ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

GitHub LINK

https://github.com/spiroschal/Information-Retrieval.git

ΟΝΟΜΑΤΕΠΩΝΥΜΟ - ΑΜ

Σπυρίδων Χαλιδιάς – 4830

ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ

Εισαγωγή

Ο στόχος αυτού του project είναι να υλοποιήσουμε ένα σύστημα αναζήτησης πληροφορίας από επιστημονικά άρθρα. Μέσα από ένα μεγάλο σύνολο από επιστημονικά άρθρα θα μπορούμε να ψάχνουμε λέξεις-φράσεις που υπάρχουν σε αυτά τα δεδομένα. Είναι δηλαδή σαν ένας κλώνος μιας μηχανής αναζήτησης στο ίντερνετ, αλλά μόνο για επιστημονικά άρθρα και με κάποια ίδια ή και παραπάνω features, όπως είναι η ταξινόμηση κατά αύξουσα ή φθίνουσα σειρά με βάση την χρονολογία και την στοχευμένη αναζήτηση σε συγκεκριμένα πεδία.

Η λειτουργικότητα του συστήματος χωρίζεται σε 3 βασικούς τομής:

- **Data** [κώδικα γραμμένο σε Python με την χρήση της βιβλιοθήκης Pandas], για την συλλογή και την επεξεργασία των δεδομένων ώστε να έρθουν στην κατάλληλη μορφή για τα επόμενα βήματα
- **Back End** [κώδικα γραμμένο σε Java με την χρήση της βιβλιοθήκης Lucene], για την υλοποίηση ευρετηρίων για την υποστήριξη διαφορετικών τρόπων αναζήτησης
- **Front End** [κώδικα γραμμένο σε Java με την χρήση της βιβλιοθήκης JavaFX], για την παρουσίαση των αποτελεσμάτων με εύχρηστο και λειτουργικό τρόπο, κατανοητό για τον χρήστη

Συλλογή Εγγράφων (corpus)

Ένα από τα πρώτα και σημαντικά βήματα είναι η συλλογή των εγγράφων μας, όπου στην συγκεκριμένη εργασία τα έγγραφα που θέλουμε είναι επιστημονικά άρθρα.

Μια συλλογή τέτοιων άρθρων θα γίνει από το διαδίκτυο και πιο συγκεκριμένα από το site της σελίδας Kaggle, με link:

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

Τα 2 αρχεία που συναντάμε, είναι τύπου '.csv', και το ένα αφορά τα άρθρα έχοντας 5 στήλες(source_id, year, title, abstract, full_text) και το άλλο αφορά τους συγγραφής έχοντας 4 στήλες(source_id, first_name, last_name, institution), και συνδέονται μεταξύ τους αυτά τα 2 αρχεία με ένα κοινό κλειδί βάση των source_ids.

Χρησιμοποιήθηκε, μέσω της Python, η βιβλιοθήκη Pandas, όπου πήραμε τα 2 αυτά csv αρχεία που προ αναφέραμε. Παρακάτω θα δούμε τα βήματα που ακολουθήθηκαν ώστε να καταλήξουμε σε ένα csv αρχείο το οποίο θα είναι σε κατάλληλη μορφή για να περαστεί στο επόμενο βήμα, το οποίο είναι η κατασκευή ευρετηρίου.

