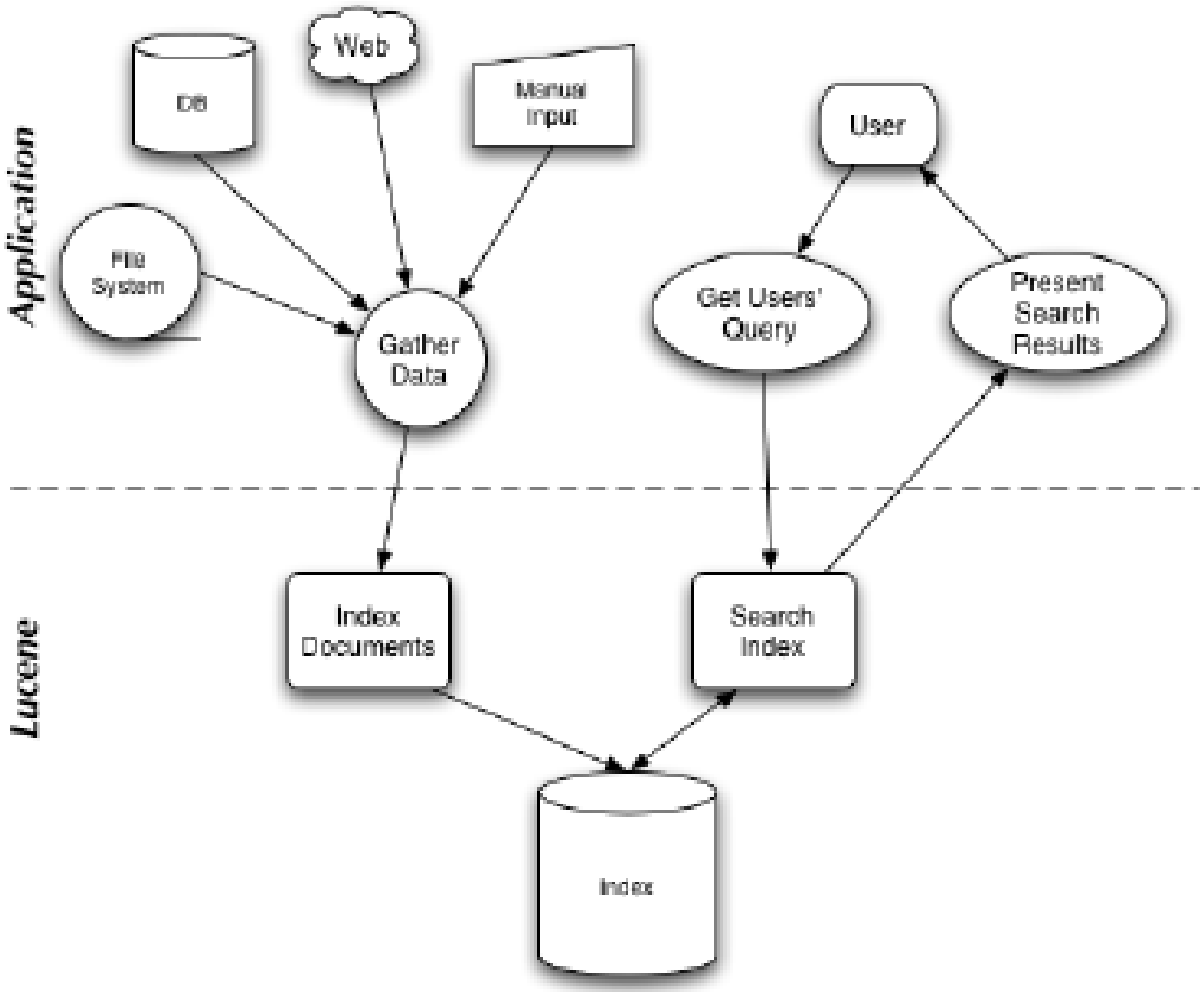
Εργασία στο μάθημα «Ανάκτηση Πληροφοριών

