

Εργασία: Μηχανή αναζήτησης άρθρων σχετικών με τον COVID-19

Ιωάννα Κουγιά 2731
Ιωάννης Πρίφτη 3321

1. Εισαγωγή

Η εργασία αφορά στο σχεδιασμό και υλοποίηση ενός συστήματος αναζήτησης άρθρων, στην συγκεκριμένη περίπτωση tweets, σχετικών με τον Covid-19. Η υλοποίηση γίνεται με Java 11 χρησιμοποιώντας την βιβλιοθήκη Lucene-8.8.2, μια βιβλιοθήκη ανοιχτού κώδικα για την κατασκευή μηχανών αναζήτησης κειμένου. Για το UI χρησιμοποιούμε Java swing.

2. Συλλογή Εγγράφων

Για τα έγγραφα μας, επιλέξαμε μια έτοιμη συλλογή απο το kaggle. Λέγεται “COVID-19 Tweets” και περιέχει τα tweets με hashtag #covid19 σε ένα csv αρχείο. Από αυτό χρησιμοποιούμε τις 3000 πρώτες εγγραφές. Σε κάθε γραμμή βρίσκεται ένα διαφορετικό tweet, το οποίο αποτελεί την μονάδα εγγράφου, και σε κάθε στήλη ένα διαφορετικό πεδίο. Από τα πεδία του αρχικού csv κρατήσαμε τα εξής: “created_at”, “user_id”, “username”, “name”, “tweet”, “language”, “replies_count”, “retweets_count”, “likes_count”, “link”.

3. Δομή Προγράμματος

Στην εργασία χρησιμοποιούμε το σχεδιαστικό πρότυπο MVC(Model -View -Controller). Στο πακέτο model υλοποιούνται ο Indexer, ο Searcher και ο Spell_Checker. Στο πακέτο view βρίσκεται ό τι αφορά τη γραφική διεπαφή, και το πακέτο controller είναι ο διαμεσολάβητής υπεύθυνος για την επικοινωνία του model με το view. Η main του προγράμματος βρίσκεται στη MainWindow του view. Η mainWindow έχει ένα αντικείμενο Controller, μέσω του οποίου ενημερώνονται δεδομένα ή καλούνται μέθοδοι του πακέτου model.

Παρακάτω βλέπουμε πιο περιεκτικά τα περιεχόμενα του κάθε πακέτου.

3.1. Model

3.1.1. Indexer (Ανάλυση κειμένου και κατασκευή ευρετηρίου)

Την δημιουργία του ευρετηρίου την αναλαμβάνει η `createIndex`. Για την ανάλυση των document σε tokens χρησιμοποιούμε τον EnglishAnalyzer ο οποίος μεταξύ άλλων μετατρέπει τα πάντα σε lowercase, κάνει απαλλοιφή των stopwords, και κάνει stemming με τον αλγόριθμο του Porter. Η createIndex παίρνει ως είσοδο το csv, το οποίο κάνει parse και το αποθηκεύει σε μια λίστα. Κάθε στοιχείο της λίστας περιέχει μία γραμμή του csv, δηλαδή μία μονάδα εγγράφου (document). Έπειτα για κάθε στοιχείο της λίστας δημιουργούμε το document με τα πεδία (fields) του. Τα documents αποθηκεύονται μέσω της IndexWriter στον φάκελο Index.

3.1.2. Searcher (Αναζήτηση)

Στο πάνω μέρος της σελίδας, παρέχεται στον χρήστη ένα πεδίο εισαγωγής κειμένου στο οποίο μπορεί να πληκτρολογήσει λέξεις κλειδιά. Υπάρχει η δυνατότητα φθίνουσας ταξινόμησης με βάση κάποιο από τα εξής πεδία: ημερομηνία δημοσίευσης, απαντήσεις, retweets και likes καθώς επίσης και η δυνατότητα ταξινόμησης με βάση τη σχετικότητα των αποτελεσμάτων ή την σειρά που έχουν στο ευρετήριο.

Επειδή πρόκειται για posts στα social media και όχι άρθρα αποφασίσαμε ότι δεν θα είχε νόημα να δίνεται στον χρήστη η επιλογή αναζήτησης σε κάποιο συγκεκριμένο πεδίο. Έτσι η αναζήτηση γίνεται πάντα σε ένα συγκεκριμένο υποσύνολο των πεδίων, μέσω ενός αντικειμένου MultiFieldQueryParser για τα πεδία username, name, tweet και created_at.

Η μέθοδος `getHits` παίρνει σαν είσοδο το String που πληκτρολόγησε ο χρήστης, και δημιουργεί ένα αντικείμενο Query της Lucene. Στη συνέχεια, μέσω ενός αντικειμένου IndexSearcher, καλούμε την συνάρτηση `search` της Lucene η οποία και επιστρέφει τα αποτελέσματα.

