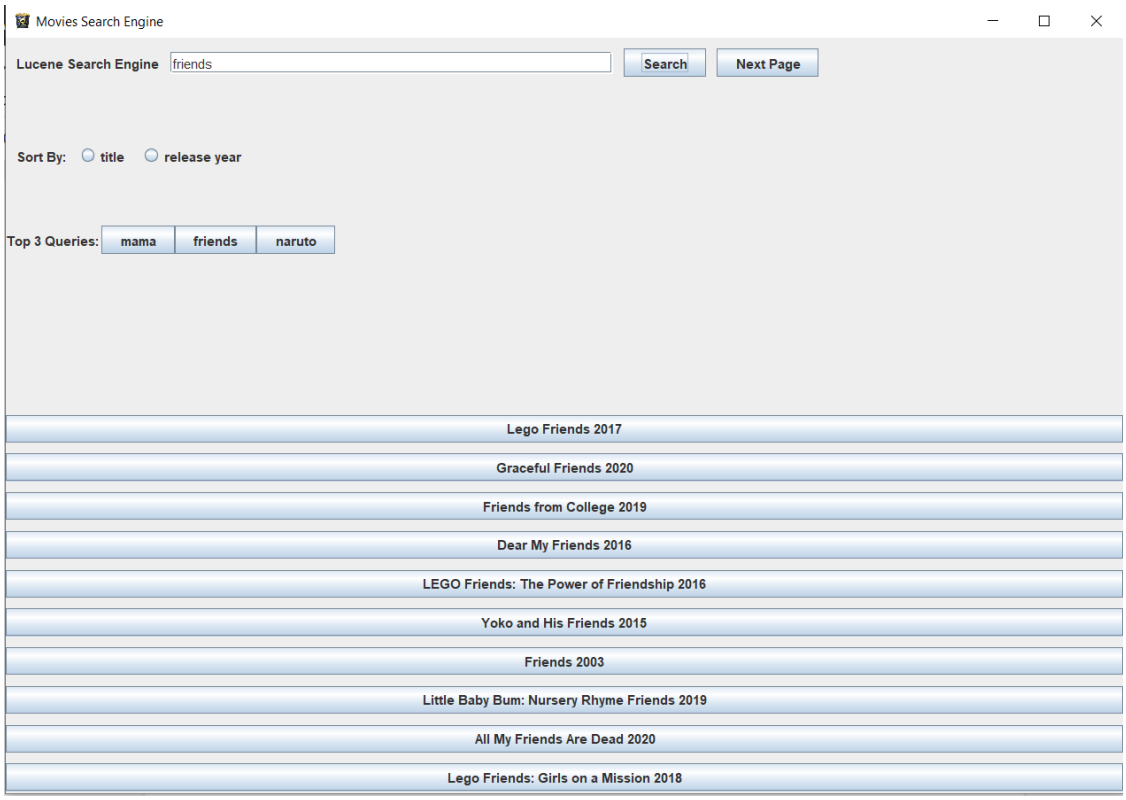
Εργασία: Μηχανή αναζήτησης ταινιών

Φάση 2

Github Link:https://github.com/filippospr/luceneProject



ΟΜΑΔΑ:
Φίλιππο Πρίφτης 4162
Άγκνες-Μοναλίσα Τουκαλιούκ 3346

Περιεχόμενα

1. Εισαγωγή

Η εργασία αφορά στο σχεδιασμό και υλοποίηση ενός συστήματος αναζήτησης ταινιών και σειρών, με την χρήση της βιβλιοθήκης Lucene(8.8.2) (https://lucene.apache.org/), μια βιβλιοθήκη ανοικτού κώδικα για την κατασκευή μηχανών αναζήτησης κειμένου. Για το σχεδιασμό του Graphical User Interface (GUI) χρησιμοποιήσαμε την Java Swing. Να σημειωθεί επίσης ότι για τον σχεδιασμό του συστήματος μας ακολουθήθηκε το design pattern MVC.

