Συστήματα Ανάκτησης Πληροφοριών

1ο ΠΑΡΑΔΟΤΕΟ

ΜΑΥ 13, 2024

ΜΠΙΛΙΑΣ ΓΕΩΡΓΙΟΣ 3200278 ΙΩΑΝΝΗΣ ΠΑΤΟΥΧΑΣ 3200149

ΠΕΡΙΕΧΟΜΕΝΑ

1.

Προεπεξεργαστείτε τη συλλογή κειμένων IR2024 (αρχείο documents.txt) προκειμένου να είναι σε κατάλληλη μορφή για να χρησιμοποιηθεί από τη μηχανή αναζήτησης Lucene.

FirstPhase.java

Αρχικά, δημιουργούμε το αρχείο **FirstPhase.java** με το οποίο θα επεξεργαστούμε το **documents.txt**.

Πρώτα καλούμε την **TXTParsing** ώστε να κάνει Parse τα documents του **documents.txt**.

```
List<MyDoc> docs = TXTParsing.parse("Phase 1\\src\\main\\documents.txt"); //parse the documents
```

TXTParsing.java

Η **TXTParsing.java** διαβάζει ολόκληρο το αρχείο με την συνάρτηση της **IO.java**, **ReadEntireFileIntoAString**. (παραλλαγή αυτής που έχει δοθεί στο φροντιστήριο ώστε να διαβάζει όλο το αρχείο)

```
  // Parse txt file
            String txt_file = IO.ReadEntireFileIntoAString(file); // Read the entire file into a string
```

Στην συνέχεια, χωρίζει το αρχείο σε **docs** με την χρήση της split.

```
 String[] docs = txt_file.split("///\\s*\\n+"); // Split the string into documents
```

Και εκτυπώνει τον αριθμό των διαβασμένων docs.

```
System.out.println("Read: " + docs.length + " docs"); // Print the number of documents
```

Συμπληρωματικά, δημιουργεί ένα **ArrayList parsed_docs** που θα περιέχει τα **docs** στην κατάλληλη μορφή.

```
// Parse each document from the txt file

            List<MyDoc> parsed_docs = new ArrayList<MyDoc>(); // Create a list to store the parsed documents
```

Αυτή η διαδικασία πραγματοποιείται μέσα σε ένα **for loop για κάθε document** όπου το splitάρει με τον κατάλληλο τρόπο.

```
 for (String doc : docs) { // For each document
                String[] parts = doc.split("\n", 2); // Split document into ID,
title, and content

                    // Get the ID, title, and content

                    String id = parts[0].trim(); // Get the ID
                    String parts2[] = parts[1].split(":", 2); // Split the rest
of the document into title and content
                    String title = parts2[0].trim(); // Get the title
                    String content = parts2[1].trim(); // Get the content

                    parsed_docs.add(new MyDoc(id, title, content)); // Add the
parsed document to the list
```

Τέλος, εκτυπώνει τον αριθμό των parsed docs και τα επιστρέφει πίσω στην **FirstPhase** οπου αποθηκεύονται στην **List 'docs'**.

```
            System.out.println(parsed_docs.size() + " docs parsed"); // Print the
number of parsed documents

            return parsed_docs; // Return the parsed documents
```

2.

Δημιουργήστε ένα ευρετήριο από τη συλλογή χρησιμοποιώντας τη μηχανή αναζήτησης Lucene. Επιλέξτε κατάλληλο Analyzer και συνάρτηση ομοιότητας την ClassicSimilarity. Κάθε κείμενο θα πρέπει να αποθηκευτεί σε ένα field της Lucene.