Ακ. Έτος: 2007-2008

Εισαγωγή – Lucene library

- Lucene είναι μία βιβλιοθήκη με υλοποιημένες βασικές λειτουργίες Ανάκτησης Πληροφοριών:
  - Οργάνωση κειμένων με τη δημιουργία ευρετήριου
  - αναζήτηση στα κείμενα με βάση το ευρετήριο
- Η βιβλιοθήκη είναι υλοποιημένη σε Java
- Είναι ελεύθερη προς χρήση
- Συμπεριλαμβάνεται στα έργα (projects) της Apache Jakarta (http://lucene.apache.org/)
- Η Lucene σήμερα χρησιμοποιείται ευρέως για την ανάπτυξη εφαρμογών που έχουν δυνατότητες ανάκτησης κειμένων
- Παράδειγμα ερωτήσεων:
  - +George +Rice -eat -pudding
  - Apple –pie +Tiger,
  - animal:monkey AND food:banana

Τυπική περίπτωση χρήσης της Lucene σε εφαρμογές



Δυνατότητα επεξεργασίας πολλών τύπων κειμένων

- Η Lucene μπορεί να «χτίσει» ευρετήριο και κατόπιν να αναζητήσει σε οποιαδήποτε δεδομένα που μπορούν να μετατραπούν σε απλό κείμενο.
- HTML και XML σελίδες, Word και PDF κείμενα: μετατροπή σε απλή μορφή κειμένου πριν την επεξεργασία από τη Lucene.
- Η βιβλιοθήκη προσφέρει φίλτρα που κάνουν αυτή την μετατροπή.

Παράδειγμα ανάπτυξης βασικής εφαρμογής αναζήτησης

- Δημιουργία ευρετηρίου

Listing 1.1 `Indexer`: traverses a file system and indexes .txt files

```
/**
 * This code was originally written for
 * Erik's Lucene intro java.net article
 */
public class Indexer {

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      throw new Exception("Usage: java " + Indexer.class.getName()
        + " <index dir> <data dir>");
    }
    File indexDir = new File(args[0]);
    File dataDir = new File(args[1]);

    long start = new Date().getTime();
    int numIndexed = index(indexDir, dataDir);
    long end = new Date().getTime();

    System.out.println("Indexing " + numIndexed + " files took "
      + (end - start) + " milliseconds");
  }

  // open an index and start file directory traversal
  public static int index(File indexDir, File dataDir)
    throws IOException {
```

Create Lucene index in this directory

Index files in this directory

Δημιουργία ευρετηρίου

```
    if (!dataDir.exists() || !dataDir.isDirectory()) {
      throw new IOException(dataDir
        + " does not exist or is not a directory");
    }

    IndexWriter writer = new IndexWriter(indexDir,
      new StandardAnalyzer(), true);
    writer.setUseCompoundFile(false);

    indexDirectory(writer, dataDir);

    int numIndexed = writer.docCount();
    writer.optimize();
    writer.close();
    return numIndexed;
  }
```

❶ Create Lucene index

Close index

```
// recursive method that calls itself when it finds a directory
private static void indexDirectory(IndexWriter writer, File dir)
  throws IOException {

  File[] files = dir.listFiles();

  for (int i = 0; i < files.length; i++) {
    File f = files[i];
    if (f.isDirectory()) {
      indexDirectory(writer, f);
    } else if (f.getName().endsWith(".txt")) {
      indexFile(writer, f);
    }
  }
}
```

❷ Recurse

Index .txt files only

Δημιουργία ευρετηρίου

```
  // method to actually index a file using Lucene
  private static void indexFile(IndexWriter writer, File f)
    throws IOException {

    if (f.isHidden() || !f.exists() || !f.canRead()) {
      return;
    }

    System.out.println("Indexing " + f.getCanonicalPath());

    Document doc = new Document();
    doc.add(Field.Text("contents", new FileReader(f)));
    doc.add(Field.Keyword("filename", f.getCanonicalPath()));
    writer.addDocument(doc);
  }
}
```

