Ανάκτηση Πληροφορίας

Ράμμος Θωμάς ΑΜ: 4583
Μητρόπουλος Γεώργιος ΑΜ:4733

Link στο GitHub repository: https://github.com/Thomas-Rammos/InformationRetrieval

Εισαγωγή

Lucene πρόγραμμα ειδικά σχεδιασμένο για αναζήτηση πληροφορίας από επιστημονικά άρθρα από CSV αρχείο. Το πρόγραμμα διαβάζει δεδομένα από επιστημονικά άρθρα από ένα αρχείο CSV, ευρετηριάζει τα δεδομένα αυτά με τη χρήση της Lucene βιβλιοθήκης και δίνει τη δυνατότητα στους χρήστες να αναζητήσουν τα ευρετηριασμένα αυτά δεδομένα χρησιμοποιώντας λέξεις-κλειδιά και αναζήτηση πεδίου δηλαδή μπορούν να φιλτράρουν την αναζήτησή τους με βάση πεδία όπως ο τίτλος, abstract και full text. Το σύστημα διατηρεί επίσης ένα ιστορικό των ερωτημάτων των χρηστών και παρέχει προτάσεις για μελλοντικά ερωτήματα.

Συλλογή

Για την δημιουργία της συλλογής (*corpus*) κατεβάσαμε μια έτοιμη συλλογή από το Kaggle: **All NeurIPS (NIPS) Papers.**

Από όλα τα δεδομένα της παραπάνω συλλογής, επιλέχθηκαν τυχαία 250 γραμμές που να έχουν συμπληρωμένα όλα τα πεδία ‘source_id’, ’year’, ’title’, ’abstract’ και ‘full_text’. Αυτά τα δεδομένα τα αποθηκεύσαμε σε ένα νέο αρχείο csv που θα είναι αυτό που θα επεξεργαστούμε.
Οι συγκεκριμένες ενέργειες έγιναν με τη χρήση ενός script σε python.

Ανάλυση κειμένου και κατασκευή ευρετηρίου

Το αρχείο CSV, το οποίο αποτελεί την πηγή δεδομένων έχει πέντε στήλες: ‘source_id’, ’year’, ’title’, ’abstract’ και ‘full_text’. Το σύστημα διαβάζει το αρχείο CSV γραμμή προς γραμμή και αναλύει αυτά τα πέντε πεδία. Κάθε άρθρο αντιμετωπίζεται ως έγγραφο και το αντίστοιχο ‘source_id’, ’year’, ’title’, ’abstract’ και ‘full_text’ ευρετηριάζονται στο Lucene χρησιμοποιώντας έναν StandardAnalyzer. Για την αποθήκευση του ευρετηρίου στο δίσκο χρησιμοποιείτε η συνάρτηση

FSDirectory.open() του Lucene API org.apache.lucene.store. Με την κλάση IndexWriter() του package org.apache.lucene.index δημιουργείτε ένα καινούργιο **index** στο οποίο θα αποθηκευθούν τα Documents αντικείμενα.

Πιο συγκεκριμένα, τα πεδία "source_id", "year" , "title", "abstract" και "full_text" προστίθενται στο έγγραφο ως πεδία TextField. Αυτά τα πεδία αποθηκεύουν τις πληροφορίες για το άρθρο και επιτρέπουν την αναζήτηση πλήρους κειμένου, τίτλου, year και abstract. Επιπλέον, το πεδίο "year" επιτρέπει λειτουργίες ταξινόμησης παρέχοντας αποτελεσματικές δυνατότητες ταξινόμησης και ομαδοποίησης.

Για την ανάλυση των πεδίων θα χρησιμοποιηθεί ο StandardAnalyzer() ο οποίος διαχωρίζει το κείμενο σε μεμονωμένους όρους, χρησιμοποιώντας ένα tokenizer που διαχωρίζει τα whitespace και τα σημεία στίξης, μετατρέπει όλους τους όρους σε lowercase, αφαιρεί τα stop words και παρέχει τη δυνατότητα του stemming. Αποτελεί μια λογική επιλογή όταν δεν γνωρίζουμε τα χαρακτηριστικά του κειμένου μας. Το API που χρησιμοποιήθηκε για αυτή την περίπτωση είναι το “org.apache.lucene.analysis”.

Το ευρετήριο κατασκευάζεται χρησιμοποιώντας πέντε πεδία (‘source_id’, ’year’, ’title’, ’abstract’ και ‘full_text’), καθένα από τα οποία είναι ένα TextField. Τα TextFields τόσο αναλύονται όσο και ευρετηριάζονται, πράγμα που σημαίνει ότι πριν από την ευρετηρίαση, γίνεται το tokenized και επίσης αποθηκεύονται στο ευρετήριο για ανάκτηση. Για την δημιουργία του **Document** θα χρησιμοποιήθεί το package της Lucene: org.apache.lucene.document και ύστερα θα προστεθεί στο ευρετήριο.

