*):** wildcard queries can be submitted appending * at the end of a term
  - Example: **infor***

- **Index specifiers**: prefixing a query with the name of an index followed by **:** you can restrict the search to that index
  - Example: **title:computer**
  - Example: **text:computer science AND title:FOCS**

Sophisticated queries (1)

- MG4J provides sophisticated query tuning
- To use this features, we must use the command line interface
- **$** --- to get some help on the available options
- Some examples:
  - **$mode** --- to choose the kind of results

    Example: **> $mode short**
  - **$selector** --- to choose the way the snippet or intervals are shown

    Example: **> $selector 3 40**

Sophisticated queries (2)

- Other examples:
  - **$mplex** --- when multiplexing is on, each query is multiplexed to all indices. When a scorer is used, it is a good idea to use multiplexing

    Example: **> $mplex on**
  - **$score** --- to choose the scorer

    Example: **> $score VignaScorer**
  - **$weight** --- to change the weight of the indices. This is useful when multiplexing is on

    Example: **>$weight text:1 title:3**

Scorer (1)

- Scorer are important for ranking the documents result of a query.

  Default: **BM25Scorer** and **VignaScorer**
- *ConstantScorer*. Each document has a constant score (default is 0)

  >$score ConstantScorer
- *CountScorer*. It is the product between the number of occurrences of the term in the document and the weight assigned to the index

  >$score CountScorer

Scorer (2)

- *TfIdfScorer*. It implements TF/IDF

  TF is the term frequency of the term *t* for the document *d*: *c/l*; where *c* is the number of occurrences of *t* in *d* and *l* is the length of *d*

  IDF is the inverse document frequency of the term *t* in the collection: *log(N/f);* where *N* is the number of documents in the collection and *f* is the number of documents where *t* appears

  >$score TfIdfScorer

Second Exercise: your turn, again :)

1. Create an Inverted Index on an a set of html pages representing songs: **lyrics_collection.zip**

2. Find queries able to produce wrong results.

3. Create an Inverted Index on an a “clean” set of html pages representing songs: **CLEAN_lyrics_collection.zip**

4. Execute the challenging queries for the first collection on the second collection. Do you have wrong results now? Why?

5. For any problem, have a look at MG4J manual: http://mg4j.di.unimi.it/man/manual.pdf