Εργασία Ανάκτησης Πληροφορίας

Ομάδα ΚΟΥΡΕΑΣ ΑΘΑΝΑΣΙΟΣ ΑΜ:4392

ΓΕΩΡΓΙΟΥ ΚΩΝΣΤΑΝΤΙΝΟΣ ΑΜ:4333

Github Repository: thanoskour/ANAKTISI-PLIROFORIAS (github.com)

Για να δημιουργήσουμε μια συλλογή με τουλάχιστον 500 έγγραφα, μπορούμε να συλλέξουμε στίχους από τουλάχιστον 500 διαφορετικά τραγούδια θα πάρουμε έτοιμη συλλογή από το Kaggle η οποία έχει 643 artist και 44824 τραγούδια και εμείς θα επιλέξουμε 500 τραγούδια από αυτήν την συλλογή. Η συλλογή μας θα είναι σε αρχείο csv η οποία θα έχει το όνομα του artist,το όνομα του τραγουδιού,link για το τραγούδι και τους στίχους του τραγουδιού. Στο GitHub θα υπάρχει και ένα παράδειγμα για το πως είναι η μορφή του αρχείου μας η οποία βρίσκεται σε μορφή csv.

Μετά τη συλλογή των εγγράφων, πρέπει να τα επεξεργαστούμε για να μπορέσουμε να τα εισάγουμε στο σύστημα αναζήτησης. Πρώτα απ' όλα, πρέπει να εξάγουμε το κείμενο από τα έγγραφα και να το αποθηκεύσουμε σε κατάλληλη μορφή.

Στη συνέχεια, πρέπει να δημιουργήσετε ένα index με τη βοήθεια της Lucene για την αποθήκευση των εγγράφων και τη δυνατότητα αναζήτησής τους. Το index είναι μια βάση δεδομένων που δημιουργείται από τη Lucene και περιέχει μεταδεδομένα για τα εγγραφόμενα έγγραφα, καθώς και τα διανύσματα αναπαράστασης του κειμένου των εγγράφων που χρησιμοποιούνται για την αναζήτηση.

Για τη δημιουργία του index, πρέπει να δημιουργήσετε ένα νέο αντικείμενο IndexWriter και να προσθέσετε κάθε έγγραφο στο index με τη μέθοδο addDocument(). Πρέπει να ορίσετε έναν Analyzer για το index, ο οποίος είναι υπεύθυνος για το διαχωρισμό του κειμένου σε λέξεις και την εφαρμογή της διαδικασίας της stemming (απόκτηση της ρίζας μιας λέξης) και της αφαίρεσης των stopwords (λέξεων που δεν έχουν σημασία ως προς την αναζήτηση, όπως οι "ο", "η", "το").

Αφού δημιουργήσετε το index, μπορείτε να πραγματοποιήσετε αναζητήσεις στα εγγραφόμενα έγγραφα.Για να κάνετε αναζήτηση στα index σας με τη χρήση της βιβλιοθήκης Lucene, πρέπει να δημιουργήσετε έναν αντικείμενο της κλάσης IndexReader για να διαβάσετε το index και έναν αντικείμενο της κλάσης IndexSearcher για να εκτελέσετε αναζητήσεις στο index. Ακολουθεί ένα παράδειγμα κώδικα:

Directory indexDirectory = FSDirectory.open(Paths.get("path/to/index")); // Δημιουργία ενός directory object για το index

IndexReader indexReader = DirectoryReader.open(indexDirectory); // Δημιουργία ενός index reader object για να διαβάσετε το index

IndexSearcher indexSearcher = new IndexSearcher(indexReader); // Δημιουργία ενός index searcher object για να εκτελέσετε αναζητήσεις στο index

// Κατασκευή ενός query object για μια αναζήτηση στο index

```
QueryParser queryParser = new QueryParser("content", new StandardAnalyzer()); // Δημιουργία ενός query parser object για να διαβάσετε την αναζήτηση και να δημιουργήσετε ένα query object

Query query = queryParser.parse("search query"); // Δημιουργία ενός query object για την αναζήτηση


// Εκτέλεση της αναζήτησης με το query object

TopDocs topDocs = indexSearcher.search(query, 10); // Εκτέλεση της αναζήτησης με το query object και επιστροφή των κορυφαίων αποτελεσμάτων


// Επεξεργασία των αποτελεσμάτων

ScoreDoc[] hits = topDocs.scoreDocs;

for (ScoreDoc hit : hits) {
   int docId = hit.doc;
   Document doc = indexSearcher.doc(docId);
   String content = doc.get("content");
   // Επεξεργασία του αποτελέσματος
}
```

