A Practical Overview

The Terrier Development team
Information Retrieval Group
Department of Computing Science
University of Glasgow

13th December 2005

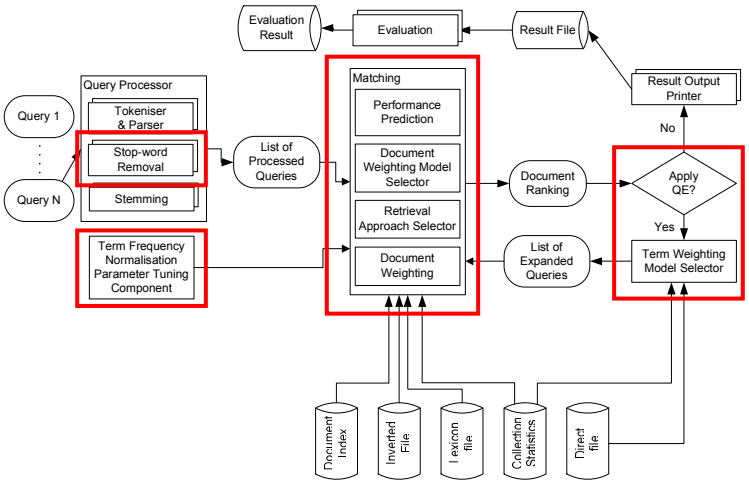
Outline

- ***What is Terrier?***
- Using Terrier
  - Installing
  - Indexing
  - Configuring
  - Retrieving
  - Evaluation
- Understanding & extending Terrier
  - Indexing API
  - Querying API
  - Compiling
- Putting it altogether

What is Terrier?

- Research project
  - 3 researchers, 5 PhD students and 5 programmers
  - Part of its outcome is released as open source software
- Evaluation of Terrier
  - TREC Web, Robust, Terabyte, and Enterprise tracks
  - CLEF Ad-hoc and Web tracks
- Applications
  - Desktop search application
  - Web search engine

Research in the Terrier Project



Goals of Open Source Terrier

- Effective retrieval models
  - Divergence From Randomness (DFR) framework
- Easy cross-comparison of different models
  - Classical models, such as *tf-idf* and BM25
- Cross-platform developed in Java
  - runs on Windows, *nix, MacOS X
- Indexing and Querying APIs
  - Easy to extend – adapt for new applications
  - Modular architecture
  - Simple to start working with
  - Many configuration options

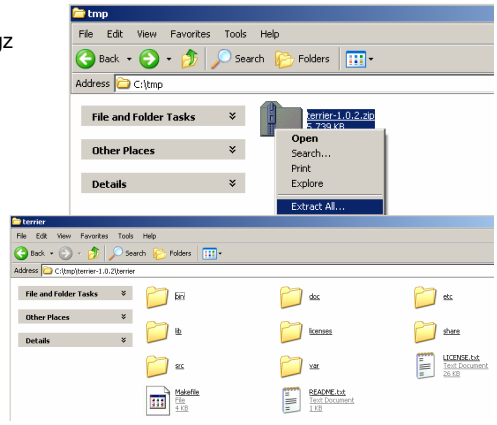
Outline

- What is Terrier?
- ***Using Terrier***
  - ***Installing***
  - ***Indexing***
  - ***Configuring***
  - ***Retrieving***
  - ***Evaluation***
- Understanding & extending Terrier
  - Indexing API
  - Querying API
  - Compiling
- Putting it altogether

Installing Terrier

- What do I need:
  - Sun Java 1.4 or newer
  - The package, e.g. terrier-1.0.2.tar.gz or terrier-1.0.2.zip
  - Linux / Windows operating system

```
[toto@boano tmp]$ ls
terrier-1.0.2.tar.gz
[toto@boano tmp]$ tar -xzf terrier-1.0.2.tar.gz
[toto@boano tmp]$ ls
terrier  terrier-1.0.2.tar.gz
[toto@boano tmp]$ cd terrier
[toto@boano terrier]$ ls
bin  etc  licenses     Makefile    share  var
doc  lib  LICENSE.txt  README.txt  src
[toto@boano terrier]$
```



- Terrier is pre-compiled
  - ***No need to compile***

Directory structure of Terrier

- The directories of Terrier are
  - bin/ : contains useful scripts for running Terrier
  - etc/ : contains the configuration files
  - doc/ : contains the documentation of Terrier
  - lib/ : contains the compiled Terrier classes and the external libraries used by Terrier
  - licenses/ : contains the license information of the components included with Terrier
  - share/ : contains a stop word list, an example of documents to test with Terrier, and other infrequently changing files
  - src/ : contains the source code of Terrier
  - var/index : contains the data structures
  - var/results : contains the retrieval results