Στην συνέχεια, ανοίγουμε το directory για τον index writer, δηλώνουμε τον **EnglishAnalyser** (Analyser που επιλέξαμε), την **ClassicSimilarity** και κάνουμε configure τον **IndexWriter**.

```
 String indexLocation = ("Phase 1\\src\\main\\index"); //define were to store the
index

            Directory dir = FSDirectory.open(Paths.get(indexLocation)); //open
the directory
            Analyzer analyzer = new EnglishAnalyzer(); //define Analyzer

            // define which similarity to use
            Similarity similarity = new ClassicSimilarity();
```

```
            // configure IndexWriter
            IndexWriterConfig iwc = new IndexWriterConfig(analyzer);
            iwc.setSimilarity(similarity); // set the similarity

            // Create a new index in the directory, removing any
            // previously indexed documents:
            iwc.setOpenMode(OpenMode.CREATE);

            // create the IndexWriter with the configuration as above
            IndexWriter indexWriter = new IndexWriter(dir, iwc);
```

Έπειτα καλούμε την **IndexDoc** για όλα τα **docs**.

IndexDoc Method

Η **IndexDoc** δημιουργεί ένα document file στο οποίο με την χρήση της **MyDoc** αποθηκεύει μέσα του τα **id, Title και Content** ως **StoreField**.

```
// make a new, empty document
            Document doc = new Document();

            // create the fields of the document and add them to the document
            StoredField id = new StoredField("id", mydoc.getID());
            doc.add(id);
            StoredField title = new StoredField("title", mydoc.getTitle());
            doc.add(title);
            StoredField content = new StoredField("content", mydoc.getContent());
            doc.add(content);
```

Τέλος, πηγαίνει και βάζει όλα αυτά σε **TextField** named **contents**, το βάζει στο **doc** και το **doc** μέσα στον **IndexWriter**.

```
String fullSearchableText = mydoc.getID() + " " + mydoc.getTitle() + " " +
mydoc.getContent();
            TextField contents = new TextField("contents", fullSearchableText,
Field.Store.NO);
            doc.add(contents);

            if (indexWriter.getConfig().getOpenMode() == OpenMode.CREATE) {
                // New index, so we just add the document (no old document can be
there):
                // System.out.println("adding " + mydoc);
                indexWriter.addDocument(doc);
            }
```

Η **MyDoc.java είναι μια μικρή παραλλαγή απ' αυτή που μας έχει δοθεί στο φροντιστήριο, έχουν αλλαχθεί τα ονόματα των μεταβλητών**

3.

Εκτελέστε τα ερωτήματα (αρχείο queries.txt) πάνω στο ευρετήριο και συλλέξτε τις απαντήσεις της μηχανής, τα k πρώτα ανακτηθέντα κείμενα, για k = 50.

Αρχικά, δηλώνουμε το location του index, το field των docs που θα χρησιμοποιήσουμε και κάνουμε initialize τους **IndexReader,IndexWriter** και το **Similarity** που θα χρησιμοποιήσουμε.

```
String indexLocation = ("Phase 1\\src\\main\\index");
            String field = "contents";

            IndexReader indexReader =
DirectoryReader.open(FSDirectory.open(Paths.get(indexLocation)));
            IndexSearcher indexSearcher = new IndexSearcher(indexReader);
            indexSearcher.setSimilarity(new ClassicSimilarity());
```

Τέλος, δημιουργούμε τον **FileWriter** μας όπου θα αποθηκεύσουμε τα αποτελέσματα μας και καλούμε την μέθοδο **readQueriesFromFile**.

```
// Create a FileWriter to write results to a file
            FileWriter writer = new FileWriter("Phase
1\\src\\main\\trec_eval\\results.txt");

            // Read queries from the file and write results to the file
            readQueriesFromFile(indexSearcher, field, writer);
```

Αφού τελειώσει η διαδικασία κλείνουν ο writer και ο **IndexReader** και το πρόγραμμα τερματίζει.

readQueriesFromFile

Αρχικά, ανοίγουμε ένα αρχείο κειμένου ("**queries.txt**") για ανάγνωση γραμμή προς γραμμή και το χωρίζουμε σε κομμάτια οπου υπάρχει «///».

```
 BufferedReader reader = new BufferedReader(new FileReader("Phase
1\\src\\main\\queries.txt"));
            String line;
```

```
            int queryNumber = 1;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split("///");
```

Αν η λίστα με τα κομμάτια δεν είναι κενή τότε αν η γραμμή αρχίζει με "Q" και έχει μήκος μέχρι τρία χαρακτήρες, θεωρείται αναγνωριστικό ερωτήματος και παραλείπεται.

```
if (queryText.startsWith("Q") && queryText.length() <= 3) {
                        System.out.println("Found Q: " + queryText);
                        continue; // Skip lines containing only query identifier
                    }
```

