



ΑΝΑΦΟΡΑ PROJECT

Συστήματα Ανάκτησης Πληροφοριών

ANASTASIOS ZACHARIOUDAKIS
(3170048) – NIKOLAOS VATTIS (3170203)

Φάση 2: LSI (Συλλογή LISA)

ΒΗΜΑ 1

Για να δημιουργήσουμε το **txd** φτιάχνουμε το ευρετήριο μας από την αρχή και βάζουμε κάθε **document** με την μέθοδο **addDocWithTermVector** στην οποία φτιάχνουμε **field** από τα **document** και έχουμε ορίσει τα **index options** στην **fieldType type**.

```
private static void addDocWithTermVector(IndexWriter indexWriter, MyDoc mydoc, FieldType type) throws
    Document doc = new Document();
    //TextField title = new TextField("title", value, Field.Store.YES);

    // create the fields of the document and add them to the document
    String value = mydoc.getTitle() + " " + mydoc.getText();

    Field field = new Field("contents", value, type);

    doc.add(field); //this field has term Vector enabled.
    indexWriter.addDocument(doc);
}
```

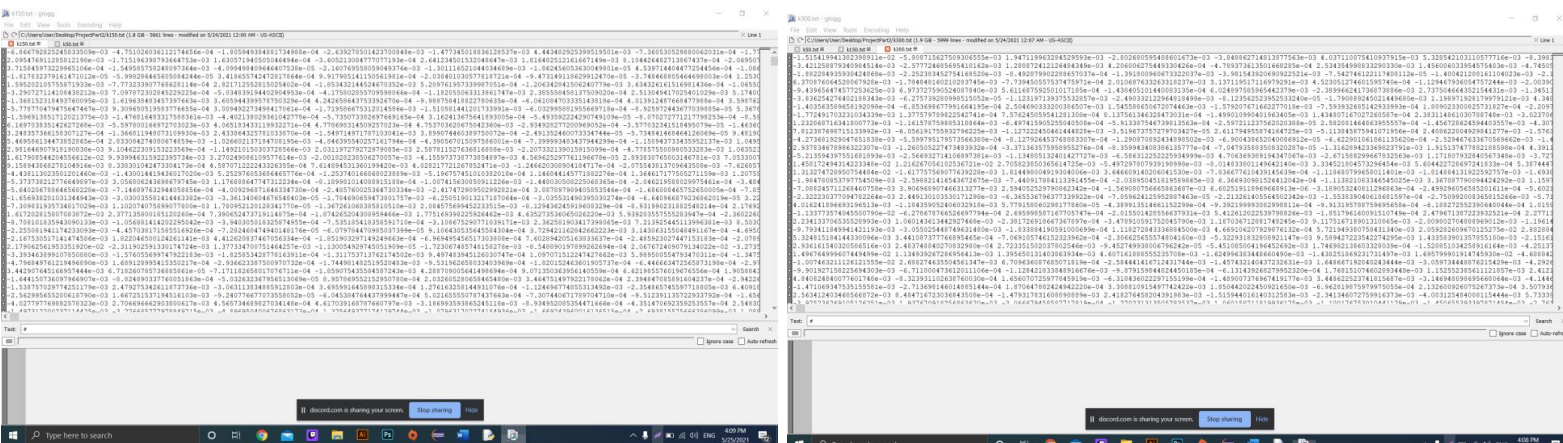
Έπειτα, αφού έχει φτιαχτεί το ευρετήριο μας καλούμε την μέθοδο **testSparseFreqDoubleArrayConversion** έτσι ώστε να γράψουμε σε ένα καινούργιο αρχείο τα αποτελέσματα των **terms** για κάθε **document**.

```
private static void testSparseFreqDoubleArrayConversion(IndexReader reader) throws Exception {
    //File svd = new File("C:\\Users\\User\\Desktop\\ProjectPart2\\svd.txt");
    //FileWriter writer = new FileWriter(svd,true);
    //BufferedWriter bufferwriter = new BufferedWriter(writer);
    Terms fieldTerms = MultiFields.getTerms(reader, "contents"); //the number of terms in the lexicon after analysis of the file
    System.out.println("Terms:" + fieldTerms.size());
    TermsEnum it = fieldTerms.iterator(); //iterates through the terms of the lexicon
    while(it.next() != null) {
        System.out.print(it.term().utf8ToString() + " "); //prints the terms
    }
    System.out.println();
    System.out.println();
    if (fieldTerms != null && fieldTerms.size() != -1) {
        IndexSearcher indexSearcher = new IndexSearcher(reader);
        for (ScoreDoc scoreDoc : indexSearcher.search(new MatchAllDocsQuery(), Integer.MAX_VALUE).scoreDocs) { //retrieves all
            System.out.println("DocID: " + scoreDoc.doc);
            //bufferwriter.write("DocID: " + scoreDoc.doc);
            //bufferwriter.newLine();
            Terms docTerms = reader.getTermVector(scoreDoc.doc, "contents");

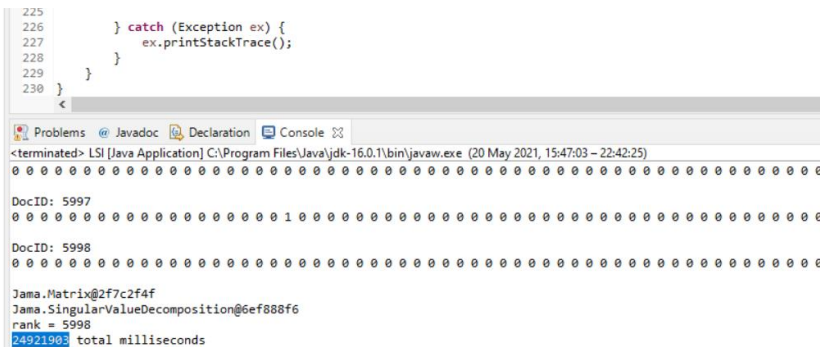
            Double[] vector = DocToDoubleVectorUtils.toSparseLocalFreqDoubleArray(docTerms, fieldTerms); //creates document's vec
            NumberFormat nf = new DecimalFormat("0.#");
            for(int i = 0; i<vector.length-1; i++) {
                System.out.print(nf.format(vector[i]) + " "); //prints document's vector
                //bufferwriter.write(nf.format(vector[i]) + " ");
            }
            //bufferwriter.newLine();
            System.out.println();
            System.out.println();
        }
    }
    //bufferwriter.close();
}
```