Indexing with Terrier (1)

- Terrier can readily index tagged and TREC formatted test collections

TREC AP Collection

```
<DOC>
<DOCNO> AP890101-0002 </DOCNO>
<FILEID>AP-NR-01-01-89 2359EST</FILEID>
<FIRST>r a PM-FutureFactory     01-01 0872</FIRST>
<SECOND>PM-Future Factory,0897</SECOND>
<HEAD>University Erects A Factory Of The
    Future</HEAD>
<HEAD>Eds: Also in Monday AMs report.</HEAD>
<BYLINE>By DONNA BRYSON</BYLINE>
<BYLINE>Associated Press Writer</BYLINE>
<DATELINE>ROLLA, Mo. (AP) </DATELINE>
<TEXT>For students working in a miniature factory
    at the University of Missouri-Rolla, the
    future of American business is now...</TEXT>
</DOC>
<DOC>...
```

TREC .GOV2 Collection

```
<DOC>
<DOCNO>GX010-60-0164440</DOCNO>
<DOCHDR>
http://www.emsc.nysed.gov/repcrd2003/links/sg29.html
HTTP/1.1 200 OK
Server: Netscape-Enterprise/3.6 SP1
Date: Wed, 10 Dec 2003 08:52:41 GMT
Content-type: text/html
Last-modified: Mon, 19 May 2003 20:49:43 GMT
Content-length: 17183
Accept-ranges: bytes
Connection: close
</DOCHDR>
<html>
<head>
<title>Similar Schools Group #29 for 2001-2002</title>
</head>
<body bgcolor="#FFFFFF">
<p align="center"><img src="../images/emscban.gif"
    width="476" height="111" alt="Office of
    Elementary, Middle, Secondary and Continuing
    Education"></p>...
</DOC>
<DOC>...
```


Indexing with Terrier (2)

- Setup Terrier with:

  `bin/trec_setup.sh /path/to/collection`

- Makes a default configuration properties file *etc/terrier.properties*
- Creates a *etc/collection.spec* that contains a list of files to index
  - *Before proceeding, it's worth checking that the etc/collection.spec file contains only the files you want to index*

Indexing with Terrier (3)

```
[toto@boano terrier]$ bin/trec_setup.sh /path/to/collection/
Setting TERRIER_HOME to /users/toto/tmp/terrier
Setting JAVA_HOME to /local/java/linux/jdk1.5.0
Creating collection.spec file.
Creating trec.qrels file.
Creating topics file.
Creating models file.
Creating query expansion models (qemodels) file.
Creating terrier.properties file.
#add the files to index
/path/to/collection/AP890103.gz
/path/to/collection/AP890104.gz
...
/path/to/collection/AP891231.gz
/path/to/collection/README.gz
/path/to/collection/AP890101.gz
/path/to/collection/AP890102.gz
Updated collection.spec file. Please check that it contains
all and only all the files to be indexed, or create it manually.
[toto@boano terrier]$
```


Indexing with Terrier (4)

- Index the collection with `bin/trec_terrier.sh -i`

```
[toto@boano terrier]$ bin/trec_terrier.sh -i
Setting TERRIER_HOME to /users/toto/tmp/terrier
Setting JAVA_HOME to /local/java/linux/jdk1.5.0
read collection specification
Processing /path/to/collection/AP890103.gz
Processing /path/to/collection/AP890104.gz
Processing /path/to/collection/AP890105.gz
Processing /path/to/collection/AP890106.gz
Processing /path/to/collection/AP890107.gz
Processing /path/to/collection/AP890108.gz
...
Finished building the inverted index...
Time elapsed for inverted file: 239.298
Time elapsed: 1119.203 seconds.
[toto@boano terrier]$
```

- Indexing time for AP collection (242918 documents) ~ 20 mins

Indexing with Terrier (5)

The indexing process generates files in directory var/index:

1. Document index (data.docid) — set $\{doc_i\}$
2. Vocabulary/Lexicon (data.lex) — set $\{kw_j\}$
3. Direct index (data.df) — $doc_i \xrightarrow{about} \{kw_j\}$
4. Inverted index (data.if) — $kw_j \xrightarrow{describes} \{doc_i\}$
5. Collection statistics (data.log)

Configuring Terrier (1)

