**ΕΡΓΑΣΙΑ ΑΝΑΚΤΗΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ**

Η μηχανή αναζήτησης έχει αναπτυχτεί με τη βιβλιοθήκης Lucene της Apache και έκδοση 7.1.0. Επίσης χρησιμοποιήθηκε και η βιβλιοθήκη jsoup για την υλοποίηση του crawler. Παρακάτω περιγράφονται όλες οι κλάσεις της εφαρμογής.

**Indexer:** Η κλάση αύτη δημιουργεί τον αντεστραμμένο κατάλογο και προσφέρει λειτουργιές πρόσθεσης αρχείων σε αυτόν βάσει ενός path ή ενός link (αφετηρία για crawling) που δίνεται ως είσοδος. Κάθε είσοδος στον κατάλογο περιέχει 4 πεδία: CONTENTS (Το επεξεργασμένο περιεχόμενο του αρχείου), NAME (το όνομα του αρχείου), PATH (το file path αν είναι αρχείο κειμένου) και URL (αν είναι ιστοσελίδα). Οι μέθοδοι περιγράφονται παρακάτω:

* **Constructor:** Ουσιαστικά ο κατασκευαστής δημιουργεί 2 ξεχωριστούς αντεστραμμένους καταλόγους, έναν με Boolean βαθμολόγηση και έναν με βαθμολόγηση συνημίτονου, και ένα αντικείμενο της κλάσης TextFileFilter που ελέγχει εάν τα αρχεία είναι τύπου txt.
* **close & commit:** Άπλες μέθοδοι που κλείνουν τους index writers και καταγράφουν τα αρχεία που έχουν επεξεργαστεί οι writers στην μνήμη αντίστοιχα.
* **index:** Διαβάζει όλα τα περιεχόμενα ενός path που έχει οριστεί ως είσοδος, επιλεγεί μόνο όσα είναι τύπου txt και στέλνει κάθε ένα από αυτά για επεξεργασία μέσω της *indexFile.*
* **indexFile:** Δημιουργεί ένα αντικείμενο τύπου *Document* το οποίο στη συνεχεία αποθηκεύει στον κατάλογο εάν δεν είναι ήδη αποθηκευμένο (Ο έλεγχος γίνεται με βάση το πεδίο PATH όποτε εάν δυο αρχεία έχουν κοινό περιεχόμενο αλλά διαφορετικό όνομα θα αποθηκευτούν και τα 2).
* **indexUrl:** Παρόμοια μέθοδος με την *indexFile* η όποια χρησιμοποιείται για ιστοσελίδες αντί για αρχεία κειμένου.
* **getFileDocument:** Δημιουργεί το αντικείμενο Document με βάση το αρχείο κειμένου που παίρνει ως είσοδο. Για τα πεδία CONTENTS και NAME γίνεται tokenization, stemming και αφαίρεση stop words. Το πεδίο PATH αποθηκεύεται αυτούσιο ώστε να είναι δυνατή η ανάκτηση και το πεδίο URL παίρνει την τιμή "-" αφού δεν πρόκειται για ιστοσελίδα.
* **getUrlDocument:** Παρόμοιε με την *getFileDocument* με διαφορά ότι το πεδίο URL παίρνει αυτούσιο τον σύνδεσμο της ιστοσελίδας και το πεδίο PATH παίρνει την τιμή "-" αφού δεν πρόκειται για αρχείο κειμένου.
* **crawl:** Δημιουργεί ένα αντικείμενο της κλάσης *Crawler* και το χρησιμοποιεί για να διασχίσει τον ιστό με αφετηρία τον σύνδεσμο που δίνεται ως είσοδος και στη συνεχεία καλεί την *indexUrl* για κάθε μια από τις ιστοσελίδες που επιστραφήκαν ώστε να προστεθούν στον κατάλογο.