Αν δεν είναι κενή τότε κάνουμε initialize τους **analyser** και **QueryParser** μας, τρέχουμε το query και αποθηκεύουμε τα 50 καλύτερα αποτελέσματα. Κρατάμε το score για το **results.txt** και τον αριθμό των hits ώστε αν υπάρχουν να φτιάξουμε το αρχείο, αλλιώς το παραλείπουμε.

```
if (!queryText.isEmpty()) {
                        // Process the query
                        System.out.println("Processing query " + queryNumber + ":
" + queryText);

                        Analyzer analyzer = new EnglishAnalyzer();
                        QueryParser parser = new QueryParser(field, analyzer);
                        System.out.println("Parsing query: " + queryText);
                        Query query = parser.parse(queryText);
                        TopDocs results = indexSearcher.search(query, 50);
                        ScoreDoc[] hits = results.scoreDocs;
                        long numTotalHits = results.totalHits;


                        if (numTotalHits == 0) {
                            System.out.println("No results found for query " +
queryNumber + ": " + queryText);
                        } else {
                            for (int i = 0; i < hits.length; i++) {
                                Document hitDoc = indexSearcher.doc(hits[i].doc);
                                writer.write("Q"+String.format("%02d",
queryNumber)+"\t"+ "0\t"+hitDoc.get("id") + "\t0\t" + hits[i].score +
"\tmyIRmethod"+ "\n");
                            }
                        }
```

Τελος, κλείνουμε τον reader μας και το πρόγραμμα τερματίζει.

```
    reader.close();
```

ΕΚΤΈΛΕΣΗ ΤΟΥ ΚΏΔΙΚΑ ΚΑΙ ΠΑΡΑΓΩΓΉ ΑΠΟΤΕΛΕΣΜΆΤΩΝ

4.

Αξιολογήστε τις απαντήσεις σας συγκρίνοντάς τες με τις σωστές απαντήσεις (αρχείο qrels.txt) χρησιμοποιώντας το εργαλείο αξιολόγησης trec_eval και τα μέτρα αξιολόγησης MAP (mean average precision) και Precision@k (ακρίβεια στα k πρώτα ανακτηθέντα

κείμενα) για k = 5, 10, 15, 20.

Αρχικά, βάζουμε το **documents.txt** μέσα στο path του εκτελέσιμου κώδικα.

Για την συγκεκριμένη διαδικασία εκτελούμε το **FirstPhase.java** που δημιουργεί το **index** folder και το **Searcher.java** που μας δίνει ένα txt με τα αποτελέσματα (**results.txt**). Στην συνέχεια, βάζουμε το **qrels.txt** που μας έχει δοθεί στο ίδιο directory με τα δικά μας results καθώς και τα **trec_eval.exe** και **cygwin1.dll**. Τέλος, τρέχουμε την παρακάτω εντολή στο cmd:

```
trec_eval -m official -m map -m P.5,10,15,20 qrels.txt results.txt > eval.txt
```

όπου το συγκεκριμένο command τρέχει το trec_eval.exe με τις μετρικές

official: δίνει κάποιες έξτρα μετρικές

map: δίνει την mean average precision που μας ζητείται

P.5,10,15,20: ακρίβεια στα k πρώτα ανακτηθέντα κείμενα για k = 5, 10, 15, 20

Και αποθηκεύει τα αποτελέσματα σε ένα αρχείο eval.txt.

ΣΥΓΚΡΙΣΗ ANALYZERS

Αφού τρέξαμε το πρόγραμμα με διάφορους Analyzers και συγκρίναμε τα αποτελέσματα που φαίνονται και παρακάτω, καταλήξαμε πως στην προκειμένη περίπτωση ο EnglishAnalyzer έχει την καλύτερη απόδοση. Έτσι, επικράτησε ένταντι των άλλων.