❸ Index file content

❹ Index filename

❺ Add document to Lucene index

Ένα παράδειγμα εκτέλεσης

```
% java lia.meetlucene.Indexer build/index/lucene

Indexing /lucene/build/test/TestDoc/test.txt
Indexing /lucene/build/test/TestDoc/test2.txt
Indexing /lucene/BUILD.txt
Indexing /lucene/CHANGES.txt
Indexing /lucene/LICENSE.txt
Indexing /lucene/README.txt
Indexing /lucene/src/jsp/README.txt
Indexing /lucene/src/test/org/apache/lucene/analysis/ru/
⇒  stemsUnicode.txt
Indexing /lucene/src/test/org/apache/lucene/analysis/ru/test1251.txt
Indexing /lucene/src/test/org/apache/lucene/analysis/ru/testKOI8.txt
Indexing /lucene/src/test/org/apache/lucene/analysis/ru/
⇒  testUnicode.txt
Indexing /lucene/src/test/org/apache/lucene/analysis/ru/
⇒  wordsUnicode.txt
Indexing /lucene/todo.txt
Indexing 13 files took 2205 milliseconds
```

Αναζήτηση ευρετηρίου

Listing 1.2 `Searcher`: searches a Lucene index for a query passed as an argument

```
/**
 * This code was originally written for
 * Erik's Lucene intro java.net article
 */
public class Searcher {

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      throw new Exception("Usage: java " + Searcher.class.getName()
        + " <index dir> <query>");
    }

    File indexDir = new File(args[0]);   <-- Index directory created by Indexer
    String q = args[1];   <-- Query string

    if (!indexDir.exists() || !indexDir.isDirectory()) {
      throw new Exception(indexDir +
        " does not exist or is not a directory.");
    }

    search(indexDir, q);
  }
```

Αναζήτηση ευρετηρίου

```
public static void search(File indexDir, String q)
  throws Exception {
  Directory fsDir = FSDirectory.getDirectory(indexDir, false);
  IndexSearcher is = new IndexSearcher(fsDir);          ❶ Open index

  Query query = QueryParser.parse(q, "contents",        ❷ Parse query
    new StandardAnalyzer());
  long start = new Date().getTime();
  Hits hits = is.search(query);                         ❸ Search index
  long end = new Date().getTime();

  System.err.println("Found " + hits.length() +          ◁ Write search stats
    " document(s) (in " + (end - start) +
    " milliseconds) that matched query '" +
      q + "':");

  for (int i = 0; i < hits.length(); i++) {
    Document doc = hits.doc(i);                         ❹ Retrieve matching document
    System.out.println(doc.get("filename"));            ◁ Display filename
  }
 }
}
```

Παράδειγμα εκτέλεσης

```
%java lia.meetlucene.Searcher build/index 'lucene'

Found 6 document(s) (in 66 milliseconds) that matched
⇒  query 'lucene':
/lucene/README.txt
/lucene/src/jsp/README.txt
/lucene/BUILD.txt
/lucene/todo.txt
/lucene/LICENSE.txt
/lucene/CHANGES.txt
```

Περιγραφή των κλάσεων για δημιουργία ευρετηρίου

- *IndexWriter:* κύρια κλάση για τη δημιουργία ευρετηρίου
- Η κλάση δημιουργεί ένα νέο ευρετήριο και προσθέτει κείμενα σε ένα ευρετήριο που ήδη υπάρχει
- Επίσης παρέχει δυνατότητα τροποποίησης τροποποίησης του ευρετηρίου
- Δεν δίνει όμως δυνατότητα ανάγνωσης ή αναζήτησης στο ευρετήριο