- You can configure Terrier by editing the file ***etc/terrier.properties***

```
#directory names
terrier.home=/users/toto/tmp/terrier
...
#stop-words file
stopwords.filename=stopword-list.txt
...
#the processing stages a term goes through
termpipelines=Stopwords,PorterStemmer
```

- Look at ***etc/terrier.properties.sample*** for examples

Configuring Terrier (2)

- Use a different stemmer

  ```
  termpipelines=Stopwords,WeakPorterStemmer
  ```

  – {PorterStemmer,WeakPorterStemmer,<blank>}
- Disable removing of stopwords

  ```
  termpipelines=PorterStemmer
  ```

  – {Stopwords,<blank>}
- Show terms in pipeline

  ```
  termpipelines=DumpTerm,PorterStemmer,DumpTerm
  ```
- Save exact positions of terms in order to do “phrase search”

  ```
  block.indexing=true
  block.size=1
  ```


Configuring Terrier (3)

- Collection tags
  - Which is the document delimiter?
  - Which is the document identifier?
  - Which parts of the documents to index?
  - Collection specific and HTML tags
  - Tags indicating the document’s language
- Specify which tags will be indexed

  ```
  TrecDocTags.doctag=DOC
  TrecDocTags.idtag=DOCNO
  TrecDocTags.skip=DOCHDR
  ```
- Index only the titles of documents

  ```
  TrecDocTags.doctag=DOC
  TrecDocTags.idtag=DOCNO
  TrecDocTags.process=TITLE
  ```

```
<DOC>
<DOCNO>DOC-X1</DOCNO>
<DOCHDR>
. . .
</DOCHDR>
<TITLE>. . .</TITLE>
. . .
<H1>. . .</H1>
</DOC>
```

Configuring Terrier (4)

- Index fields
  - Does a term occur in a given field?
  - Collection specific and HTML tags
  - Tags indicating the document's language

- Specify which tags to check
  `FieldTags.process=TITLE,H1`

- For each posting in the index, Terrier saves a 1 bit flag per field

```
<DOC>
<DOCNO>DOC-X1</DOCNO>
<DOCHDR>
. . .
</DOCHDR>
<TITLE>. . .</TITLE>
. . .
<H1>. . .</H1>
</DOC>
```


Interactive Retrieval with Terrier

- Run the script `bin/interactive_terrier.sh`

```
[toto@boano terrier]$ bin/interactive_terrier.sh
Setting TERRIER_HOME to /users/toto/tmp/terrier
Setting JAVA_HOME to /local/java/linux/jdk1.5.0
time to intialise indexes : 0.269
Please enter your query: cellular
1 : cellular
weighting model: PL2c1.0
1: cellular with 451 documents (TF is 1216).
number of retrieved documents: 451

        Displaying 1-451 results
0 AP900725-0227 210549 9.964763620794955
1 AP900523-0277 196850 9.912967288227696
...
449 AP901009-0235 225761 1.7048076030739692
450 AP900808-0105 213356 1.4994280197490206
Please enter your query: <return>
[toto@boano terrier]$
```

Query Language (1)

- Terrier has an advanced query language with the following operators

```
t1 t2 : retrieves documents with either t1 or t2
t1^2.3: the weight of t1 is boosted to 2.3
+t1 -t2: retrieve docs with t1 but not t2

"t1 t2": retrieve docs with the phrase 't1 t2'
"t1 t2"~n: retrieve docs where the terms t1, t2
  appear within the given distance
```

  - Requires indexing with blocks

Query Language (2)

- More query language operators
  - `+(t1 t2): both terms t1 and t2 are required`
  - `field:t1 : retrieves docs where t1 appears in the specified field`
  - `control:on/off : enables or disables a given control`
    - → like properties, but for query settings
    - → enable query expansion with `qe:on`
- Controls are used to control the querying process on a per-query basis

  `querying.default.controls=c:1.0,start:0,end:999`
- To avoid potential security problems, a list of allowed control is defined as follows:

  `querying.allowed.controls=c,scope,qe,qemodel,start,end`

Batch Retrieval with Terrier

- Example TREC topic

```
<top>
<head> Tipster Topic Description
<num> Number:  051
<dom> Domain:  International Economics
<title>   Airbus Subsidies
<desc> Description:
Document will discuss government assistance to Airbus Industrie, or mention a
   trade dispute between Airbus and a U.S. aircraft producer over the issue of
   subsidies.
<narr> Narrative:
A relevant document will cite or discuss assistance to Airbus Industrie by
   the French, German, British or Spanish government(s), or will discuss a
   trade dispute between Airbus or the European governments and a U.S.
   aircraft producer, most likely Boeing Co. or McDonnell Douglas Corp., or
   the U.S. government, over federal subsidies to Airbus.
</top>
```