Η Searcher περιέχει ακόμα τρεις βασικές μεθόδους:

- Την `getTweetsAndLinks` που παίρνει σαν όρισμα τα 10 αποτελέσματα μιας σελίδας, και επιστρέφει ένα ArrayList με ζευγάρια τις εγγραφές των πεδίων tweet, link. Θα δούμε αργότερα που χρησιμοποιείται.
- Την `sortBy` που καλείται μέσα στην `searchIndex` της Searcher, παίρνει σαν όρισμα το query και την επιλογή του χρήστη για την ταξινόμηση, είτε με βάση το Relevance είτε με βάση το Index Order, και επιστρέφει όλα τα αποτελέσματα της αναζήτησης ταξινομημένα.
- Την `filterSorter` η οποία παίρνει σαν όρισμα το πεδίο φθίνουσας ταξινόμησης που επέλεξε ο χρήστης, και χρησιμοποιώντας τον αλγόριθμο selection sort τροποποιεί τον Score Doc πίνακα (hits) με όλα τα αποτελέσματα.

3.2.3. Spell_Checker

Ένα αντικείμενο της κλάσης αυτής δημιουργείται όταν η λέξη που πληκτρολόγησε ο χρήστης δεν βρίσκεται στο index. Ο constructor της, παίρνει σαν όρισμα το String που πληκτρολόγησε ο χρήστης και χρησιμοποιώντας την SpellChecker της Lucene, και το λεξικό `eng_dictionary.txt` δημιουργεί μια λίστα suggestions, με 5 πιθανές προτάσεις. Επίσης έχει μια μέθοδο `getSuggestions` για να γυρνάει αυτή τη λίστα στον Controller.

3.2 Controller

3.3.1. Controller

Η κλάση του Controller αποτελείται από τις εξής βασικές μεθόδους.

- Την `makeIndex` που καλεί τη μέθοδο `createIndex` του Indexer
- Την `searchIndex`, που παίρνει σαν όρισμα το ερώτημα του χρήστη και υπολογίζει τα αποτελέσματα (hits). Αν ο χρήστης έχει επιλέξει φθίνουσα ταξινόμηση με βάση κάποιο πεδίο, η `filterSorter` της Searcher αναδιατάζει τα αποτελέσματα. Τέλος υπολογίζεται ο αριθμός των σελίδων που θα χρειαστεί για την παρουσίαση των αποτελεσμάτων και η `createPages` ομαδοποιεί τα αποτελέσματα στις σελίδες τους. Η εμφάνιση των αποτελεσμάτων ανα 10 ήταν κάτι που αποτέλεσε μεγάλη πρόκληση και θα αναλυθεί παρακάτω. Σε αυτό το σημείο αδράξαμε την ευκαιρία να επιστρέψουμε και τον αριθμό των σελίδων στο view. Στην περίπτωση που το hits είναι κενό, δημιουργεί ένα αντικείμενο Spell_Checker για να υπολογίσει τον πίνακα suggestions και η μέθοδος επιστρέφει -1 στην MainWindow για να σηματοδοτήσει αυτό το γεγονός.

- Η `createPages` παίρνει τα αποτελέσματα ανα 10 μέσω της `getNextPagesHits` και τα αποθηκεύει σε μια δομή ώστε να είναι εύκολα προσπελάσιμα.
- Η `getNextPagesHits` παίρνει σαν όρισμα την τρέχουσα σελίδα. Με την σκέψη ότι, το τελευταίο αποτέλεσμα κάθε σελίδας θα είναι το (τρέχουσα σελίδα * 10) – οστό αποτέλεσμα στον πίνακα `hits` [π.χ. το τελευταίο αποτέλεσμα της 3ης σελίδας θα είναι στην 30η θέση (πρακτικά) ή στην 29η θέση (προγραμματιστικά)], τότε, αν θεωρήσω ότι τρέχουσα σελίδα * 10 = `top`, για να πάρουμε τα αποτελέσματα της τρέχουσας σελίδας αρκεί να προσπελάσουμε και να κρατήσουμε τα στοιχεία από το `top-10` μέχρι το `top`, σε κάποια νέα προσωρινή δομή.
- Η `getResults` παίρνει σαν όρισμα την σελίδα, και εμφανίζει τα περιεχόμενα της στο χρήστη.
- Η `setVisibleResults` παίρνει σαν όρισμα ένα boolean, και είτε εμφανίζει είτε εξαφανίζει τα περιεχόμενα των Labels. Χρησιμοποιείται συχνά για τις εναλλαγές των σελίδων.
- Η `setContent` παίρνει σαν όρισμα τον αριθμό των αποτελεσμάτων κάποιας συγκεκριμένης σελίδας, και ενημερώνει το κείμενο των labels με το κείμενο του κάθε αποτελέσματος, και τους `mouse listeners` με το link του κάθε αποτελέσματος.
- Η `setSuggestions` παίρνει σαν όρισμα τον πίνακα `suggestions`, και ενημερώνει τα Labels με αυτά.