ΒΗΜΑΤΑ 2

Σε αυτό το βήμα διαβάσαμε το αρχείο με τα **terms** σε **Python** το οποίο έχουμε δημιουργήσει ποιο πάνω έτσι ώστε να καταφέρουμε να κάνουμε **SVD Analysis**. Για να εξάγουμε τα αποτελέσματα που ήθελε η άσκηση δηλαδή για διαφορετικές τάξεις κάθε φορά παραγοντοποίησαμε τον πίνακα με την SVD και κάναμε προσέγγιση χαμηλού βαθμού τάξης 50, 150, 300 όπως βλέπεται ποιο κάτω.



**** Προσπαθήσαμε να κάνουμε SVD Analysis με JAVA με την βιβλιοθήκη JAMA άλλα χρειάστηκε 7 ώρες για να κάνει Compile η πράξη SVD. Σας παραθέτουμε το Screenshot ποιο κάτω**



ΒΗΜΑ 3

Για το τρίτο βήμα δημιουργήσαμε την μέθοδο που βλέπετε πιο κάτω η οποία προσθέτει όλα τα ερωτήματα στο ευρετήριο με τον ίδιο τρόπο όπως στο βήμα 1 με την διαφορά ότι τα **Queries** τα αποθηκεύουμε με διαφορετικό όνομα field **"fullQuery"**

```
private void indexQueries(IndexWriter indexWriter, QueryDoc queryDoc, FieldType type){
    try {
        // make a new, empty document
        Document query = new Document();

        // create the fields of the document and add them to the document
        String value = queryDoc.getText();

        Field field = new Field("fullQuery", value, type);

        query.add(field); //this field has term Vector enabled.
        indexWriter.addDocument(query);

    } catch (Exception e){
        e.printStackTrace();
    }
}
```