- Specify the topics file in *etc/trec.topics.list*

```
[toto@boano]$ cat etc/trec.topics.list
/path/to/topics
[toto@boano]$
```


Configuring Batch Retrieval (1)

- In the properties file, specify whether to use short, normal, or long queries

```
#short: title only
TrecQueryTags.doctag=TOP
TrecQueryTags.idtag=NUM
TrecQueryTags.process=TOP,NUM,TITLE
TrecQueryTags.skip=DESC,NARR

#normal: title + description
TrecQueryTags.doctag=TOP
TrecQueryTags.idtag=NUM
TrecQueryTags.process=TOP,NUM,TITLE,DESC
TrecQueryTags.skip=NARR

#long: title + description + narrative
TrecQueryTags.doctag=TOP
TrecQueryTags.idtag=NUM
TrecQueryTags.process=TOP,NUM,TITLE,DESC,NARR
```

```
<top>
<num> Number: TOPIC-X1
<title> . . . </title>
<desc> . . . </desc>
<narr> . . . </narr>
</top>
```

Configuring Batch retrieval (2)

- Set the weighting models to use
  - Divergence From Randomness (DFR) framework models, such as PL2
  - Classical models, such as *tf-idf*, BM25

```
[toto@boano]$ cat etc/trec.models
PL2
[toto@boano]$
```

- You can specify more than one weighting models in *etc/trec.models,* allowing the easy evaluation of many weighting models

```
[toto@boano]$ cat etc/trec.models
PL2
In_expB2
[toto@boano]$
```

- Terrier will process all queries for each weighting model

Let's (batch) retrieve!

- Using trec_terrier.sh script to retrieve all queries

```
[toto@boano terrier]$ bin/trec_terrier.sh -r
 Setting TERRIER_HOME to /users/toto/tmp/terrier
 Setting JAVA_HOME to /local/java/linux/jdk1.5.0
 time to intialise indexes : 0.226
 Extracting queries from 51-200.topics
 051 : airbus subsidies
 processing query 051
 weighting model: InL2c1.0
 1: subsidi with 2781 documents (TF is 5468).
 2: airbu with 632 documents (TF is 1499).
 number of retrieved documents: 1000
 time to process query: 0.262
 ...
```

- Results are stored in the folder var/results/, numbered…
  - E.g. InL2_c1.0_0.res then InL2_c1.0_1.res etc

Query Expansion

- Automatically extracts informative terms from top ranked documents and adds them to the query
- Use query expansion when batch querying

```
bin/trec_terrier.sh -r -q
```

- Specify the query expansion model

```
[toto@boano]$ cat etc/qemodels
Bo1
[toto@boano]$
```

- Available models: Bo1, Bo2 and KL
- The weights of the added terms in the new query are adjusted with:
  - Rocchio's beta: `rocchio_beta=0.5`
  - Terrier's parameter-free formula: `parameter.free.expansion=true`

Term re-weighting for query expansion

Term	Weight	Term	Weight
Scottish	*1.5000*	*highland*	*1.4087*
games	*1.3105*	*Ligonier*	*0.3609*
kilt	*0.2897*	*caber*	*0.1347*
clan	*0.1291*	*tradition*	*0.1189*
dance	*0.1115*	*Celtic*	*0.1067*
toss	*0.1062*	*dancer*	*0.1013*
grandfather	*0.1009*	*Scot*	*0.0895*
athlete	*0.0745*	*heather*	*0.0673*
artist	*0.0643*	*heavy*	*0.0606*
tartan	*0.0587*	*competitor*	*0.0427*

Illustrative Example of Query Expansion

- TREC Query: *Scottish highland games*
- What are the possible expanded query terms?
- The expanded query (Using one of Terrier's QE mechanisms and Weak Stemming):
  - *Scottish highland games* Ligonier kilt caber clan toss Scot tartan grandfather artist heavy tradition dance Celtic dancer athlete heather competitor
- In the expanded query (using the relevance assessment)
  - These terms are helpful: Ligonier kilt caber clan toss Scot tartan
  - These terms bring noise: grandfather artist heavy
  - The rest of the added query terms are neutral: dancer, tradition, etc.