Αρχικά, μια σημαντική προεργασία που πρέπει να γίνει, είναι να φιλτράρουμε/καθαρίσουμε κάποιες πλειάδες από τους πίνακές μας που κάποιες στήλες τους μπορεί να έχουν missing values. Αυτό πρέπει να γίνει γιατί θα μας προκαλέσει προβλήματα λόγω του ότι θα χαλάει τα αποτελέσματά μας επειδή θα λείπει πληροφορία. Έπειτα, παρατηρήθηκε ένα σημαντικό πρόβλημα που υπήρχε στα δεδομένα που πήραμε από το Kaggle, το οποίο είναι ότι οι τιμές του πεδίου source_id στο papers.csv δεν είναι μοναδικές, όπως λογικά θα περιμέναμε. Αυτό δημιουργεί κάποια προβλήματα στην συνένωση των δύο δεδομένων μας βάζοντας περισσότερους συγγραφής σε ένα άρθρο από όσους πραγματικά είναι(στην ουσία τα άρθρα με το κοινό source_id θα έχουν τους ίδιους συνενωμένους συγγραφής). Για να λυθεί αυτό το πρόβλημα, κρατήθηκαν μόνο οι γραμμές που το source_id εμφανίστηκε μόνο μια φορά στον πίνακα των papers. Άρα τώρα μπορούμε να συνεχίσουμε και να κάνουμε με ασφάλεια την συνένωση των papers(με μοναδικό πλέον source_id) με τους συγγραφής, με βάση το source_id. Επίσης, για τον καλύτερο και πιο εύκολο χειρισμό, ενώνουμε σε μία κολώνα το first_name με το last_name, για να έχουμε κατευθείαν το ονοματεπώνυμό του συγγραφέα. Ένα από τα τελευταία βήματα είναι να ομαδοποιήσουμε, με βάση το source_id, το συνενωμένο μας dataFrame, για να μαζέψουμε όλους τους συγγραφής με τα institution τους σε μία γραμμή, που εν τέλη αυτό θα είναι και το παραχθέν μας αρχείο με όνομα corpus.csv, με 2029 επιστημονικά άρθρα, και πεδία source_id(δεν θα χρειαστεί αργότερα, είναι μόνο

για λόγους αποσφαλμάτωσης), full_name, institution, year, title, abstract, full_text.

Ανάλυση κειμένου και Κατασκευή ευρετηρίου

Αρχικά, καλό είναι να παραθέσουμε τα πεδία που θα χρειαστούμε και για την κατασκευή των ευρετηρίων. Τα πεδία είναι τα:

full_name, institution, year, title, abstract, full_text

Χρησιμοποιήθηκε, μέσω της Java, η βιβλιοθήκη Lucene, προκειμένου να μας βοηθήσει να αναλύσουμε το κείμενο και να κατασκευάσουμε το ευρετήριο. Πιο συγκεκριμένα, αρχικά, με τον FileReader και τον CSVParser επιτεύχθηκε η σωστή ανάγνωση του csv αρχείου που είχαμε φτιάξει στο προηγούμενο βήμα. Δοκιμάστηκαν και άλλοι τέτοιου είδους Parsers και διαφορετικά encodings για να φορτώσουμε με επιτυχία το csv αρχείο μας, αλλά λόγω του ότι στο πεδίο *full_text* είχε αρκετούς ειδικούς χαρακτήρες, προκαλούσε διάφορα προβλήματα στο διαχωρισμό των κολώνων. Έπειτα, χρησιμοποιήθηκε ο StandardAnalyzer της Lucene, ένας από τους πιο απλούς αναλυτές κειμένου αλλά είναι ότι χρειαζόμαστε για το συγκεκριμένο σύστημα. Κάποια βασικά χαρακτηριστικά που έχει και μας εξυπηρετούν τις ανάγκες μας, είναι το «Tokenization» δηλαδή την διάσπαση του κειμένου σε tokens, «Lowercasing» δηλαδή την μετατροπή όλων των χαρακτήρων σε πεζά, «Stop Words» δηλαδή την αφαίρεση μικρών και κοινών λέξεων που δεν έχουν πολύ σημασία στο νόημα του κειμένου, «Stemming» δηλαδή την επιστροφή των λέξεων στην βασική τους μορφή. Το ευρετήριο που φτιάχτηκε είναι τύπου ByteBuffersDirectory, που αυτό σημαίνει ότι αποθηκεύεται στην μνήμη, πράγμα που το καθιστά πολύ γρήγορο, ένα χαρακτηριστικό που είναι σημαντικό στην αναζήτηση. Το πρόβλημα/πλεονέκτημα είναι ότι δεν αποθηκεύει μόνιμα το ευρετήριο στον δίσκο, αλλά αυτό μας δίνει την δυνατότητα κάθε φορά που το τρέχουμε, να μας ανανεώνει τα δεδομένα μας αν τυχόν υπάρχουν αλλαγές. Για το τελευταίο βήμα που αφορά την κατασκευή του ευρετηρίου, είναι να περάσουμε σε αυτό δεδομένα. Αυτό γίνεται με την χρήση του IndexWriterConfig(μπορούσαν να γίνουν περισσότερες ρυθμίσεις στον κώδικα ώστε να έχουμε μεγαλύτερο έλεγχο για βέλτιστα αποτελέσματα στο σύστημά μας, αλλά για χάρη απλότητας δεν έγινε κάτι τέτοιο) που χρειάζεται τον αναλυτή που φτιάξαμε λίγο παραπάνω ο οποίος χρησιμοποιείτε με την σειρά του ως ο ρυθμιστείς του εγγραφέα του ευρετηρίου(IndexWriter) και εν τέλη γεμίζει το ευρετήριο με τα δεδομένα από το corpus, που στην περίπτωσή μας είναι το csv όπως είδαμε στην προηγούμενη ενότητα.