Περιγραφή των κλάσεων για δημιουργία ευρετηρίου

- *Directory:* Κλάση που αναπαριστά τη θέση του ευρετηρίου
- Δύο υποκλάσεις:
  - FSDirectory: Αποθήκευση στο δίσκο
  - RAMDirectory: Αποθήκευση στη κύρια μνήμη

Περιγραφή των κλάσεων για δημιουργία ευρετηρίου

- *Analyzer:* ανάλυση των κειμένων σε λέξεις. Επιλογή των όρων (λέξεων κλειδιών) που θα αποθηκευθούν στο ευρετήριο
- Αν το έγγραφο (document) δεν είναι απλό κείμενο, θα πρέπει μετατραπεί πρώτα σε αυτή τη μορφή
- Είναι abstract class με πολλές υλοποιήσεις:
  - παράληψη των *stop words*
  - Μετατροπή κεφαλαίων σε μικρά
  - Η επιλογή του Analyzer είναι κρίσιμο σημείο κατά την ανάπτυξη της εφαρμογής.

Περιγραφή των κλάσεων για δημιουργία ευρετηρίου

Primary analyzers available in Lucene

Analyzer	Steps taken
`WhitespaceAnalyzer`	Splits tokens at whitespace
`SimpleAnalyzer`	Divides text at nonletter characters and lowercases
`StopAnalyzer`	Divides text at nonletter characters, lowercases, and removes stop words
`StandardAnalyzer`	Tokenizes based on a sophisticated grammar that recognizes e-mail addresses, acronyms, Chinese-Japanese-Korean characters, alphanumerics, and more; lowercases; and removes stop words

Περιγραφή των κλάσεων για δημιουργία ευρετηρίου

- *Document:* αποτελείται από πεδία
- Είναι οτιδήποτε (π.χ. ιστοσελίδα, ηλεκτρονικό μήνυμα, αρχείο κειμένου) για το οποίο ενδιαφερόμαστε να ανακτηθεί αργότερα.
- Τα πεδία αντιπροσωπεύουν το αρχικό κείμενο ή μετα-δεδομένα που συσχετίζονται με το κείμενου
- Η αρχική προέλευση του κειμένου δεν έχει σημασία για την προσθήκη του κειμένου στο ευρετήριο
- Τα μετα-δεδομένα όπως συγγραφέας, τίτλος, θέμα, ημερομηνία τροποποίησης εισάγονται στο ευρετήριο αλλά αποθηκεύονται ξεχωριστά ως πεδία του κειμένου

Περιγραφή των κλάσεων για δημιουργία ευρετηρίου

- *Field*
- Κάθε κείμενο στο ευρετήριο περιέχει ένα ή περισσότερα πεδία.
- Field: η κλάση που περιγράφει ένα πεδίο
- Κάθε πεδίο αντιστοιχεί σε ένα τμήμα δεδομένων για το οποίο:
  - ερωτήσεις μπορούν να υποβληθούν
  - ή απλά να επιστραφεί ως αποτέλεσμα απάντησης σε ερώτημα

Περιγραφή των κλάσεων για δημιουργία ευρετηρίου

- Τύποι πεδίων
  - Keyword: Δεν αναλύεται αλλά εισάγεται στο ευρετήριο (μπορεί να αναζητηθεί) και αποθηκεύεται αυτούσιο
  - UnIndexed: Δεν αναλύεται, δεν εισάγεται στο ευρετήριο (δεν μπορεί να αναζητηθεί) αλλά αποθηκεύεται αυτούσιο. Είναι κατάλληλο για πεδία τα περιεχόμενα των οποίων προβάλλονται στα αποτελέσματα αναζήτησης
  - UnStored: Το αντίθετο του UnIndexed – Αναλύεται και εισάγεται στο ευρετήριο αλλά δεν αποθηκεύεται αυτούσιο. Είναι κατάλληλο για κείμενα μεγάλου μεγέθους τα οποία δεν χρειάζεται να ανακτηθούν στην αρχική τους μορφή.
  - Text: Αναλύεται και εισάγεται στο ευρετήριο. Εάν τα δεδομένα είναι τύπου String αποθηκεύονται ενώ αν προέρχονται από ένα Reader δεν αποθηκεύονται