Experiments with ad-hoc Retrieval

- TREC2005 Terabyte Track ad-hoc task
- Each query has many relevant documents
- Using TF-IDF without query expansion, MAP=0.3024
- Query expansion improves the performance
- The setting of Rocchio's beta affects the retrieval performance
- The parameter-free query expansion provides is robust without requiring tuning.

β	*MAP*
0.1	*0.3303*
0.2	*0.3382*
0.3	*0.3432*
0.4	*0.3426*
0.5	*0.3413*
0.6	*0.3401*
0.7	*0.3384*
0.8	*0.3365*
0.9	*0.3347*
1.0	*0.3327*
free	*0.3428*

Experiments With Topic-Distillation Task

- TREC2003 topic distillation task
  - It involves finding a list of key resources for a particular topic
- Each query has only a few relevant documents
- Using TF-IDF without query expansion, MAP=0.0970
- Query expansion leads to a degradation of the retrieval performance when there are only a few relevant documents

β	*MAP*
0.1	*0.0607*
0.2	*0.0550*
0.3	*0.0524*
0.4	*0.0507*
0.5	*0.0501*
0.6	*0.0485*
0.7	*0.0478*
0.8	*0.0471*
0.9	*0.0469*
1.0	*0.0464*
free	*0.0497*

Evaluation

- *How well did the system perform?*
- Specify the qrels file with the relevance assessments to use in *etc/trec.qrels*

```
[toto@boano]$ cat etc/trec.qrels
/path/to/qrels
[toto@boano]$
```

- Evaluate all the result files in the var/results directory

```
[toto@boano]$ bin/trec_terrier -e
Setting TERRIER_HOME to /users/toto/tmp/terrier
/users/toto/tmp/terrier/var/results/InL2c1.0_0.res
Average Precision: 0.0806
Time elapsed: 0.26 seconds
```

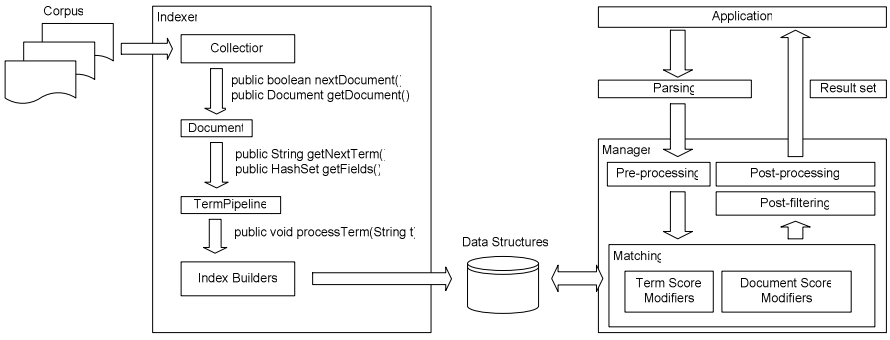
Outline

- What is Terrier?
- Using Terrier
  - Installing
  - Indexing
  - Configuring
  - Retrieving
  - Evaluation
- ***Understanding & extending Terrier***
  - ***Indexing API***
  - ***Querying API***
  - ***Compiling***
- Putting it altogether

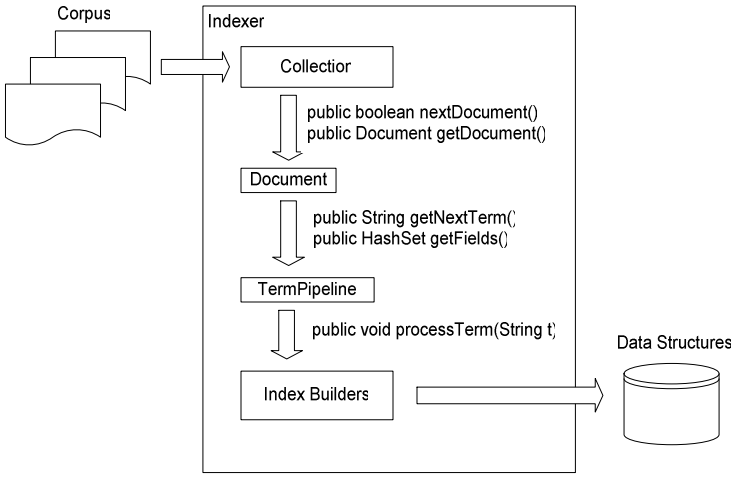
Terrier Architecture

Indexing API

Querying API



Indexing API



Collection and Document

- If you want to parse your own collection, you need to:
  - implement the `Collection` interface for obtaining documents from the collection

```
public Document getDocument();
public boolean nextDocument();
public String getDocid();
public boolean endOfCollection();
```