Όλη η εκτέλεση αυτής της διαδικασίας, για το συγκεκριμένο project και με τις συγκεκριμένες κλάσεις που πάρθηκαν, σε συνδυασμό με το συγκεκριμένο corpus που κατασκευάστηκε, διήρκησε περίπου 10 δευτερόλεπτα.

Αναζήτηση

Το πρόγραμμά μας πρέπει να υποστηρίζει 2*(+1 έξτρα)* διαφορετικά είδη αναζήτησης:

(α) αναζήτηση με λέξεις κλειδιά(keywords)

(β) αναζήτηση πεδίου [full_name, institution, year, title, abstract, full_text]

(γ) αναζήτηση με φράση(δηλαδή πολλά keywords μαζί). Αυτό είναι μία έξτρα λειτουργία, που μας επιτρέπει να θέσουμε προς αναζήτηση όχι μόνο μία λέξη, αλλά και περισσότερες, δηλαδή σαν μια φράση που ψάχνουμε, που αρχικά θα προσπαθήσει να την βρει αυτούσια, αλλά και να αποτύχει ή μη, θα βγάλει και άλλα αποτελέσματα, με σειρά όμως προτεραιότητας τα επιστημονικά άρθρα που είχαν περισσότερες κοινές λέξεις με την αρχική μας φράση προς αναζήτηση. Όσον αφορά το οπτικό κομμάτι, θα υπογραμμίσει μόνο ολόκληρη την φράση στην περίπτωση που υπάρχει, και αν δεν τύχει να την βρει ολόκληρη, δεν θα υπογραμμίσει τις απομένον κοινές λέξεις.

Αρχικά, για την υλοποίηση της αναζήτησης για την περίπτωση του (α), θα χρειαστούμε τον αναλυτή αναζήτησης MultiFieldQueryParser ο οποίος μπορεί να πάρει πολλά πεδία σαν όρισμα, το οποίο το εκμεταλλευόμαστε για να περάσουμε, ως παράμετρο, όλα μας τα πεδία και να μας αφορά μόνο το keyword που θα ορίσει ο χρήστης ως προς αναζήτηση. Για την περίπτωση αναζήτησης του (β), θα χρειαστούμε τον αναλυτή αναζήτησης QueryParser ο οποίος δέχεται ως όρισμα, πέρα ένα μόνο πεδίο και θα κάνει αναζήτηση μόνο στο συγκεκριμένο. Επιπλέον θα χρησιμοποιηθεί το DirectoryReader, που χρειάζεται για την ανάγνωση του ευραιτηρίου, αφού μας δίνει πρόσβαση στα δεδομένα και στις ευρύτερες πληροφορίες του ευρετηρίου, σε συνδυασμό με το IndexSearcher που κάνει εν τέλη την ερώτηση που θέλουμε για να ξεκινήσει η διαδικασία της αναζήτησης.

Επίσης θα έχει την ικανότητα να κρατάει ένα ιστορικό αναζητήσεων για να μπορεί να προσωποποιεί την αναζήτηση για τον κάθε χρήστη. Με αυτόν τον τρόπο ο χρήστης θα ενθυμείται και θα βλέπει παλιές του αναζητήσεις και είτε θα τις βρίσκει και θα τις πατάει είτε καθώς πληκτρολογεί θα γίνεται ταυτόχρονο φιλτράρισμα του ιστορικού με βάση το τι πάει να γράψει εκείνη την στιγμή.