Αναζήτηση

Για τα ερωτήματα αναζήτησης, το σύστημα υποστηρίζει αναζήτηση με λέξεις-κλειδιά και αναζήτηση με βάση τα πεδία. Ο χρήστης μπορεί να κάνει αναζήτηση με λέξεις-κλειδιά (π.χ., ’title’, ’abstract’ , ’year’ και ‘full_text’).

Ένας MultiFieldQueryParser χρησιμοποιείται για την ανάλυση του ερωτήματος σε πολλαπλά πεδία. Αυτό επιτρέπει στο σύστημα να αναζητήσει το κείμενο του ερωτήματος σε όλα τα καθορισμένα πεδία του εγγράφου χρησιμοποιόντας το πακέτο “org.apache.lucene.queryparser.classic” της Lucene.Τα αποτελέσματα επιστρέφονται ανά δεκάδες με βάση τη συνάφειά τους με το ερώτημα χρησιμοποιώντας τον searcher της “org.apache.lucene.search”.Η συνάφεια υπολογίζεται με βάση διάφορους παράγοντες, όπως η συχνότητα όρων, η αντίστροφη συχνότητα εγγράφων, το μήκος πεδίου κ.λπ. σύμφωνα με τον προεπιλεγμένο αλγόριθμο ομοιότητας που χρησιμοποιεί το Lucene.

Ακόμα, το σύστημα αναζήτησης θα διατηρεί ιστορικό των ερωτημάτων του χρήστη. Κάθε νέο ερώτημα προστίθεται σε ένα ευρετήριο προηγούμενων ερωτημάτων και αυτό το ευρετήριο αποθηκεύεται στο δίσκο μέσω των API’s. Εάν ένα ερώτημα υπάρχει ήδη στο ευρετήριο δεν προστίθεται ξανά διασφαλίζοντας ότι το ιστορικό περιέχει μοναδικά ερωτήματα αναζήτησης. Αυτό το ευρετήριο ιστορικού χρησιμοποιείται για να παρέχει στους χρήστες προτάσεις αναζήτησης. Όταν ένας χρήστης εισάγει ένα μερικό ερώτημα (δηλ. όταν έχει πληκτρολογήσει μερικά

γράμματα του query που θέλει να αναζητήσει), το σύστημα χρησιμοποιεί την suggestQueries για να βρει προηγούμενα ερωτήματα που ξεκίνησαν με το ίδιο κείμενο επιστρέφοντας αποτελέσματα σε κάθε αναζήτηση στο ευρετήριο. Αυτά τα προηγούμενα ερωτήματα επιστρέφονται ως προτάσεις στον χρήστη. Αυτή δυνατότητα παρέχεται μέσω του “org.apache.lucene”.

Παρουσίαση αποτελεσμάτων

Για την παρουσίαση των αποτελεσμάτων χρησιμοποιήσαμε JavaFX. Συγκεκριμένα τα αποτελέσματα παρουσιάζονται σε ένα περιβάλλον που μοιάζει με ιστοσελίδα με τη βοήθεια ετικετών HTML. Κάθε συναφές έγγραφο ενός αποτελέσματος αναζήτησης παρουσιάζεται σε ξεχωριστές ετικέτες παραγράφου. Τα αποτελέσματα αναζήτησης είναι σελιδοποιημένα μέσω μιας μεταβλητής currentPage η οποία αρχικοποιείται στο 0. Μέσα από έναν έλεγχο περιορίζουμε μόνο 10 αποτελέσματα να εμφανίζονται σε μια σελίδα κάθε φορά.

Ο χρήστης μπορεί να επιλέξει να κάνει αναζήτηση σε συγκεκριμένα πεδία το ’title’, ’abstract’ ,’year’ και ‘full_text’ χρησιμοποιώντας επιλογές σε check boxes. Εάν επιλεγούν πολλαπλά πλαίσια ελέγχου, η αναζήτηση θα εκτελεστεί σε όλα αυτά τα πεδία με την ίδια λογική που αναφέρθηκε μέσω της MultiFieldQueryParser. Όταν εκτελείται μια αναζήτηση, το ερώτημα αναζήτησης προστίθεται στο ιστορικό αναζήτησης. Εάν τα ευρετήρια δεν υπάρχουν κατά την εκκίνηση της εφαρμογής, δημιουργούνται.

Ακόμα, χρησιμοποιείται ξανά η MultiFieldQueryParser κατά την διαδικασία της εναλλαγής των σελίδων μεταξύ των αποτελεσμάτων αναζήτησης καθώς σε κάθε σελίδα επιστρέφουμε μόνο 10 αποτελέσματα. Στην συγκεκριμένη περίπτωση χρησιμοποιείται ακόμα η showNextPage της “org.apache.lucene.search” η οποία επιστρέφει τα επόμενα 10 πιο συναφή έγγραφα βάση του κειμένου της αναζήτησης. Η σελιδοποίηση παρέχεται με τα κουμπιά "Previous" και "Next". Η εφαρμογή παρακολουθεί την τρέχουσα σελίδα και τα κουμπιά αυτά ενεργοποιούνται ή απενεργοποιούνται ανάλογα με τη σελίδα στην οποία βρίσκεται ο χρήστης.