  - implement the `Document` interface for parsing the documents

```
public String getNextTerm();
public boolean endOfDocument();
```

- See also:
  - doc/indexing.html
  - doc/javadoc/uk/ac/gla/terrier/indexing/Collection.html
  - doc/javadoc/uk/ac/gla/terrier/indexing/Document.html
- Examples in package `uk.ac.gla.terrier.indexing`:
  - TRECCollection -> TRECDocument
  - SimpleFileCollection -> FileDocument; HTMLDocument; PDFDocument; MSWordDocument

Term Pipeline

- When terms are indexed, they are passed through the TermPipeline
  - You can implement your own TermPipeline objects
  - Alter/remove/add terms as they pass through the term pipeline
- Examples found in package `uk.ac.gla.terrier.terms`
  - Stemming, Removing stopwords, Noun phrase extraction, etc etc

```
public class DumpTerm implements TermPipeline {
  TermPipeline next = null;
  public DumpTerm(TermPipeline next) {
    this.next = next;
  }
  public void processTerm(String t) {
    if (t == null)
      return;
    System.err.println("term: "+t); //display term
    next.processTerm(t); //pass onto next term pipeline object
  }
}
```


Data Structures Builders

- Builders for the 4 main data structures
  - Lexicon and Lexicon index : stores the vocabulary
  - Document Index : stores information about documents
  - Direct File (used for fast query expansion) : stores the terms for each document
  - Inverted File : stores the postings lists
- Found in package `uk.ac.gla.terrier.structures.indexing`
- Data structures size for the .GOV2 (426GB) in a distributed setting with 7 query servers

	Content-only	With anchor text
Total size	**17.48GB**	**18.29GB**
Inverted index size	7.77GB	8.47GB
Direct index size	7.00GB	7.70GB
Lexicon size	1.84GB*	1.25GB
Doc index size	0.87GB	0.87GB

* Includes the size of a lexicon with global statistics

Lexicon

- Stores information about the vocabulary – which terms are in collection

```
public boolean findTerm(int termId)
public boolean findTerm(String term)

public String getTerm()
public int getTermId()
public int getTF()
public int getNt()
```

```
public long getStartOffset()
public byte getStartBitOffset()
public long getEndOffset()
public byte getEndBitOffset()

public int getNumberOfLexiconEntries()
```

- Found in package `uk.ac.gla.terrier.structures`
- Using lexicon as a random access file
  - `Lexicon`
- Using lexicon as a stream
  - `LexiconInputStream`
  - `LexiconOutputStream`

Lexicon	**Term** (20 bytes), **Term id** (4 bytes), **Document frequency** (4 bytes), **Term Frequency** (4 bytes), **End byte offset in inverted file** (8 bytes), **End bit offset in inverted file** (1 byte)
Lexicon Index	**Offset of an entry in the lexicon** (8 bytes)

Document Index

- Stores information about documents

```
public String getDocumentNumber(int docid)
public int getDocumentId(String docno)
public int getDocumentLength(int docid)
public int getDocumentLength(String docno)

public byte getStartBitOffset()
public byte getEndBitOffset()
public long getStartOffset()
public long getEndOffset()

Public int getNumberOfDocuments()
```

- Found in package `uk.ac.gla.terrier.structures`
- Using document index as a random access file
  - `DocumentIndex`
  - `DocumentIndexInMemory`
  - `DocumentIndexEncoded`
- Using document index as a stream
  - `DocumentIndexInputStream`

Document Index	**Document id** (4 bytes), **Document Length** (4 bytes), **Document number** (20 bytes), **End byte offset in direct file** (8 bytes), **End bit offset in direct file** (1 byte)

Direct Index

- Useful for fast query expansion or clustering
- Stores the terms that are contained in each document

```
public int[][] getTerms(int docid)
```

- The method `getTerms` returns a two dimensional array:

```
int[][] terms = getTerms(docid);
terms[0] //contains term identifiers
terms[1] //contains term frequencies in the document
terms[2] //is null, or contains field information if fields are indexed
```

- If blocks are indexed

```
terms[4] //contains the number of blocks in which a term appears
terms[5] //contains the block identifiers
```

- (The length of terms[5] is different from the length of terms[4])
- Found in package `uk.ac.gla.terrier.structures`
- Using direct index as a random access file

```
DirectIndex, BlockDirectIndex
```

- Using direct index as an input stream

```
DirectIndexInputStream, BlockDirectIndexInputStream
```