Παρουσίαση Αποτελεσμάτων

Το τελευταίο βήμα είναι η οπτικοποιημένη απόδοση του project μας, με σκοπό να παίρνει κάποιες εισόδου από τον χρήστη και με βάση αυτά να εμφανίζει τα

επιθυμητά αποτελέσματα που του ζητάει ο χρήστης. Όλη η υλοποίηση έγινε με την χρήση της βιβλιοθήκης JavaFX.

Αρχικά μπορούμε να παρουσιάσουμε την βασική λειτουργικότητα της εφαρμογής, η οποία είναι ότι ο χρήστης βλέπει ένα text placeholder όπου θα μπορεί, σε μορφή κειμένου, να περάσει όποιον όρο θέλει. Από κάποια toggled buttons, ο χρήστης έχει την δυνατότητα να επιλέξει έναν από τους 1 συν 6 τρόπους αναζήτησης που έχουν αναλυθεί παραπάνω. Επίσης, έχει ένα button όπου όταν ο χρήστης έχει γράψει αυτό που θέλει, πατώντας το, του παράγει τα αποτελέσματα που επιθυμεί, δηλαδή το ή τα επιστημονικά άρθρα που ψάχνει.

Επίσης, υλοποιήθηκαν και κάποιες περαιτέρω λειτουργίες με σκοπό να βοηθήσουν στην οπτικοποίηση των αποτελεσμάτων. Τα αποτελέσματα της αναζήτησης παρουσιάζονται ανά 10 σε μία HTML περιοχή, όπου τα αποτελέσματα των πεδίων abstract και full_text, επειδή είναι μεγάλα σε όγκο πληροφορίας, έχουν μπει σε ένα scrollable «κουτί» περιορισμένων διαστάσεων για την καλύτερη οπτικοποίηση. Υπάρχει η δυνατότητα πλοήγησης, με την χρήση κουμπιών, σε επόμενες ή προηγούμενες σελίδες που περιέχουν τα επόμενα ή τα προηγούμενα επιστημονικά άρθρα αντίστοιχα. Οι όροι που αναζητήθηκαν είναι επισημασμένοι με κίτρινο χρώμα για να διευκολυνθεί ο εντοπισμό τους. Επιπλέον, ένα τελευταίο χαρακτηριστικό που υλοποιήθηκε, είναι ότι με το πάτημα ενός κουμπιού, ο χρήστης έχει την δυνατότητα αναδιάταξης της σειράς των άρθρων με βάση την χρονολογία, είτε από το πιο σύγχρονο προς το πιο παλιό, είτε το αντίθετο.

Εδώ παρουσιάζονται μόνο επιγραμματικά οι λειτουργίες και οι δυνατότητες του γραφικού περιβάλλουν της εφαρμογής που υλοποιήθηκαν. Εκτενέστερη και καλύτερη παρουσίαση των λειτουργιών γίνεται στο README_V2.md, που συμπεριλαμβάνει και εικόνες.

Προαιρετικό Τμήμα

Ως επιπρόσθετη λειτουργία, υλοποιήθηκε η δυνατότητα αναζήτησης ενός επιστημονικού άρθρου, και με βάση το όνομα του συγγραφέα αλλά μέχρι και με βάση του ινστιτούτων του κάθε συγγραφέα. Έχει προστεθεί κανονικά και επιπρόσθετο πεδίο για την κάθε περίπτωση, αλλά υπάρχει και η δυνατότητα αναζήτησης μέσω keywords , γράφοντας κανονικά όποιο όνομα/επώνυμο ή ινστιτούτο οποιουδήποτε συγγραφέα θέλει ο χρήστης.

Γενικά έχει γίνει λόγος κατά μήκος όλης αυτής της αναφοράς για αυτήν την επιπρόσθετη λειτουργεία, εξηγώντας τον τρόπο που υλοποιήθηκε και παρουσιάζοντας τους τρόπους που αντιμετωπίστηκαν τα προβλήματα που προέκυπταν.