**Searcher:** Η κλάση αύτη είναι υπεύθυνη για την αναζήτηση αρχείων στον αντεστραμμένο κατάλογο και επιστροφή των αποτελεσμάτων με βάση ένα query. Παρέχει την επιλογή αναζήτησης με βάση το Boolean ή το Vector Space μοντέλο. Σε περίπτωση αναζήτησης με το πρώτο το query πρέπει να δομηθεί με εισαγωγικά ("") ανάμεσα στους όρους που επιθυμεί ο χρήστης δηλαδή ως phrase query. Τα κενά έκτος τω εισαγωγικών διερμηνεύονται ως OR και για αυτό δεν είναι απαραίτητο να δοθεί λογική έκφραση ως ερώτημα (πχ το ερώτημα |Sherlock Holmes is great| μετατρέπεται σε |"Sherlock" OR "Holmes" OR "is" OR "great"| ενώ το |"Sherlock Holmes" "is great"| σε|"Sherlock Holmes" OR "is great"|). Μπορούν να χρησιμοποιηθούν οι λέξεις AND OR NOT (κεφάλαια μόνο) σε συνδυασμό με παρενθέσεις για την δόμηση λογικών εκφράσεων ανεξαρτήτως του μοντέλου με το οποίο πραγματοποιείται η αναζήτηση. Επίσης υποστηρίζονται queries όπως wildcard με χρήση αστερίσκου (πχ |food\*| βρίσκει όρους που αρχίζουν με food) σε οπουδήποτε σημείο του ορού έκτος από την αρχή, fuzzy με χρήση ~ μετά τον όρο για παρόμοιες λέξεις (πχ |roam~| βρίσκει το foam) phrase με χρήση εισαγωγικών, range με χρήση ~ και έναν θετικό ακέραιο στο τέλος του ορού και boosting ορών με χρήση ^ και έναν θετικό αριθμό στο τέλος του ορού (πχ |roam^3|). Οι μέθοδοι περιγράφονται παρακάτω:

* **Constructor:** Ο κατασκευαστής δημιουργεί έναν αναγνώστη του καταλόγου και στη συνεχεία δυο μηχανές αναζήτησης, μια για αναζήτηση με το Boolean μοντέλο και μια με το Vector space μοντέλο. Στη συνεχεία δημιουργεί και έναν query parser για την επεξεργασία των ερωτημάτων χρηστή.
* **search:** Επεξεργάζεται το query και στη συνεχεία αναζητά και επιστρέφει τα top-k αποτελέσματα ως αντικείμενο TopDocs.
* **getDocument:** Επιστρέφει το περιεχόμενο ενός αντικείμενου scoreDoc (από τα top-k) σε μορφή Document.
* **escapeQuery:** Χρησιμοποιείται για την μορφοποίηση ενός query.

**Crawler:** Η κλάση αύτη υλοποιεί τον web crawler χρησιμοποιώντας ως agent για την είσοδο σε σελίδες αυτόν που χρησιμοποιεί ο crawler 80legs (Δεν έχει καμιά άλλη σχέση με τον 80legs). Οι μέθοδοι περιγράφονται παρακάτω:

* **Constructor:** Αρχικοποιεί ένα HashSet για τις να κρατεί τις σελίδες από τις οποίες έχει περάσει και ένα LinkedList για τις σελίδες τις οποίες θα επισκεφτεί στο μέλλον.
* **start:** Αρχικοποιεί μια λίστα τύπου *UrlNode* που θα κρατεί τα περιεχόμενα των σελίδων και καλεί την μέθοδο *crawl* μέχρι να φτάσει των μέγιστο αριθμό σελίδων ο όποιος δίνεται ως είσοδος (10 default), ξεκινώντας από url που δίνεται και αυτό ως είσοδος. Εάν "κολλήσει" για πάνω από 1000 κύκλους σταματήσει. Στο τέλος επιστρέφεται η λίστα με τις σελίδες που επισκέφτηκε. \*\*Σε περίπτωση που διακοπεί το crawling πριν την ολοκλήρωση του δεν προσθέτεται καμιά σελίδα στον κατάλογο καθώς γίνεται indexing όλων μαζί στο τέλος\*\*
* **nextUrl:** Επιστρέφει τον πρώτο σύνδεσμο τον οποίο δεν έχει ήδη επισκεφτεί ο crawler από την λίστα συνδέσμων προς επίσκεψη.
* **crawl:** Διαβάζει τα περιεχόμενα της σελίδας που παίρνει ως είσοδο με τη βοήθεια της βιβλιοθήκης jsoup, επιστρέφει το body και το title της σελίδας με καθώς και την διεύθυνση της ως ένα αντικείμενο UrlNode και προσθέτει όλους τους συνδέσμους που διαβάζει στην λίστα σελίδων προς επίσκεψη.