Περιγραφή των κλάσεων για δημιουργία ευρετηρίου

Table 1.2 An overview of different field types, their characteristics, and their usage

Field method/type	Analyzed	Indexed	Stored	Example usage
Field.Keyword(String, String) Field.Keyword(String, Date)		✔	✔	Telephone and Social Security numbers, URLs, personal names Dates
Field.UnIndexed(String, String)			✔	Document type (PDF, HTML, and so on), if not used as a search criteria

Table 1.2 An overview of different field types, their characteristics, and their usage *(continued)*

Field method/type	Analyzed	Indexed	Stored	Example usage
Field.UnStored(String, String)	✔	✔		Document titles and content
Field.Text(String, String)	✔	✔	✔	Document titles and content
Field.Text(String, Reader)	✔	✔		Document titles and content

Περιγραφή των κλάσεων για αναζήτηση ευρετηρίου

- *IndexSearcher:* κεντρική κλάση για την αναζήτηση του ευρετηρίου που έχει δημιουργηθεί από την IndexWriter
- «Ανοίγει» το ευρετήριο για ανάγνωση μόνο
- Προσφέρει πολλές μεθόδους αναζήτησης
- Η απλούστερη μέθοδος λαμβάνει σαν είσοδο ένα αντικείμενο Query και επιστρέφει ένα αντικείμενο Hits
- Ένα παράδειγμα:

- **IndexSearcher is = new IndexSearcher(FSDirectory.getDirectory("/tmp/index", false));**
- Query q = new TermQuery(new Term("contents", "lucene"));
- Hits hits = **is.search**(q);

Περιγραφή των κλάσεων για αναζήτηση ευρετηρίου

- *Term: το* βασικό στοιχείο αναζήτησης
- Αποτελείται από ένα ζεύγος συμβολοσειρών: το όνομα του πεδίου και η τιμή του πεδίου.
- TermQuery: ο πιο απλός τύπος ερώτησης
- Query q = new TermQuery(**new Term("contents", "lucene")**);
- Hits hits = is.search(q);

Περιγραφή των κλάσεων για αναζήτηση ευρετηρίου

- *Query:* αφηρημένη κλάση με πολλές υποκλάσεις για διαφορετικούς τύπους ερωτήσεων
  - TermQuery
  - BooleanQuery: Συνδυάζει με λογικούς τελεστές τους υπόλοιπους τύπους ερωτήσεων
  - PhraseQuery: Επιστρέφει κείμενα που περιέχουν μία συγκεκριμένη πρόταση π.χ. “New York”

    Με τη μέθοδο **setSlop**(int s) μπορούμε να επιστρέψουμε κείμενα στα οποία οι όροι της φράσης απέχουν το πολύ s όρους
  - PrefixQuery: Επιστρέφει κείμενα τα οποία περιέχουν όρους που αρχίζουν με ένα συγκεκριμένο πρόθεμα
  - PhrasePrefixQuery: Ένα παράδειγμα στην επόμενη διαφάνεια
  - RangeQuery: Επιστρέφει όλα τα κείμενα που περιέχουν όρους που αλφαβητικά είναι μεταξύ δύο συγκεκριμένων όρων
  - FilteredQuery: Εφαρμόζει ένα φίλτρο πάνω στα αποτελέσματα μίας άλλης ερώτησης,
  - SpanQuery: Μπορούμε να διατυπώσουμε ερωτήσεις όπως "quick fox" is near "lazy dog"