Για την ανάλυση των ερωτημάτων και των εγγράφων, θα επιλέγαμε μια προχωρημένη μέθοδο ανάλυσης κειμένου που θα περιλαμβάνει stemming, απαλοιφή stop words, επέκταση συνωνύμων, κλπ. Αυτό θα βελτιώσει την απόδοση του συστήματος αναζήτησης, καθώς θα επιτρέπει στους χρήστες να βρίσκουν ευκολότερα και αποτελεσματικότερα τις αναζητούμενες πληροφορίες.

Ωστόσο, για τις ειδικές ανάγκες αναζήτησης, όπως η αναζήτηση σε συγκεκριμένα πεδία (π.χ. στον τίτλο, όνομα δημιουργού), θα πρέπει να επιλέξουμε μια μέθοδο ανάλυσης που λαμβάνει υπόψη τα πεδία αυτά και να τα διαχωρίζει από το υπόλοιπο κείμενο. Αυτό μπορεί να γίνει μέσω της χρήσης της ανάλυσης κειμένου που υποστηρίζει την αναζήτηση πεδίου.

Για να διατηρήσετε πληροφορίες για την ιστορία αναζητήσεων, μπορείτε να χρησιμοποιήσετε μια βάση δεδομένων ή ένα αρχείο καταγραφής. Κάθε φορά που ένας χρήστης κάνει μια αναζήτηση, αποθηκεύετε τα στοιχεία της αναζήτησης στη βάση δεδομένων ή το αρχείο καταγραφής.

Αυτά τα στοιχεία μπορούν να περιλαμβάνουν την αναζητούμενη λέξη, την ημερομηνία και ώρα της αναζήτησης, τον αριθμό των αποτελεσμάτων που επιστράφηκαν, καθώς και άλλες πληροφορίες που μπορεί να θεωρείτε σημαντικές για την ανάλυση των δεδομένων.

Στη συνέχεια, μπορείτε να χρησιμοποιήσετε αυτές τις πληροφορίες για να προτείνετε εναλλακτικά ερωτήματα στους χρήστες. Για παράδειγμα, αν παρατηρήσετε ότι οι χρήστες σας αναζητούν συχνά ένα συγκεκριμένο θέμα, μπορείτε να προτείνετε ένα ερώτημα που σχετίζεται με αυτό το θέμα. Επίσης,

μπορείτε να χρησιμοποιήσετε την ιστορία των αναζητήσεων. Για παράδειγμα, αν ένας χρήστης αναζητά συνεχώς "τραγούδια", αλλά δεν βρίσκει αρκετά ενδιαφέροντα αποτελέσματα, το σύστημα μπορεί να προτείνει εναλλακτικά ερωτήματα όπως "είδη τραγουδιών ", "διαφορετικά είδη μουσικής".

Για την παρουσίαση των αποτελεσμάτων σε διάταξη με βάση τη συνάφεια τους με το ερώτημα, μπορούμε να χρησιμοποιήσουμε το βαθμονομικό σύστημα (ranking system) που παρέχει η Lucene. Η Lucene χρησιμοποιεί έναν αλγόριθμο που υπολογίζει το βαθμό συνάφειας (relevance score) ενός εγγράφου με το ερώτημα και ταξινομεί τα έγγραφα από το πιο σχετικό στο λιγότερο σχετικό.

Για την παρουσίαση των αποτελεσμάτων, μπορούμε να εμφανίζουμε τα αποτελέσματα σε σελίδες των 10 εγγράφων και να παρέχουμε στο χρήστη δυνατότητα προχώρησης στα επόμενα αποτελέσματα. Επίσης, μπορούμε να τονίζουμε τις λέξεις κλειδιά που βρίσκονται στα αποτελέσματα για να επισημάνουμε τη συνάφειά τους με το ερώτημα.

Επιπλέον, μπορούμε να παρέχουμε δυνατότητα ομαδοποίησης των αποτελεσμάτων με βάση κάποιο κριτήριο που θα ορίσουμε εμείς. Για παράδειγμα, μπορούμε να ομαδοποιήσουμε τα αποτελέσματα ανά κατηγορία εγγράφου, όπως άρθρα, βίντεο, εικόνες κλπ.