Πανεπιστήμιο Ιωαννίνων **Κοντός Πέτρος ΑΜ:2898**
Τμήμα Μηχανικών Η/Υ & Πληροφορικής **Χατζηλαζάρου Παναγιώτης ΑΜ:2856**

Μηχανή αναζήτησης άρθρων σχετικών με τον COVID-19

Περιγραφή και Προ-επεξεργασία Συλλογής:

Η συλλογή των εγγράφων έγινε πραγματοποιώντας μία αναζήτηση στην έτοιμη συλλογή που μας δόθηκε, ώστε να κρατήσουμε τα άρθρα που σχετίζονται με τον ιό Covid-19 του τρέχοντος έτους. Πιο συγκεκριμένα στο αρχείο **corpusCreation.py** αποθηκεύουμε τα κατάλληλα άρθρα σε μορφή json σε έναν φάκελο corpus ενώ επίσης κρατήσαμε σε ένα αρχείο date.txt τις ημερομηνίες δημοσίευσης των εκάστοτε άρθρων. Η συλλογή έχει μέγεθος 678 άρθρων.

Η προ-επεξεργασία της συλλογής υλοποιείτε στην κλάση **txtGenerator.java.** Για να γίνει εφικτή η υλοποίηση του ευρετηρίου με τα κατάλληλα έγγραφα, κατασκευάσαμε στην κλάση ένα αρχείο **euretirio.txt** στο οποίο αποθηκεύσαμε για κάθε άρθρο, το **URL** , τον **τίτλο** , τους **αρθρογράφους**, την **περίληψη** και τέλος το **βασικό κείμενό** του χωρισμένα ανά γραμμή.

Ευρετηριοποίηση:

Η κατασκευή του ευρετηρίου υλοποιείτε στην κλάση **Indexer.java**, στην οποία δημιουργούνται τα έγγραφα με τα κατάλληλα πεδία τα οποία προστίθενται έπειτα στο ευρετήριο που θα χρησιμοποιήσουμε για την αναζήτηση. Αναλυτικότερα για την ανάλυση των πεδίων χρησιμοποιήθηκε ο **StandardAnalyzer** που παρέχει η Lucene, ο οποίος μπορεί και αναγνωρίζει URL, emails, κάνει απαλοιφή stop words , μετατροπή σε lower case, ενώ επίσης αφαιρεί σημεία στίξης. Έτσι για κάθε άρθρο στο αρχείο euretirio txt προστίθενται τα πέντε πεδία που αναφέραμε παραπάνω ως **STORED**, και επιπρόσθετα δύο ακόμα πεδία, η **ημερομηνία δημοσίευσης** published_time και όλο **το περιεχόμενο του κάθε άρθρου** ως **contents**. Στη συνέχεια αυτά τα άρθρα προστίθενται στο ευρετήριο με τον **IndexWriter** που μας παρέχει η Lucene και αποθηκεύετε σε έναν φάκελο στον δίσκο ώστε να χρησιμοποιηθεί για την αναζήτηση.

Αναζήτηση των άρθρων:

Η αναζήτηση των άρθρων υλοποιείτε στην κλάση **Searcher.java** σε συνεργασία με την κλάση **GUI.java** στην οποία κατασκευάζετε το περιβάλλον διεπαφής με τον χρήστη. Στον χρήστη δίνετε η δυνατότητα να κάνει αναζήτηση με αρκετούς τρόπους, όπως για παράδειγμα με λέξη κλειδί , με φράση, αναζήτηση σε συγκεκριμένο πεδίο ακόμα και αναζήτηση με συνδυασμό πεδίων. Επίσης του δίνεται δυνατότητα να κάνει αναζήτηση σε ένα πεδίο με Boolean ερωτήματα με δύο όρους.

Αναλυτικότερα:

- Αναζήτηση με βάση τα περιεχόμενα(contents) του αρχείου:
  Όταν ο χρήστης επιλέξει την συγκεκριμένη επιλογή από το μενού τότε καλείτε η συνάρτηση **passValuesAndSearch** στην κλάση GUI με παραμέτρους το όνομα του πεδίου, στην συγκεκριμένη περίπτωση "contents" και το αλφαριθμητικό – λέξη κλειδί με το οποίο θέλει να γίνει αναζήτηση άρθρων. Στην συνάρτηση αυτή αρχικά δημιουργείτε ένας νέος Searcher και ελέγχοντας αν ο χρήστης έδωσε **λέξη κλειδί** ή **φράση** καλείτε αντίστοιχα η μέθοδος **searchIndex** ή **searchPhraseQuery** οι οποίες θα αναλυθούν παρακάτω.