Παράδειγμα: PhrasePrefixQuery

```
public void testBasic() throws Exception {
    PhrasePrefixQuery query = new PhrasePrefixQuery();
    query.add(new Term[] {
    new Term("field", "quick"),
    new Term("field", "fast")
    });
    query.add(new Term("field", "fox"));
    /* Any of these terms may be
       in first position to match
       only one in
    second position */
    Hits hits = searcher.search(query);
    assertEquals("fast fox match", 1, hits.length());
    query.setSlop(1);
    hits = searcher.search(query);
    assertEquals("both match", 2, hits.length());
    }
```

Περιγραφή των κλάσεων για αναζήτηση ευρετηρίου

- *Hits:* περιέχει τα αποτελέσματα της αναζήτησης ταξινομημένα κατά σειρά σχετικότητας
- Για λόγους απόδοσης, η κλάση δεν περιλαμβάνει όλα τα κείμενα που ανακτήθηκαν από την ερώτηση αλλά κάθε φορά ένα μικρό ποσοστό.

Περιγραφή των κλάσεων για αναζήτηση ευρετηρίου

- *QueryParser:* μετατρέπει μία ερώτηση που έχει υποβληθεί από το χρήστη σε ένα αντικείμενο Query
- Π.χ.
  - **Query query = QueryParser.parse("+JUNIT +ANT -MOCK", "contents", new SimpleAnalyzer());**
  - Hits hits = searcher.search(query);
  - **query = QueryParser.parse("mock OR junit", "contents", new SimpleAnalyzer());**
  - hits = searcher.search(query);

Περιγραφή των κλάσεων για αναζήτηση ευρετηρίου

Expression examples that `QueryParser` handles

Query expression	Matches documents that...
`java`	Contain the term *java* in the default field
`java junit` `java or junit`	Contain the term *java* or *junit*, or both, in the default field[a]
`+java +junit` `java AND junit`	Contain both *java* and *junit* in the default field
`title:ant`	Contain the term *ant* in the `title` field

`title:extreme`
`-subject:sports`
`title:extreme`
`AND NOT subject:sports`
	Have *extreme* in the `title` field and don't have *sports* in the `subject` field

`(agile OR extreme) AND` `methodology`	Contain *methocology* and must also contain *agile* and/or *extreme*, all in the default field
`title:"junit in action"`	Contain the exact phrase "*junit in action*" in the `title` field
`title:"junit action"~5`	Contain the terms *junit* and *action* within five positions of one another
`java*`	Contain terms that begin with *java*, like *javaspaces*, *javaserver*, and *java.net*
`java~`	Contain terms that are close *to* the word *java*, such as *lava*
`lastmodified:` `[1/1/04 TO 12/31/04]`	Have `lastmodified` field values between the dates January 1, 2004 and December 31, 2004

[a] *The default operator is OR.*

Συνάρτηση Ομοιότητας στη Lucene

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left(\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot t.\text{getBoost}() \cdot \text{norm}(t,d) \right)$$

- **tf(t in d):** το πλήθος των εμφανίσεων του όρου *t* στο κείμενο *d*.

$$\text{tf}(t \text{ in } d) = \text{frequency}^{1/2}$$

- **idf(t) =** 1 + log (numDocs /(docFreq+1)) όπου numDocs είναι ο συνολικός αριθμός κειμένων και docFreq το πλήθος των κειμένων στα οποία εμφανίζεται ο όρος *t*
- ***coord(q,d)*:** ένα κείμενο που περιέχει πολλούς όρους της ερώτησης q θα έχει υψηλότερη τιμή coord σε σχέση με άλλο κείμενο με λιγότερους όρους από την ερώτηση

Συνάρτηση Ομοιότητας στη Lucene

$$queryNorm(q) = queryNorm(sumOfSquaredWeights) = \frac{1}{sumOfSquaredWeights^{1/2}}$$

$$sumOfSquaredWeights = q.getBoost()^2 \cdot \sum_{t\ in\ q} \left(idf(t) \cdot t.getBoost() \right)^2$$

