

2η Φάση εργασίας (Μηχανή αναζήτησης πληροφορίας από επιστημονικά άρθρα)

Αναφορά Εργασίας

22.05.2024

Άγγελος Ανδρέου

A.M. 4628

GITHUB LINK:

<https://github.com/AngelAndreou/Scientific-Papers-Search-Engine.git>



GITHUB LINK: <https://github.com/AngelAndreou/Scientific-Papers-Search-Engine.git>

| | |
|--|----------|
| 1. Εισαγωγή: | 2 |
| 2. Συλλογή εγγράφων (corpus). | 2 |
| 3. Ανάλυση κειμένου και κατασκευή ευρετηρίου. | 2 |
| 4. Αναζήτηση. | 3 |
| 5. Παρουσίαση Αποτελεσμάτων. | 4 |

1. Εισαγωγή:

Στην εργασία του μαθήματος μας ζητήθηκε ο σχεδιασμός και η υλοποίηση ενός συστήματος αναζήτησης πληροφορίας από επιστημονικά άρθρα. Για την υλοποίηση του ζητούμενου κατασκευάστηκε μηχανή αναζήτησης κειμένου με χρήση της δωρεάν βιβλιοθήκης Lucene 9.10, που σε μια μεγάλη συλλογή από άρθρα αναζητά τους όρους που δίνει ο χρήστης και εμφανίζει σαν αποτέλεσμα το id αριθμό του άρθρου τον τίτλο και απόσπασμα κειμένου που βρέθηκε ο όρος αναζήτησης. Η μηχανή γράφτηκε σε γλώσσα Java με τη χρήση eclipse και είναι εφαρμογή παραθύρου.

2. Συλλογή εγγράφων (corpus).

Για την δημιουργία συλλογής (corpus) έγινε χρήση έτοιμου dataset από επιστημονικά άρθρα στο kaggle. Συγκεκριμένα η συλλογή “All NeurIPS (NIPS) Papers”, στο papers.csv αρχείο που τα πεδία χωρίζονται με κόμμα και είναι τα εξής: source_id, year, title, abstract, full_text. Το source_id είναι ο κωδικός του άρθρου, το year η ημερομηνία έκδοσης, το abstract περιέχει την περίληψη του άρθρου και το full_text είναι το σώμα κειμένου του άρθρου και βρίσκεται ανάμεσα από εισαγωγικά(“”).

Link:

<https://www.kaggle.com/datasets/rowhitsuami/nips-papers-1987-2019-updated/data?select=papers.csv>

Συγκεκριμένα για την εφαρμογή χρησιμοποιήθηκαν 250 άρθρα από την παραπάνω συλλογή που είχαν διαθέσιμη περίληψη (abstract) ώστε να έχουμε τιμές σε όλα τα πεδία..

3. Ανάλυση κειμένου και κατασκευή ευρετηρίου.

Η ανάλυση κειμένου έγινε με τη χρήση του api της Lucene “analysis.standard.StandardAnalyzer” για την αγγλική γλώσσα με τη “standar analyzer” έκδοχή που θα κάνει το βασικό stemming, απαλοιφή stop words κ.α. που χρειαζόμαστε.. Τα πεδία (fields) που

δημιουργούν το ευρετήριο είναι το id, η χρονολογία, ο τίτλος, η περίληψη και το κείμενο του άρθρου, ώστε να υποστηρίζονται οι αντίστοιχοι τρόποι αναζήτησης βάση του πεδίου τίτλος, περίληψη ή και στο βασικό κείμενο του άρθρου και η αναδιάταξη βάση την χρονιά. Η συλλογή που χρησιμοποιήθηκε για την δημιουργία είναι η “lucene.document” και “lucene.index”, ώστε να ορίσουμε τα indexes και τις υποκατηγορίες για αναζήτηση.

Για να γίνει η αποθήκευση στο σκληρό δίσκο έγινε χρήση των apis lucene.store.Directory και lucene.store.FSDirectory και παραχωρήθηκε το “path” στον τοπικό δίσκο μου (“indexPath=”C:\\Users\\angel\\eclipse-workspace2\\index”;”). Όλη η διαδικασία πραγματοποιείται από την κλάση: “CSVIndexer” που φτιάχνει το index μία φορά (πρώτη φορά) όταν καλείτε από την main της βασικής κλάσης “OfflineSearchEngine” με το πεδίο (“boolean buildIndex=true”).