3.3.2. HistoryManager

Έχει ένα πεδίο `ArrayList` στο οποίο κρατάει το ιστορικό και αποτελείται από τις εξής μεθόδους:

- Η `create` παίρνει σαν όρισμα το `String` που πληκτρολόγησε ο χρήστης, και το προσθέτει στο `history.txt`.
- Η `loadHistory`, δημιουργεί ένα `ArrayList` και φορτώνει σε αυτό όλα τα περιεχόμενα του `history.txt`. Καλείτε στην `main` όταν ξεκινάει το πρόγραμμα.

3.3. View

3.3.1 MainWindow

Η `MainWindow` περιέχει την `main` του προγράμματος, στην οποία αρχικοποιούνται τα περιεχόμενα του UI. Δημιουργήσαμε τα περιεχόμενα αυτά μέσω του `windowBuilder` που προσφέρει το `eclipse`. Ακόμα η `main` ξεκινά την δημιουργία του `index` και φορτώνει το ιστορικό.

Μέσα στην μέθοδο `initialize` βρίσκονται και οι `actionListeners` των κουμπιών.

Ακολουθεί η περιγραφή της ροής του προγράμματος

Μόλις ο χρήστης πληκτρολογήσει λέξεις κλειδιά στο πεδίο κειμένου και πατήσει το κουμπί `Search` (ή `enter` στο πληκτρολόγιο), τότε καλείται ο αντίστοιχος `actionListener`, περνάει το `query` στον `Controller` και αυτός εκκινεί τη διαδικασία αναζήτησης. Ο `Controller` επιστρέφει τον αριθμό σελίδων στην `view`. Αν ο αριθμός σελίδων, είναι έγκυρος αριθμός (≥ 1) τότε γίνονται ορατά τα αποτελέσματα της πρώτης σελίδας. Αν οι σελίδες των αποτελεσμάτων είναι περισσότερες από μια, τότε εμφανίζεται το κουμπί `Next Page` κάτω δεξιά. Τα αποτελέσματα που εμφανίζονται στον χρήστη (10 σε κάθε σελίδα εκτός από την τελευταία που μπορεί να είναι λιγότερα) είναι τα `tweet` που συμφωνούν με τον όρο αναζήτησης και το ή τα πεδία ταξινόμησης και πηγέοντας το ποντίκι πάνω σε κάποιο αυτόματα γίνεται μπλέ όσο το ποντίκι βρίσκεται πάνω του. Όταν ο χρήστης κάνει κλικ με το ποντίκι πάνω σε κάποιο αποτέλεσμα τότε αυτόματα ανακατευθύνεται μέσω του `browser` στο `twitter` όπου βρίσκεται το αντίστοιχο `tweet`. Στην περίπτωση που οι σελίδες είναι -1, αυτό για εμάς σημαίνει ότι ο χρήστης πληκτρολόγησε μια λέξη που δεν υπάρχει στο `index` και τότε αντί για αποτελέσματα, θα εμφανιστούν πεντε παραπλήσιες λέξεις. Δεν καταφέραμε να τις κάνουμε `clickable` και να γίνεται ανακατευθυνση στην πρώτη σελίδα των αποτελεσμάτων τους. Συνεπώς ο χρήστης θα πρέπει να πληκτρολογήσει ξανά. Ωστόσο, υπάρχει και η πιθανότητα να μην βρεθούν παραπλήσιες λέξεις. Σε αυτή την περίπτωση δεν θα εμφανιστεί τίποτα.

Αν ο χρήστης πατήσει το κουμπί **Next Page**, καλείται ο αντίστοιχος `addListener`. Ο χρήστης τώρα βρίσκεται στην 2η σελίδα, όπως αναγράφεται και στο κάτω μέσο μέρος της σελίδας, κάτω αριστερά είναι το κουμπί **Previous Page** και τα επόμενα 10 αποτελέσματα είναι πλέον ορατά. Ο χρήστης μπορεί να πατάει το κουμπί **Next Page** μέχρι να φτάσει στην τελευταία σελίδα, όπου πλέον δεν θα υπάρχει **Next Page**.

Αντίστοιχα λειτουργεί ο `addListener` του κουμπιού **“ Previous Page ”**.