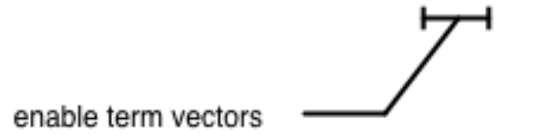
$$norm(t,d) = doc.getBoost() \cdot lengthNorm(field) \cdot \prod_{field\ f\ in\ d\ named\ as\ t} f.getBoost()$$

lengthNorm = 1 / (numTerms in field)^0.5

Term vectors

- Συλλογή (Collection) των όρων και των συχνοτήτων εμφάνισης τους σε ένα πεδίο κειμένου.
- Δεν αποθηκεύονται αυτόματα τα διανύσματα αυτά όταν εισάγουμε ένα νέο κείμενο στο ευρετήριο
- Γίνεται ως εξής:

```
doc.add(Field.UnStored("subject", subject, true));
```



- Για να να ανακτήσουμε το διάνυσμα των όρων ενός πεδίου, εκτελούμε το εξής:
- TermFreqVector termFreqVector =

  reader.getTermFreqVector(id, "subject");

  όπου reader είναι ένας IndexReader

Εργασία – 1ο μέρος (60%)

- Σύστημα ανάκτησης πληροφοριών βασισμένο στη βιβλιοθήκη Lucene
- Διατύπωση των ερωτημάτων μέσα από γραφικό περιβάλλον
- Προβολή των κειμένων που ανακτήθηκαν κατά σειρά σχετικότητας
- Θα πρέπει επίσης να προβάλλεται και το περιεχόμενο του αρχείου αν ο χρήστης το επιλέξει στη λίστα των αποτελεσμάτων
- Εκτίμηση της απόδοσης του συστήματος:
  - χρήση συγκεκριμένης συλλογής κειμένων - CACM (αρχείο cacm.tar.gz): συνόψεις άρθρων δημοσιευμένα στο περιοδικό Communications of the ACM
  - Διαγράμματα Precision – Recall, R-precision, Average Precision at Seen Relevant Documents
  - Η παραγωγή των διαγραμμάτων απόδοσης του συστήματος μπορεί να γίνει είτε από το ίδιο σύστημα ή εναλλακτικά με τη βοήθεια κάποιου εξωτερικού προγράμματος (π.χ. Excel).

Συλλογή CACM

3204 ερευνητικά άρθρα σχετικά με την πληροφορική

64 ερωτήσεις

Έχουν επίσης καθορίσει τα σχετικά κείμενα για κάθε ερώτηση.

Παράδειγμα κειμένου:

.I 2195
.T
On the Optimal Detection of Curves in Noisy Pictures
.W
A technique for recognizing systems of lines
is presented. In this technique the heuristic
of the problem is not embedded in the recognition algorithm
but is expressed in a figure of merit.
A multistage decision process is then able to recognize
in the input picture the optimal system of lines
according to the given figure of merit. Due to the
global approach, greater flexibility and adequacy
in the particular problem is achieved. The relation
between the structure of the figure of merit and
the complexity of the optimization process is then discussed.
The method described is suitable for parallel
processing because the operations relative to each
state can be computed in parallel, and the number
of stages is equal to the length N of the curves (or
to log2 N if the approximate method is used).
.B
CACM May, 1971
.A
Montanari, U.
.K
picture processing, picture recognition, picture
description, curve detection, line detection,
edge detection,optimal detection, heuristic methods,
global recognition, parallel processing, dynamic
programming, interaction graph, secondary optimization problem
.C
3.63 3.66 5.42
.N
CA710504 JB February 3, 1978 2:49 PM
.X
1663 4 2195
2195 4 2195
2679 4 2195
1190 5 2195
2195 5 2195
2195 5 2195
2195 5 2195
2883 5 2195
2195 6 2195

Συλλογή CACM

Μία ενδεικτική ερώτηση (query.text):

I 43

W

Analysis and perception of shape by humans and computers. Shape descriptions, shape recognition by computer. Two-dimensional shapes. Measures of circularity. Shape matching.