2. Συλλογή των ταινιών και σειρών

Η συλλογή των δεδομένων που χρησιμοποιήσαμε στην εργασία είναι μία έτοιμη συλλογή που υπάρχει στο kaggle(https://www.kaggle.com/datasets/shivamb/netflix-shows) και περιέχει δεδομένα για shows του netflix σε μορφή csv. Η συλλογή περιέχει 12 πεδία μερικά από τα οποία είναι τα εξής: **showid**(μοναδικό id για κάθε show), **type**(αν είναι ταινία ή σειρά), το **σκηνοθέτη**, το **cast**, την **χώρα** στην οποία γυρίστηκε, το **release year**, την διάρκεια, το **είδος** και την περιγραφή(**description**) της. Παρόλα αυτά εμείς κρατήσαμε τα 7 σημαντικότερα, κατά τη γνώμη μας, από τα 12 πεδία, τα οποία είναι τα εξής: *title, director, cast, country, release_year, listed_in, description*.

3. Ευρετοποίηση και Αναζήτηση

Το σύστημα μας αποτελείται από 3 packages: το **model**, το **view**, το **controller**.Στο **model** εμπεριέχονται οι κλάσεις: για την κατασκευή του ευρετηρίου, για την αναζήτηση στο ευρετήριο, την καταγραφή και την αναζήτηση του ιστορικού αναζήτησης και τον έλεγχο spelling στο ερώτημα που έχει θέσει ο χρήστης. Στο **view** εμπεριέχονται οι κλάσεις που είναι υπεύθυνες για το GUI. Τέλος το package **controller** εμπεριέχεται η κλάση controller που λειτουργεί ως “διαμεσολαβητής” μεταξύ του GUI και του backend(αναζητηση στο ευρετήριο κτλ).

Ευρετηριοποίηση κειμένου

Η ανάλυση κειμένου και η ευρετοποίηση γίνεται στη κλάση *FileIndexer* του model. Για την ανάλυση του κειμένου χρησιμοποιήθηκε ο Standard Analyzer ο οποιος αφαιρεί τα stop words και τα σημεία στίξης ενώ επιπρόσθετα μετατρέπει ολες τις λέξεις σε lowercase. Η ευρετηριοποίηση του κειμένου γίνεται μέσω της μεθόδου *createIndex()* του FileIndexer. Αρχικά στην *createIndex()* καλείται η μέθοδος *readAllDataAtOnce()* η οποία επιστρέφει μία λίστα από String πίνακες όπου κάθε πίνακας είναι μία γραμμή του αρχείου csv. Η κάθε γραμμή του csv αποτελεί τη μονάδα εγγράφου. Στη συνέχεια για κάθε γραμμή δημιουργούμε ένα document και προσθέτουμε σε αυτό τα 7 πεδία που αναφέραμε παραπάνω. Να σημειωθεί επιπρόσθετα οτι το ευρετήριο δημιουργείται μόνο μία φορά και αποθηκεύεται στη μνήμη. Αυτό επιτυγχάνεται μέσω της μεθόδου *createIndexIfNotExists()* η οποία ελέγχει αν υπάρχει αποθηκευμένο ευρετήριο στο φακελο index(στον οποίο αποθηκεύεται το ευρετήριο) και σε περίπτωση που δεν υπάρχει το δημιουργεί.