Direct Index	**Term id gap** (gamma code), **Term frequency** (unary code), **Fields** (# of fields bits), **Block frequency** (unary code), **[Block id gap** (gamma code)**]**

Inverted Index

- Stores the posting lists

```
public int[][] getDocuments(int termId)
```

- The method `getDocuments` returns a two dimensional array:

```
int[][] postings = getDocuments(termId);
postings[0] //contains document identifiers
postings[1] //contains term frequencies in the document
postings[2] //is null, or contains field information if fields are indexed
```

- If blocks are indexed

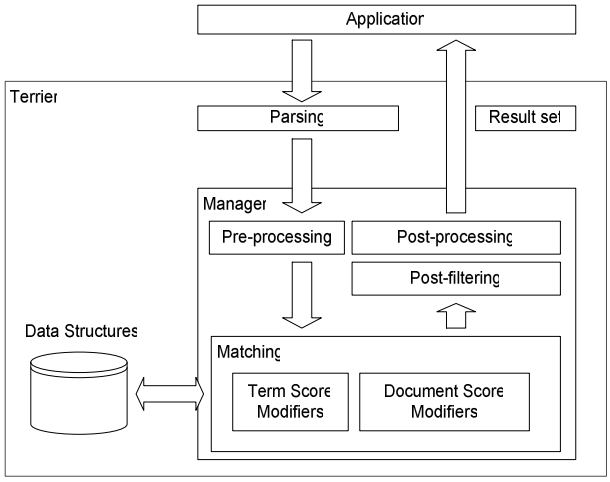
```
postings[4] //contains the number of blocks in which a term appears
postings[5] //contains the block identifiers
```

- The length of postings[5] is different from the length of postings[4]
- Found in package `uk.ac.gla.terrier.structures`
- Using inverted index as a random access file

```
InvertedIndex
BlockInvertedIndex
```

Inverted Index	**Document id gap** (gamma code), **Term frequency** (unary code), **Fields** (# of fields bits), **Block frequency** (unary code), **[Block id gap** (gamma code)**]**

Retrieving API



Before Matching

- Parsing the query
- Pre-processing the parsed query for matching (TermPipeline)

Matching

- The class Matching takes as input:
  - A query
  - The data structures
- Retrieves and ranks documents according to a weighting model
  - Returns a ResultSet
- Found in package `uk.ac.gla.terrier.matching`
- The output of the matching can be modified by applying Term Score Modifiers and Document Score Modifiers

Weighting Models for Matching

- Abstract class Model
  - WeightingModel
    - BB2, IFB2, I(n_e)C2, I(n_e)B2, InL2, PL2, DLH, DFR_BM25
    - BM25
    - tf-idf

```
public class MyModel extends WeightingModel {
  . . .
  public final String getInfo() { return "MyModel"; }
  public double score(double tf, double length) { ...; return score; }
  public double score(double tf,
                      double length,
                      double n_t,
                      double F_t,
                      double keyFrequency) { ...; return score; }
}
```

  - LanguageModel for PonteCroftLanguageModel
- Found in package `uk.ac.gla.terrier.matching.models`

Term Score Modifiers

- Alters the given scores to a term in a document
  - FieldScoreModifier
  - TermInFieldModifier
  - RequiredTermModifier
- Specify the term score modifiers to apply with the following property:
  `matching.tsms=FieldScoreModifier`
- The query operators `field:term`, `+term` result in applying the term score modifiers `TermInFieldModifier` and `RequiredTermModifier`
- Found in package `uk.ac.gla.terrier.matching.tsms`

Document Score Modifiers

- Alters the given scores to a document
  - PhraseScoreModifier
  - BooleanScoreModifier
  - BooleanFallback
- Specify the document score modifiers to apply with the following property:
  `matching.dsms=BooleanScoreModifier`
- The query "`t1 t2`" returns only documents that match the phrase
  - Applies the document score modifier `PhraseScoreModifier` as a filter
- Found in package `uk.ac.gla.terrier.matching.dsms`

ResultSet

- Matching returns the ResultSet
- The ResultSet contains
  - Array of scores
  - Array of document ids (numerical document identifiers)
  - Array of flags that denote whether a query term occurred in a document
- Found in package `uk.ac.gla.terrier.matching`

After Matching

- Post-processing
  - Alters the result set after matching has finished
  - e.g. Query expansion expands the query, then runs matching again with the new query
- Post-filtering
  - Optional filtering of documents
- Found in package `uk.ac.gla.terrier.querying`