.N

43. George R. Cross

Σχετικά κείμενα (qrels.text):

43 0122 0 0
43 0266 0 0
43 0297 0 0
43 0462 0 0
43 1113 0 0
43 1325 0 0
43 1528 0 0
43 1554 0 0

. . .

Λεπτομέρειες Υλοποίησης

- Καθορισμός των λέξεων-κλειδιών:
  - Θα βασιστείτε αποκλειστικά στον τίτλο, τα ονόματα των συγγραφέων καθώς και στις συνόψεις/αποσπάσματα των κειμένων αγνοώντας όλα τα υπόλοιπα πεδία
  - Θα αγνοήσετε κοινές λέξεις (συνδέσμους, αντωνυμίες, άρθρα κτλ.) που περιέχονται στο αρχείο common_words.txt
  - Χρήση του αλγόριθμου Porter για stemming

Εργασία – 2ο μέρος (25%)

- Αυτόματη βελτίωση /επαναδιατύπωση του αρχικού ερωτήματος:
  - Επιλέγονται τα πρώτα κ κείμενα από την κατάταξη των αποτελεσμάτων.
  - Για κάθε ένα από τα κ κείμενα, επιλέγονται οι λ πιο συχνοί όροι στο κείμενο
  - Στην αρχική ερώτηση προστίθενται οι παραπάνω όροι
  - Αν κάποιοι από τους πρόσθετους όρους υπάρχουν ήδη στο αρχικό ερώτημα αγνοούνται.
  - Η ερώτηση υποβάλλεται εκ νέου
  - Οι παράμετροι κ και λ θα μπορούν να αλλάζουν από το χρήστη δυναμικά

Εργασία – 3ο μέρος (15%)

- Βελτίωση /επαναδιατύπωση του αρχικού ερωτήματος:
  - Ανάδραση από το χρήστη: ο χρήστης καθορίζει ποια κείμενα είναι σχετικά και ποια άσχετα από αυτά που ανακτήθηκαν
  - Υπολογισμός των νέων βαρών των όρων της ερώτησης σύμφωνα με το τύπο Standard_Rochio:

$$\vec{q}_{new} = \alpha \vec{q}_{old} + \frac{\beta}{|D_r|} \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j \quad \text{Standard\_Rochio}$$

  - Η βιβλιοθήκη της Lucene δεν παρέχει απευθείας τρόπο για την υλοποίηση του παραπάνω τύπου.
  - Προσπαθήστε να προσεγγίσετε τη «λογική» του παραπάνω τύπου όσο πιστά μπορείτε με τη χρήση μεθόδων που παρέχονται από τη Lucene
  - Οι συντελεστές α, β, γ: παράμετροι λειτουργίας του συστήματος, μπορούν να μεταβάλλονται κατά τη λειτουργία του συστήματος

Παραδοτέα

- Περιγραφή του συστήματος
  - Έμφαση στις κλάσεις και τις μεθόδους της Lucene που χρησιμοποιήσατε
- Κώδικας με την κατάλληλη τεκμηρίωση
- Εκτελέσιμο πρόγραμμα και οδηγίες εγκατάστασης
- Όλα τα αρχεία τα σχετικά με τις εκτιμήσεις της απόδοσης του συστήματος (διαγράμματα precision-recall, R-precision κτλ.)
- Ημερομηνία παράδοσης: ημερομηνία εξέτασης του μαθήματος
- Η εργασία μπορεί να παραδοθεί από ομάδες αυστηρώς μέχρι δύο ατόμων.
- Η εργασία είναι υποχρεωτική
- Βαθμολογείται με άριστα το 4 και ο βαθμός προστίθεται στο βαθμό της γραπτής εξέτασης. Το άριστα στην γραπτή εξέταση είναι το 7.