Αποτελέσματα

EnglishAnalyzer

```
runid                   all myIRmethod
num_q                   all 10
num_ret                 all 500
num_rel                 all 152
num_rel_ret             all 95
map                     all 0.3735
gm_map                  all 0.2585
Rprec                   all 0.4019
bpref                   all 0.6396
recip_rank              all 0.7254
iprec_at_recall_0.00    all 0.8079
iprec_at_recall_0.10    all 0.6621
iprec_at_recall_0.20    all 0.6142
iprec_at_recall_0.30    all 0.5464
iprec_at_recall_0.40    all 0.4721
iprec_at_recall_0.50    all 0.3507
iprec_at_recall_0.60    all 0.3135
iprec_at_recall_0.70    all 0.2410
iprec_at_recall_0.80    all 0.1806
iprec_at_recall_0.90    all 0.1090
iprec_at_recall_1.00    all 0.0718
P_5                     all 0.5200
P_10                    all 0.4600
P_15                    all 0.4000
P_20                    all 0.3350
```

SimpleAnalyzer

```
runid                   all myIRmethod
num_q                   all 10
num_ret                 all 500
num_rel                 all 152
num_rel_ret             all 83
map                     all 0.3054
gm_map                  all 0.2443
Rprec                   all 0.3730
bpref                   all 0.5577
recip_rank              all 0.7667
iprec_at_recall_0.00    all 0.8326
iprec_at_recall_0.10    all 0.6966
iprec_at_recall_0.20    all 0.5915
iprec_at_recall_0.30    all 0.4795
iprec_at_recall_0.40    all 0.4164
iprec_at_recall_0.50    all 0.3068
iprec_at_recall_0.60    all 0.2178
iprec_at_recall_0.70    all 0.0862
iprec_at_recall_0.80    all 0.0771
iprec_at_recall_0.90    all 0.0000
iprec_at_recall_1.00    all 0.0000
P_5                     all 0.4800
P_10                    all 0.4700
P_15                    all 0.3667
P_20                    all 0.2900
```

StandardAnalyzer

```
runid                   all myIRmethod
num_q                   all 10
num_ret                 all 500
num_rel                 all 152
num_rel_ret             all 85
map                     all 0.3089
gm_map                  all 0.2450
Rprec                   all 0.3667
bpref                   all 0.5692
recip_rank              all 0.7500
iprec_at_recall_0.00    all 0.8028
iprec_at_recall_0.10    all 0.6569
iprec_at_recall_0.20    all 0.5938
iprec_at_recall_0.30    all 0.5082
iprec_at_recall_0.40    all 0.4202
iprec_at_recall_0.50    all 0.2971
iprec_at_recall_0.60    all 0.2200
iprec_at_recall_0.70    all 0.0976
iprec_at_recall_0.80    all 0.0839
iprec_at_recall_0.90    all 0.0326
iprec_at_recall_1.00    all 0.0000
P_5                     all 0.4800
P_10                    all 0.4400
P_15                    all 0.3600
P_20                    all 0.2950
```

WhiteSpaceAnalyzer

```
runid                   all myIRmethod
num_q                   all 10
num_ret                 all 500
num_rel                 all 152
num_rel_ret             all 63
map                     all 0.2337
gm_map                  all 0.1622
Rprec                   all 0.2789
bpref                   all 0.4128
recip_rank              all 0.7933
iprec_at_recall_0.00    all 0.8390
iprec_at_recall_0.10    all 0.7307
iprec_at_recall_0.20    all 0.3828
iprec_at_recall_0.30    all 0.3042
iprec_at_recall_0.40    all 0.2018
iprec_at_recall_0.50    all 0.1865
iprec_at_recall_0.60    all 0.1807
iprec_at_recall_0.70    all 0.0722
iprec_at_recall_0.80    all 0.0419
iprec_at_recall_0.90    all 0.0300
iprec_at_recall_1.00    all 0.0000
P_5                     all 0.4600
P_10                    all 0.3500
P_15                    all 0.2667
P_20                    all 0.2400
```

ΠΗΓΕΣ

- Διαφάνειες Φροντιστηρίων E-CLASS
- Learn how to use trec_eval to evaluate your information retrieval system - Rafael Glater