Αναζήτηση ευρετηρίου

Η αναζήτηση στο ευρετήριο γίνεται μέσω της κλάσης *FileSearcher* του model και συγκεκριμένα μέσω της μεθόδου *search()* η οποία δέχεται ως όρισμα ενα string που είναι το ερώτημα που έθεσε ο χρήστης και επιστρέφει ως αποτέλεσμα ένα arraylist απο documents που είναι σχετικά με το ερώτημα αυτό. Αρχικά προστίθεται στο ιστορικό αναζητήσεων(στο αρχείο SearchHistory.txt) το query που έθεσε ο χρήστης μέσω της μεθόδου *addToSearchHistory()*. Στη συνέχεια δημιουργούμε εναν query parser της κλάσης MultiFieldQueryParser με τον οποίο κάνουμε parse το ερώτημα που θέτει ο χρήστης. Ο λόγος για τον οποίο χρησιμοποιούμε το MultiFieldQueryParser είναι ότι κάνει αναζήτηση σε όλα τα πεδία ταυτόχρονα και είναι πιο αποδοτικός απο τον απλό query parser. Τέλος αφού αρχικοποιήσουμε σωστά τον IndexSearcher (ο οποίος κάνει την αναζήτηση) και καλέσουμε τη μέθοδο search μετατρέπουμε τα αποτελέσματα της αναζήτησης απο scoreDoc σε Document μέσω της μεθόδου *createsearchResultDocs()* και τα επιστρέφουμε στο χρήστη.

Σημείωση: Το ιστορικό αναζήτησης φορτώνεται μέσω της μεθόδου *loadSearchHistory()* κάθε φορά που δημιουργούμε ένα αντικείμενο της κλάσης FileSearcher.

Διόρθωση Τυπογραφικών Λαθών στα ερωτήματα του Χρήστη

Σε περίπτωση που ο χρήστης θέσει ένα ερώτημα το οποίο έχει ορθογραφικά λάθη και το σύστημα μας δεν εμφανίζει αποτελέσματα τότε προτείνονται στο χρήστη ερωτήματα αναζήτησης μέσω της κλάσης *QuerySpellChecker* του model. Συγκεκριμένα πρώτα πρέπει να δημιουργούμε το λεξικό (βασισμένο στο πεδίο title) το οποίο θα χρειαστεί ο spellChecker μέσω της μεθόδου *createSpellIndex()*. Αυτό δημιουργείται κάθε φορά που δημιουργούμε ένα αντικείμενο της κλάσης QuerySpellChecker. Η μέθοδος η οποία επιστρέφει τις προτάσεις για το query του χρήστη είναι η *suggestSearchQuery()* η οποία δέχεται ως όρισμα το query του χρήστη στην οποία αφού αρχικοποιήσουμε κατάλληλα το spellChecker επιστρέφουμε ως αποτέλεσμα τις 5 κοντινότερες με το ερώτημα λέξεις(τίτλους ταινιών) χρησιμοποιώντας την απόσταση του Levenshtein.

Χρήση ιστορικού αναζήτησης του χρήστη για την πρόταση εναλλακτικών ερωτημάτων

Το ιστορικό αναζήτησης που καταγράφεται αξιοποιείται από την κλάση *HistorySearcher* του model και συγκεκριμένα μέσω της μεθόδου *getTopThreeSearchQueries()* η οποία επιστρέφει τα 3 πιο δημοφιλή ερωτήματα που έχει θέσει ο χρήστης . Πρώτα δημιουργούμε ένα λεξικό με κλειδί τον όρο αναζήτησης που έθεσε ο χρήστης και τιμή το πλήθος των φορών που έθεσε αυτον τον όρο ο χρήστης διαβάζοντας το αρχείο searchHistory.txt ανα γραμμή(κάθε γραμμή στο αρχείο είναι ενα query που έθεσε ο χρήστης). Στη συνέχεια μετατρέπουμε το λεξικό σε λίστα από Entry και το ταξινομούμε(με αύξουσα σειρά) με βάση τα values του κάθε Entry και επιστρέφουμε τα 3 τελευταία στοιχεία της λίστας.

4. Παρουσίαση Αποτελεσμάτων στο χρήστη

Τα αποτελέσματα εμφανίζονται στο χρήστη ανα 10 και πατώντας το κουμπί Next Page/Previous Page) εμφανίζονται τα επόμενα/προηγούμενα 10 αποτελέσματα αντίστοιχα.