- Αναζήτηση με βάση κάποιο πεδίο του αρχείου:
  Ο χρήστης επιλέγει από το μενού **Search by Field** και του εμφανίζονται τα τέσσερα πεδία που επιλέξαμε να έχουν τα documents μαζί με ένα JTextField στο οποίο μπορεί να βάλει το ερώτημά του. Όπως και παραπάνω ανάλογα με το αν είναι φράση ή όρος καλείτε η αντίστοιχη μέθοδος του Searcher με παραμέτρους το όνομα του πεδίου που επέλεξε να κάνει αναζήτηση ο χρήστης και το περιεχόμενο του JTextField. Γίνεται έλεγχος ώστε να διαπιστωθεί αν ο χρήστης έκανε λογική ερώτηση όπως για παράδειγμα (covid AND death), αναζήτηση με φράση (π.χ. death cases) ή με απλό ορο και κατασκευάζεται το κατάλληλο **Query.**

- Αναζήτηση με συνδυασμό πεδίων:
  Στο πρόγραμμα προσθέσαμε μια ακόμη επιλογή ώστε ο χρήστης να μπορεί να κάνει αναζήτηση με συνδυασμό δύο πεδίων. Στο μενού υπάρχει επιλογή **Combine Fields** στην οποία του παρουσιάζονται τα πεδία σαν check boxes και έχει τη δυνατότητα να επιλέξει δύο από αυτά για να αναζητήσει τα πιο συναφή άρθρα. Επιλέξαμε στην συγκεκριμένη ιδιότητα να επιστρέφονται τα άρθρα που έχουν και στα δυο πεδία τον όρο που έδωσε ο χρήστης δηλαδή Boolean query τύπου AND. Για να γίνει η αναζήτηση στην συγκεκριμένη περίπτωση καλείτε η μέθοδος **searchBoolenIndex** της κλάσης Searcher με παραμέτρους τα ονόματα των πεδίων που επέλεξε ο χρήστης και το ερώτημα αναζήτησης.

Όσον αφορά την μέθοδο **serachIndex**, δημιουργείτε ένας reader με τη βοήθεια του **IndexReader** της lucene στον φάκελο που είναι αποθηκευμένο το ευρετήριο και ένας **IndexSearcher** πάνω σε αυτόν. Έπειτα το ερώτημα του χρήστη περνάει από τον Standard Analyzer και με τη βοήθεια του **Query Parser** δημιουργείτε το **Query** πάνω στο ερώτημα. Χρησιμοποιήσαμε τις **TopDocs** και **ScoreDocs** για να μας επιστραφούν τα πιο συναφή άρθρα.

Στην μέθοδο **searchPhraseQuery** ανάλογα με την ερώτηση που έκανε ο χρήστης δημιουργείτε το κατάλληλο Query το οποίο επιστρέφει η συνάρτηση **checkBooleanClause.** Η μέθοδος αυτή ελέγχει αν στο ερώτημα υπάρχει λογικός όρος(AND , OR, NOT, AND NOT, OR NOT) και κατασκευάζει ένα **BooleanQuery** με τα κατάλληλα flags το οποίο επιστρέφεται ώστε να γίνει αναζήτηση με αυτό. Αν το ερώτημα του χρήστη δεν περιέχει κάποιον λογικό έλεγχο τότε θα γίνει αναζήτηση με ολόκληρη τη φράση και για τον σκοπό αυτόν δημιουργείτε ένα query με την βοήθεια της μεθόδου **createPhraseQuery** η οποία αναζητά στο ευρετήριο ολόκληρη τη φράση.

Τέλος στη μέθοδο **searchBoolenIndex** η αλλαγή είναι ότι δημιουργούμε έναν **MultiFieldQueryParser** ο οποίος μπορεί να κάνει αναζήτηση σε πολλαπλά πεδία με βάση Boolean όρους, για το κάθε πεδίο.

Ιστορικό Αναζήτησης:

Για το ιστορικό αναζήτησης χρησιμοποιήσαμε το αρχείο wn_s.pl [1] από τη βάση δεδομένων WordNet το οποίο περιέχει ένα hash map με συνώνυμα στα οποία μπορεί να γίνει αναζήτηση με βάση έναν όρο. Για τον σκοπό αυτόν δουλέψαμε με την επιλογή που παρέχει η Lucene να κατασκευάσεις τον δικό σου **CustomAnalyzer**, στον οποίο προστίθενται τα φίλτρα LowerCase, StopFilter και SynonymGraphFilter. Ο constructor του SynonymGraphFilter παίρνει σαν είσοδο το hash map με τα συνώνυμα από το wordnet.