Ο χρήστης έχει επίσης τη δυνατότητα να ταξινομήσει τα αποτελέσματα της αναζήτησης με φθίνουσα χρονολογική σειρά, εδώ λοιπόν φαίνεται και η δυνατότητα ομαδοποίησης. Αυτό επιτυγχάνεται ενεργοποιώντας ένα πλαίσιο ελέγχου με τίτλο "Year". Όταν επιλεγεί, τα αποτελέσματα αναζήτησης θα παρουσιάζονται με χρονολογική σειρά με βάση τα πεδία δίνοντας στους χρήστες μια τακτοποιημένη λίστα. Αυτό είναι ιδιαίτερα επωφελές για μεγαλύτερα σύνολα αποτελεσμάτων, καθώς απλοποιεί τον εντοπισμό συγκεκριμένων στοιχείων ενδιαφέροντος. Αυτή η λειτουργία ενεργοποιείται με τη χρήση της κλάσης "Sort" της "org.apache.lucene.search", η οποία επιτρέπει την ταξινόμηση των αποτελεσμάτων αναζήτησης σύμφωνα με διάφορα πεδία. Ένα αντικείμενο "Sort" δημιουργείται με αντικείμενα "SortField". Όταν αυτό το αντικείμενο "sort" περάσει στη μέθοδο αναζήτησης του αντικειμένου τα επιστρεφόμενα αποτελέσματα θα ταξινομηθούν σύμφωνα με τα πεδία που έχουν καθοριστεί.

Για την προβολή των προτάσεων εναλλακτικών ερωτημάτων οι προτάσεις αντλούνται από το ευρετήριο ιστορικού αναζήτησης αφού ο χρήστης σταματήσει να πληκτρολογεί στο πλαίσιο αναζήτησης. Τα αποτελέσματα, εμφανίζονται κάτω από την μπάρα αναζήτησης.

Εμφάνιση GUI

Η μέθοδος displayResults έχει σχεδιαστεί για να εμφανίζει τα αποτελέσματα της αναζήτησης σε μια διεπαφή χρήστη βασισμένη στο διαδίκτυο. Λειτουργεί με την κατασκευή μιας συμβολοσειράς HTML που αναπαριστά τα αποτελέσματα αναζήτησης και τα εμφανίζει σε ευανάγνωστη μορφή.

Αρχικά, πραγματοποιεί επανάληψη της λίστας των αντικειμένων Document στα αποτελέσματα αναζήτησης. Για κάθε έγγραφο εξάγει τα πεδία και η μέθοδος getHighlightedText() υπογραμμίζει όλες τις εμφανίσεις των λέξεων-κλειδιών αναζήτησης στην προεπισκόπηση των άρθρων περικλείοντάς τες με ετικέτες HTML span.

Στη συνέχεια, δημιουργεί μια κεφαλίδα HTML για κάθε αποτέλεσμα που περιέχει τον τίτλο και το abstract του άρθρου. Κάνοντας κλικ στον τίτλο ενεργοποιείται μια συνάρτηση JavaScript που εμφανίζει το full_text του άρθρου. Η συνάρτηση JavaScript showFullText() υλοποιείται στο τέλος της συμβολοσειράς HTML.

Η συμβολοσειρά HTML φορτώνεται σε ένα αντικείμενο WebEngine, το οποίο αποτελεί μέρος της υλοποίησης του προγράμματος περιήγησης JavaFX WebKit. Η JavaScript είναι ενεργοποιημένη στη μηχανή ιστού, ώστε η συνάρτηση showFullText() να μπορεί να αλληλεπιδράσει με τον κώδικα Java.

Ένα αντικείμενο δημιουργείται και προστίθεται στο JavaScript της μηχανής ιστού μετά την επιτυχή φόρτωση του περιεχομένου HTML. Αυτή η γέφυρα επιτρέπει στον κώδικα JavaScript να καλεί μεθόδους Java. Η συνάρτηση showFullText() χρησιμοποιείτε για να καλέσει τη μέθοδο JOptionPane.showMessageDialog για να εμφανίσει το πλήρες κείμενο ενός επιλεγμένου εγγράφου σε ένα παράθυρο διαλόγου με δυνατότητα κύλισης (scrolling). Αυτή η μέθοδος είναι υπεύθυνη για τη δημιουργία του **JTextArea** και του **JScrollPane** που περιέχει το πλήρες κείμενο.

Η μέθοδος JOptionPane.showMessageDialog έχει σχεδιαστεί για να εμφανίζει το πλήρης κείμενο ενός άρθρου σε ένα νέο παράθυρο. Ψάχνει στη λίστα των εγγράφων στα αποτελέσματα αναζήτησης για ένα έγγραφο που ταιριάζει με τον τίτλο του άρθρου. Μόλις βρεθεί ανακτά το πλήρες κείμενο από το έγγραφο και στη συνέχεια δημιουργεί ένα νέο (παράθυρο) JavaFX που περιέχει ετικέτες και πεδία κειμένου. Παρέχεται επίσης ένα κουμπί κλεισίματος για να μπορεί ο χρήστης να κλείσει το παράθυρο.