Η προεπεξεργασία των άρθρων γίνεται διαβάζοντας το αρχείο papers.csv και επεξεργασία των γραμμών του με το εργαλείο regex της JAVA όπου διαβάζει γραμμή γραμμή το έγγραφο και παράγει τα πεδία αντιστοιχίζοντας τα δεδομένα του αρχείου με το αντίστοιχο πεδίο για 250 εγγραφές που ταιριάζουν στο πρότυπο.

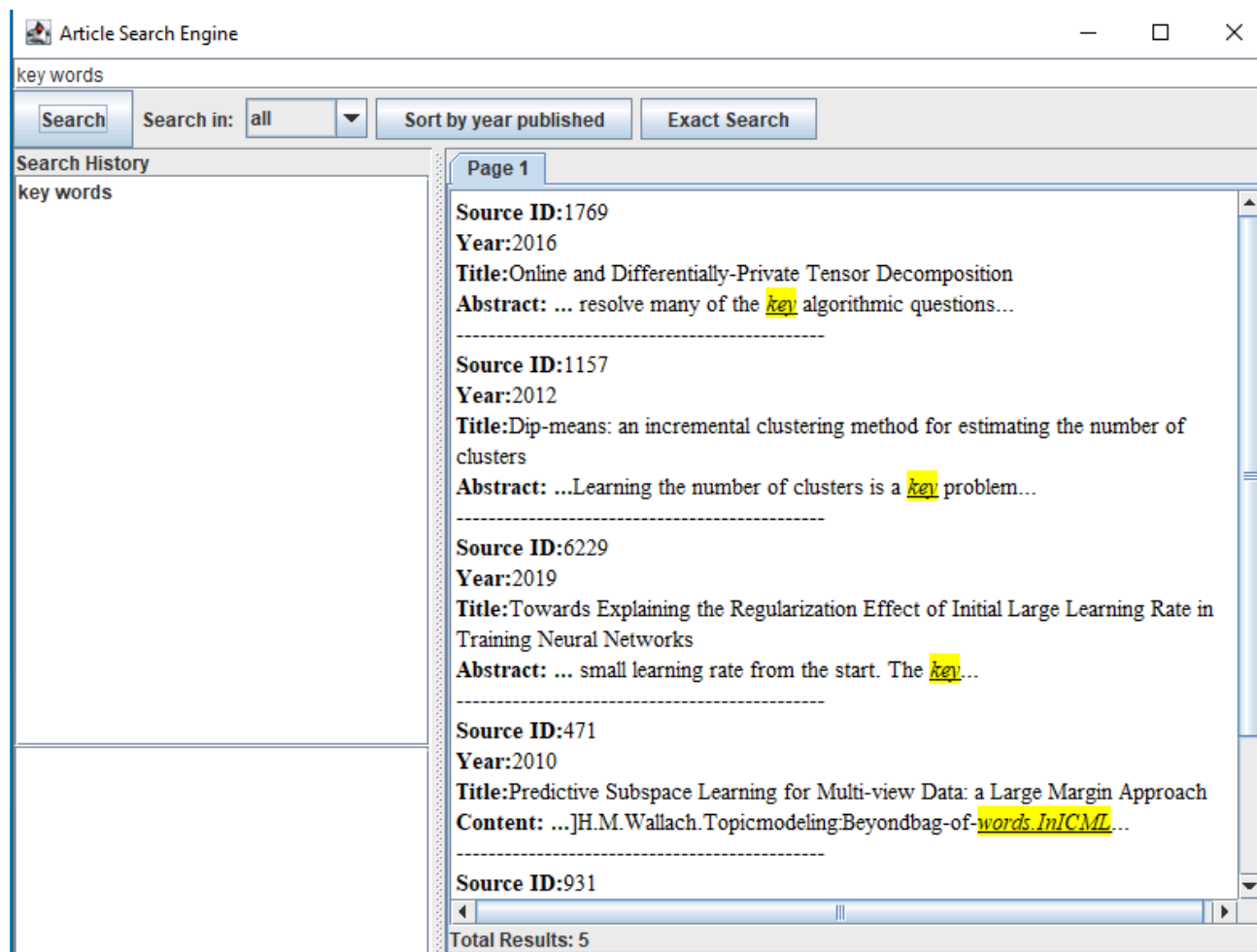
4. Αναζήτηση.