Διασύνδεση του package model με το view για την προβολή δεδομένων στο χρήστη

Τη διασύνδεση του package model με το view για την προβολή δεδομένων στο χρήστη την αναλαμβάνει η κλάση *controller.* Η κλάση αυτή καλεί την μέθοδο *search()* της κλάσης *FileSearcher* και αποκρύπτει απο τη κλάση *GUI* την ύπαρξη της κλάσης αυτης. Γενικότερα αυτή η κλάση αρχικοποιεί όλα τα αντικείμενα από τη κλάση *model* ώστε να χρησιμοποιηθούν από το την κλάση *GUI.* Επιπρόσθετα αυτή η κλάση ενημερώνει κατάλληλα το text στα κουμπιά που εμφανίζονται τα αποτελέσματα της αναζήτησης, ενημερώνει κατάλληλα τα ίδια κουμπιά οταν πατηθεί το previous η το next κ.α που θα αναλύσουμε στη συνέχεια. Τέλος ταξινομεί τα document που έχουν προκύψει από την αναζήτηση με βάση τη χρονιά η τον τίτλο(μέθοδοι sortByYear() και sortByTitle() αντίστοιχα).

Προβολή αποτελεσμάτων αναζήτησης στο χρήστη

Η προβολή των αποτελεσμάτων της αναζήτησης που πραγματοποίησε ο χρήστης γίνεται μέσω της κλάσης *GUI.* Σε αυτή τη κλάση αρχικοποιούνται όλα τα components του GUI και για τα περισσότερα απο αυτά και οι actionListeners τους. Αρχικά όταν ο χρήστης πατάει το κουμπί search καλείται η μέθοδος search του controller παίρνοντας ως όρισμα το query που εισήγαγε ο χρήστης στο textField. Αν δεν υπάρχουν αποτελέσματα(*controller.noResultsGiven()*) τότε εμφανίζουμε στον χρήστη προτάσεις αναζήτησης με βάση τις λέξεις απο τους τίτλους των ταινιών. Στην αντίθετη περίπτωση ενημερώνουμε κατάλληλα το text των κουμπιών στα οποία εμφανίζονται τα αποτελέσματα της αναζήτησης μέσω της μεθόδου setFirstSearchResultBtnsText() του cotroller. Με παρόμοια λογική ενημερώνεται και το text των κουμπιών αναζήτησης μέσω των μεθόδων *setNextSearchResultBtnsText()* και *setPreviousSearchResultBtnsText()* του controller. Όσον αφορά τα κουμπιά των suggestions(δηλαδή εκείνα που εμφανίζονται όταν δεν έχουν βρεθεί αποτελέσματα) και τα κουμπιά των top 3 queries εκτελούν ακριβώς την ίδια λειτουργία με το κουμπί search με τη μόνη διαφορά ότι χρησιμοποιείται ως query για το search το text του κουμπιού που πατάει ο χρήστης κάθε φορά. Τέλος όταν ο χρήστης πατήσει κάποιο από τα 2 κουμπιά για ταξινόμηση είτε με βάση το title είτε με βάση το release year καλείται η αντίστοιχη συνάρτηση sortBy του controller και τα αποτελέσματα ταξινομούνται κατάλληλα.

Προβολή πληροφοριών για την κάθε ταινία από τα αποτελέσματα αναζήτησης

Στην περίπτωση που ο χρήστης επιθυμεί να δει περαιτέρω πληροφορίες για κάποια από τις 10(ή λιγότερες ανάλογα με το πλήθος των αποτελεσμάτων) ταινίες που έχουν εμφανιστεί ως αποτέλεσμα αρκεί να κάνει κλικ στο κουμπί που εμφανίζεται ο τίτλος και το έτος και ένα νέο modal παράθυρο με τα περιεχόμενα της ταινίας εμφανίζεται. Το παράθυρο που εμφανίζεται είναι της κλάσης movieInfoDialog και το περιεχόμενο του καθορίζεται από τον controller κάθε φορά που ο χρήστης πατάει είτε το κουμπί search είτε το κουμπί Next Page είτε το κουμπί Previous Page(μέθοδος *getPageResults()*).