Ο λόγος που το κάναμε αυτό είναι για να καλέσουμε την μέθοδο **toSparseLocalFreqDoubleArray** και ουσιαστικά να φτιάξουμε το **queryvector** ο οποίος κάνει **match** τα **terms** του **query** εκεί που βρίσκονται τα **terms** όλων των **docs** και έχει μήκος ίσο με το μήκος όλων των **terms**(13579) για κάθε **query**.

```
private static void createQueriesTermArray(IndexReader queryReader) throws Exception {
    // File svd = new File("C:\\Users\\User\\Desktop\\ProjectPart2\\QueryTerms.txt");
    // FileWriter writer = new FileWriter(svd,true);
    // BufferedWriter bufferwriter = new BufferedWriter(writer);
    Terms QueryfieldTerms = MultiFields.getTerms(queryReader, "contents"); //the number of terms in the lexicon after analysis
    System.out.println("Terms:" + QueryfieldTerms.size());
    double[][] fullVector= new double[13579][35];
    TermsEnum it = QueryfieldTerms.iterator(); //iterates through the terms of the lexicon
    while(it.next() != null) {
        System.out.print(it.term().utf8ToString() + " "); //prints the terms
    }
    System.out.println();
    System.out.println();
    if (QueryfieldTerms != null && QueryfieldTerms.size() != -1) {
        int k =1;
        for (int i=0 ; i<=34 ; i++) { //retrieves all 35 queries
            System.out.println("QueryID: " + k );
            k++;
            Terms queryTerms = queryReader.getTermVector(i, "fullQuery");

            Double[] queryvector = DocToDoubleVectorUtils.toSparseLocalFreqDoubleArray(queryTerms, QueryfieldTerms); //creates queryvector
            NumberFormat nf = new DecimalFormat("0.#");
            for(int j= 0; j<=queryvector.length-1; j++ ) {

                fullVector[j][i]= queryvector[j]; //create termxQuery 2-dimensional array
                System.out.print(nf.format(queryvector[j])+ " "); //prints query's vector
                //bufferwriter.write(nf.format(queryvector[j])+ " ");
            }
            //bufferwriter.newLine();
            System.out.println();
            System.out.println();
        }
    }
    //bufferwriter.close();
}
```

ΒΗΜΑ 4

Σε αυτό το βήμα διαβάζουμε τα δύο αρχεία που το ένα περιέχει όλα τα **terms** των **query(13579x35)** και το δεύτερο περιέχει των πίνακα **Ak(13579x5999)** από την SVD Analysis. Έτσι ώστε να μπορέσουμε να εξάγουμε την συν ημιτονοειδή ομοιότητα των διανυσμάτων(δηλαδή να παράγουμε το εσωτερικό γινόμενο των ερωτημάτων με κάθε κείμενο και έπειτα να βρούμε τα **k** καλύτερα **score** των κειμένων για κάθε Query).

```
try {
File k_file1 = new File("C:\\Users\\User\\Desktop\\test\\k50.txt");
BufferedReader bufferedReader = new BufferedReader(new FileReader(k_file1));
Scanner scan = new Scanner(bufferedReader);
double[][] AVD_Analysis= new double[13579][5999];
String line = scan.next();
for (int j=0; j<5999 ; j++) {
    for (int i=0 ; i<13579 ; i++) {
        double number = Double.parseDouble(line);
        AVD_Analysis [i][j] = number;

        System.out.println("j=" + j);
        System.out.println("i=" + i);

        if(line.equals("-7.876406873768566474e-04")) {
            break;
        }
        line = scan.next();
    }
}




File k_file = new File("C:\\Users\\User\\Desktop\\test\\queryTerms.txt");
BufferedReader bufferedReader1 = new BufferedReader(new FileReader(k_file));
Scanner scan1 = new Scanner(bufferedReader1);
double[][] queries= new double[13579][35];
String line1 = scan1.next();
for (int j=0; j<35 ; j++) {
    for (int i=0 ; i<13579 ; i++) {
        double number2 = Double.parseDouble(line1);
        queries [i][j] = number2;

        System.out.println("j=" + j);
        System.out.println("i=" + i);

        if(line1.equals("3.784512506007922487e-03")) {
            break;
        }
        line1 = scan1.next();
    }
}

double results;
for (int k=0; k<5999; k++) {
    for (int j=0; j<35 ; j++) {
        for (int i=0 ; i<13579 ; i++) {
            results = queries[i][j]*AVD_Analysis[i][k];
            //Vriski ta result kai prepi na vroume ta kalitera
        }
    }
}
```

**** Στο zip αρχείο δεν παραθέτουμε τα txt αρχεία της SVD Analysis επειδή είναι πολύ μεγάλα για να ανέβουν στο eclass.**

 k50	5/23/2021 11:45 PM	Text Docu...	2,024,522 KB
 k150	5/24/2021 12:00 AM	Text Docu...	1,914,102 KB
 k300	5/24/2021 12:07 AM	Text Docu...	2,029,751 KB