Compiling Terrier

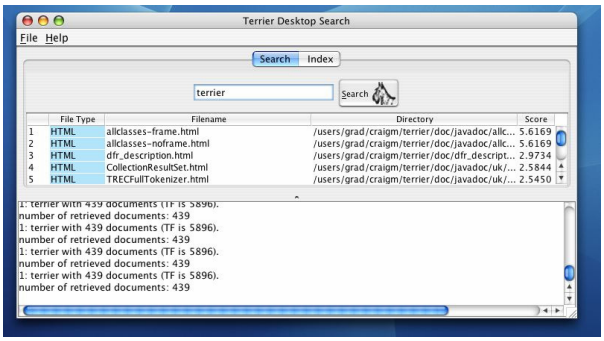
- To use your code with Terrier, add your jar file or your class folder to the CLASSPATH environment variable
- If you do need to alter the code in Terrier, then you have to recompile.
- Terrier is distributed with a script for compiling on Linux-like platforms:
  - Execute `make clean compile` to compile
  - No compiling script on windows

Outline

- What is Terrier?
- Using Terrier
  - Installing
  - Indexing
  - Configuring
  - Retrieving
  - Evaluation
- Understanding & extending Terrier
  - Indexing API
  - Querying API
  - Compiling
- ***Putting it altogether***

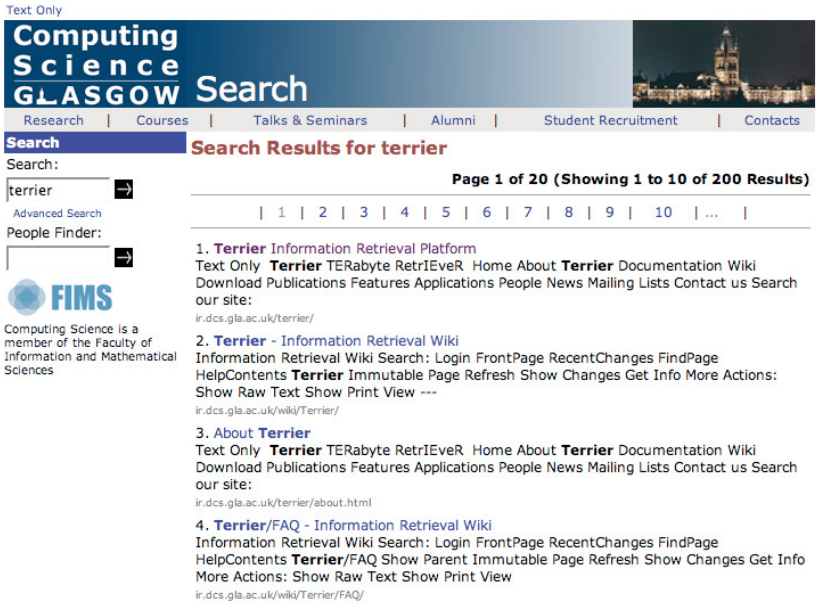
Putting it altogether (1)

- Searching your desktop:
  - Terrier Desktop Search
- Comes with Terrier
- Java Swing GUI
- SimpleFileCollection
  - FileDocument, PDFDocument, WordDocument, etc



Putting it altogether (2)

- Custom Search Engine Application
  - Terrier integrated with Web crawler
  - Indexes and retrieves Web pages from DCS
  - Results on Web interface



Summarising

- General:
  - Installing, trec_setup.sh, *terrier.properties*, trec_terrier.sh
- Indexing:
  - Stemming, Stopwords, Blocks, Fields
- Retrieving:
  - Topics path, Weighting model, Query Expansion
- Evaluation:
  - Qrels path
- Extending:
  - Indexing API, Data structures, Retrieval API

Useful links

- Terrier Website
  http://ir.dcs.gla.ac.uk/terrier/
- IR group & Terrier Wiki
  http://ir.dcs.gla.ac.uk/wiki/Terrier
  http://ir.dcs.gla.ac.uk/wiki
- Terrier Forum
  http://ir.dcs.gla.ac.uk/terrier/forum/
- Terrier Documentation
  - Contents http://ir.dcs.gla.ac.uk/terrier/doc/contents.html
  - Quickstart http://ir.dcs.gla.ac.uk/terrier/doc/quickstart.html
  - All properties http://ir.dcs.gla.ac.uk/terrier/doc/properties.html
- Terrier Research Publications
  http://ir.dcs.gla.ac.uk/terrier/publications.html