Τον παραπάνω analyzer χρησιμοποιεί η συνάρτηση **analyzeSynonyms** στην οποία καλώντας την μέθοδο tokenstream πάνω στον analyzer, μας επιστρέφονται τα συνώνυμα με βάση τον όρο που έκανε αναζήτηση ο χρήστης και τα αποθηκεύουμε σε μια λίστα. Έπειτα τις λέξεις αυτές τις κάναμε κουμπιά στο μενού μας έτσι διαλέγοντας ο χρήστης την επιλογή **History** από το μενού, να του παρέχονται η λέξη που έκανε αναζήτηση μαζί με τα συνώνυμα της. Πατώντας τώρα πάνω σε ένα από τα συνώνυμα γίνεται αναζήτηση στα περιεχόμενα του αρχείου και του επιστρέφονται τα πιο συναφή.

Παρουσίαση Αποτελεσμάτων:

Για να δώσουμε στον χρήστη την επιλογή να βλέπει τα αποτελέσματα ανά δέκα, τροποποιήσαμε την μέθοδο search του IndexSearcher ώστε να μας επιστρέφει όλα τα έγγραφα που υπάρχουν στο ευρετήριο. Έτσι στην συνέχεια η scorDocs κρατάει τα σκορ όλων τον εγγράφων που υπάρχουν στο ευρετήριο μας και σχετίζονται με την ερώτηση του χρήστη. Κατά αυτόν τον τρόπο στον χρήστη παρουσιάζονται τα πρώτα δέκα αποτελέσματα και το σκορ τους, και του παρουσιάζεται μήνυμα αν θέλει να συνεχίσει στα επόμενα δέκα, έως ότου να τελειώσουν τα έγγραφα. Επίσης στην παρουσίαση αποτελεσμάτων επιλέξαμε όταν ο χρήστης κάνει αναζήτηση σε ολόκληρο το αρχείο, δηλαδή αναζήτηση με βάση τα contents του αρχείου, να του επιστρέφουμε τον σύνδεσμο του αρχείου και το σκορ του, έτσι ώστε να μπορεί να διαβάσει ολόκληρο το άρθρο από την πηγή του.

Όταν ο χρήστης κάνει αναζήτηση με βάση κάποιο πεδίο, επιλέξαμε να του παρουσιάζουμε το περιεχόμενο του πεδίο στο εκάστοτε άρθρο και επιπρόσθετα να γίνεται επισήμανση του όρου-φράσης που αναζητήθηκε. Για τον σκοπό αυτόν χρησιμοποιήσαμε τον **Fragment** και τον **Highlighter** που μας παρέχει η Lucene οι οποίοι βρίσκουν τον όρο στο πεδίο που έγινε αναζήτηση και εμφωλεύουν τον όρο σε bold ταμπέλα html.

Τέλος όσον αφορά το **grouping** των αποτελεσμάτων, επιλέξαμε να γίνεται με βάση την ημερομηνία δημοσίευσης του άρθρου. Για τον σκοπό αυτόν προσθέσαμε στο μενού μας επιλογή **GroupResults**, την οποία όταν την επιλέξει ο χρήστης κατασκευάζεται ένα HashMap με κλειδιά τις ημερομηνίες και τιμές κάθε κλειδιού τα άρθρα(ο σύνδεσμός τους). Έτσι παρουσιάζονται για κάθε ημερομηνία τα αντίστοιχα άρθρα που δημοσιευτήκαν.

[1] Πηγή: https://wordnet.princeton.edu/

Σημείωση: Η έκδοση της lucene που χρησιμοποιήσαμε είναι η **8.8.2** και προσθέσαμε το build path τα παρακάτω jars.

- commons-io-2.8.0
- Import την θέση του αρχείου wn_s.pl με τα συνόνημα
- Json-20210307.jar
- json-simple -1.1.jar
- lucene-analyzers-common-8.8.2.jar
- lucene-core-8.8.2.jar
- lucene-demo-8.8.2.jar
- lucene-grouping-8.8.2.jar
- lucene-highlighter-8.8.2.jar
- lucene-memory-8.8.2.jar
- lucene queries-8.8.2.jar
- lucene-queryparser-8.8.2.ajr
- org-jdesktop-swingx.jar