**Engine:** Κλάση για την χρήση λειτουργιών των παραπάνω. Οι μέθοδοι περιγράφονται παρακάτω:

* **Constructor:** Δημιουργεί έναν analyzer για την επεξεργασία κειμένου (tokenization, stemming και αφαίρεση stop words) και αντικείμενα των κλάσεων *Indexer*, *Searcher* και *Crawler*.
* **index:** Παίρνει ως είσοδο ένα path και κάνει index όλα τα txt αρχεία που βρίσκονται μέσα σε αυτό.
* **search:** Παίρνει ως είσοδο ένα ερώτημα, την επιλογή Boolean h VSM αναζήτησης και τον αριθμό top-k αποτελεσμάτων και αναζητεί τον κατάλογο βάσει του ερωτήματος.
* **crawl:** Παίρνει ως είσοδο ένα url και τον αριθμό σελίδων προς crawling και ξεκινά το crawling από το url αυτό.
* **close:** Απλή μέθοδος για το κλείσιμο του indexer.

**TextFileFilter:** Απλή κλάση φίλτρο για να ελέγχεται αν ένα αρχείο είναι τύπου txt.

**UrlNode:** Απλή κλάση για την δημιουργεί αντικειμένων που περιέχουν πεδία με τον τίτλο το σώμα και τη διεύθυνση ιστοσελίδων.

**LuceneConstants:** Απλή κλάση στην όποια περιέχονται σταθερές που χρησιμοποιούνται κατά τη λειτούργει του προγράμματος.

*Όλες οι κλάσεις περιέχονται μέσα στον φάκελο dist μαζί με το jar αρχείο και τις βιβλιοθήκες που χρησιμοποιήθηκαν.*

Για εκκίνηση του jar μέσω command line δίνονται παρακάτω τα ορίσματα που πρέπει να προστεθούν για κάθε λειτούργει μετά την εντολή "java -jar path/to/SearchEngine.jar":

* indexing: "index <PATH>"

PATH: Το path στο οποίο περιέχονται τα αρχεία προς προσθήκη στον κατάλογο.

* searching: "search <QUERY> <MODEL> <TOP-K>"

QUERY: Το ερώτημα χρηστή.

MODEL: "true" για αναζήτηση με Boolean model και "false" για αναζήτηση με VSM.

TOP-K: integer με τον αριθμό των αποτελεσμάτων προς επιστροφή.

* crawling: "crawl <URL> <MAXPAGES>"

URL: Η διεύθυνση της σελίδας αφετηρίας.

MAXPAGES: integer με τον αριθμό των σελίδων προς επίσκεψη.

Βεβαιωθείτε ότι το τρέχετε ως Administrator για να μην βγάλει AccessDeniedException Error.

Γενικά υποστηρίζεται η προσπέλαση μέσω browser αλλά μόνο από localhost καθώς είχα προβλήματα με την java στον server του πανεπιστήμιου (σας είχα στείλει και σχετικό mail). Όλα τα απαραίτητα αρχεία για την ιστοσελίδα και το jar περιέχονται στον φάκελο SEhtml.

*\*Εάν θέλετε να διαπιστώσετε την απόδοση καθαρού indexing μπορείτε να τρέξετε το jar από cl και όχι μέσω php λογω του ότι εκεί χρησιμοποιείται μαζί με άλλες λειτουργιές όπως uploading και crawling οι οποίες παίρνουν παραπάνω χρόνο.*