Το παράθυρο της εφαρμογής (όπως φαίνεται στην [Εικόνα 1](#)) για την αναζήτηση δημιουργήθηκε με τη χρήση της JAVA SWING. Αποτελείται από μία απλή φόρμα για εισαγωγή των όρων αναζήτησης, το κουμπί Search που όταν πατηθεί κάνει την αναζήτηση βάση της εισαγωγής στη φόρμα και το πεδίο ή πεδία που έχει επιλεγθεί από το μενού επιλογών δίπλα “Search in”.

Τα διαθέσιμα πεδία είναι Title, abstract, content και all. Οι επιλογές “Title, abstract” κάνουν αναζήτηση μόνο στα αντίστοιχα πεδία, το “content” στο πεδίο με το κυρίως σώμα κειμένου άρθρων, ενώ το all συνδυαστικά σε όλα τα παραπάνω πεδία. Αυτή η αναζήτηση κάνει αυτόματα διόρθωση ορθογραφικών και δεν ψάχνει ακριβώς ταιριάσματα με τη βάση (fuzzy logic) σε αντίθεση με την αναζήτηση που γίνεται όταν πατηθεί το κουμπί “Exact Search” που θα αναζητήσει τον όρο που δόθηκε ακριβώς όπως είναι και σε αυτή τη σειρά λέξεων. Δηλαδή αν δοθεί στη φόρμα “new york” θα εμφανίσει μόνο άρθρα που περιέχουν “new york”, όχι μόνο “new” ή “york” ακόμα και αν υπάρχουν και οι δύο λέξεις αλλά μακριά.

Για το πέραςμα των queries και αναζήτηση στο λεξικό έγινε χρήση των πακέτου “lucene.search” και “lucene.queryparser”. Για τη διόρθωση των ορθογραφικών λαθών κατά την αναζήτηση έγινε χρήση του api search.spell.


Η πληροφορία για την αναζήτηση ενημερώνεται με κάθε νέα αναζήτηση αποθηκεύοντας τον όρο σε μία λίστα και παρουσιάζεται κάτω αριστερά στη καρτέλα “Search History”. Βάση αυτής παράγονται προτάσεις για αναζήτηση που εμφανίζονται από κάτω (δεν δουλεύει σωστά πάντα).



Εικόνα 1. Το παράθυρο της εφαρμογής αναζήτησης.

5. Παρουσίαση Αποτελεσμάτων.

Όπως φαίνεται στην [Εικόνα 1](#) δεξιά τα αποτελέσματα εμφανίζονται 10 ανά σελίδα με ταξινόμηση ανάλογα με τη σχετικότητα με το βάρος. Εμφανίζεται το id του εγγράφου με την χρονολογία και το Τίτλο και αν υπάρχει ο όρος κλειδί στη περίληψη του εγγράφου ή στο σώμα κειμένου εμφανίζεται απόσπασμα κειμένου γύρω από τον όρο στο σημείο που βρίσκεται στο άρθρο. Το επόμενο



αποτέλεσμα εμφανίζεται από κάτω και χωρίζεται από διακεκομμένη γραμμή. Για να δει επόμενα αποτελέσματα μετά τα πρώτα 10 απλά επιλέγει την επόμενη σελίδα πατώντας κλικ στο όνομά της (π.χ. Page 2). Οι λέξεις κλειδιά εμφανίζονται τονισμένες με κίτρινο, υπογράμμιση και πλάγια γράμματα. Επίσης υπάρχει κουμπί για αναδιάταξη των αποτελεσμάτων με βάση τη χρονιά του άρθρου (“Sort by year published”). Στο κάτω μέρος του παραθύρου εμφανίζεται ο συνολικός αριθμός των αποτελεσμάτων. Ο όρος αναζήτησης τονίζεται στο αποτέλεσμα με τη βοήθεια του api:”search.highlight” και η αναδιάταξη βάση της χρονιάς με το :”search.sort”.