ΜΥΕ003 - Ανάκτηση Πληροφορίας | Φάση 2η

Μύρων Κουφόπουλος ΑΜ 4398
Γρηγόρης Πούλος ΑΜ 4480

GitHub Repository: https://github.com/g-poulos/search-engine-lucene

Περιεχόμενα

1. Εισαγωγή

Στόχος της μηχανής αναζήτησης που δημιουργήσαμε είναι να κάνει αναζήτηση ερωτημάτων ενός χρήστη σε μια συλλογή. Η αναζήτηση αυτή γίνεται μέσω λέξεων κλειδιά και η συλλογή αποτελείται από τραγούδια. Η μηχανή αναζήτησης επιστρέφει τα αποτελέσματα με βάση την συνάφεια τους με το ερώτημα (ranking).

Το πρόγραμμά μας χρησιμοποιεί την βιβλιοθήκη Lucene της Java η οποία παρέχει δυνατότητες όπως δημιουργία ευρετηρίων και εκτέλεσης ερωτημάτων σε αυτό. Για την παρουσίαση των αποτελεσμάτων χρησιμοποιήσαμε τη βιβλιοθήκη JavaFX για την κατασκευή του γραφικού περιβάλλοντος (GUI).

2. Συλλογή (Corpus)

Η συλλογή δεδομένων που χρησιμοποιήσαμε προέρχεται από την ιστοσελίδα Kaggle. Ονομάζεται Spotify Million Song Dataset και αποτελείται από τραγούδια. Η συλλογή έχει τη μορφή csv αρχείου και αποτελείται από 4 στήλες οι οποίες περιέχουν ως δεδομένα τα ονόματα καλλιτεχνών, τα ονόματα τραγουδιών, το link για κάθε τραγούδι και τους στίχους του κάθε τραγουδιού. Πιο συγκεκριμένα, περιέχει 643 καλλιτέχνες και 44824 τραγούδια.

Στην άσκηση που υλοποιήσαμε αφαιρέσαμε την στήλη με το link του τραγουδιού καθώς δεν μας χρησιμεύει. Επίσης, αφαιρέσαμε τις γραμμές οι οποίες σε τουλάχιστον μία στήλη περιείχαν NULL τιμή. Άρα τελικά τα πεδία της συλλογής που χρησιμοποιούμε είναι το artist, song και lyrics (text).

3. Ανάλυση κειμένου και κατασκευή ευρετηρίου

Για την κατασκευή του ευρετηρίου και την ανάλυση του κειμένου δημιουργήθηκε η κλάση (IndexCreator). Η συγκεκριμένη υλοποίηση του ευρετηρίου σε ξεχωριστή κλάση μας βοηθάει ώστε να μπορούμε να αποθηκεύσουμε το ευρετήριο στο δίσκο και να μην δημιουργείται εκ νέου με την εκκίνηση της εφαρμογής.

3.1 Προεπεξεργασία για την δημιουργία του εγγράφου

Για την προεπεξεργασία των δεδομένων διαβάζουμε όλο το αρχείο και ξεχωρίζουμε τα δεδομένα με βάση την στήλη και την γραμμή που βρίσκονται. Πιο συγκεκριμένα, δημιουργούμε μια λίστα για την κάθε ξεχωριστή γραμμή των δεδομένων. Εσωτερικά στην λίστα ξεχωρίζουμε τα τρία πεδία που έχουμε, δηλαδή artist, song και lyrics. Τα πεδία χωρίζονται μεταξύ τους με το χαρακτήρα ‘,’.

Η ανάλυση γίνεται με τη χρήση του StandardAnalyzer, ο οποίος παρέχεται από την Lucene. Βοηθάει στον διαχωρισμό του κειμένου σε tokens όπου το κάθε token αποτελείται από χαρακτήρες. Το κάθε token μετατρέπεται σε lowercase ώστε να μην υπάρχουν προβλήματα στην αναζήτηση κεφαλαίων γραμμάτων. Επίσης, αφαιρεί τα stop words δηλαδή κάποιες πολύ συχνά εμφανιζόμενες λέξεις οι οποίες δεν

προσφέρουν σημαντικό νόημα στο κείμενο (πχ ‘an’,’the’). Τέλος, αφαιρεί σημεία στίξης και προσφέρει μια μορφή stemming στο κείμενο.

3.2 Ανάλυση και ευρετήριο

Η δημιουργία του ευρετηρίου γίνεται στην μέθοδο createIndex στην κλάση IndexCreator. Αρχικά, της δίνουμε το path στο οποίο θα αποθηκεύσει το ευρετήριο και ανοίγει αυτό το directory μέσω του FSDirectory. Επίσης, δημιουργούμε ένα αντικείμενο IndexWriterConfig χρησιμοποιώντας τον analyzer που έχουμε δημιουργήσει. Στο directory που βρισκόμαστε με τον analyzer που έχουμε, δημιουργούμε έναν IndexWriter ο οποίος δημιουργεί και διατηρεί τα Indexes.

Στη συνέχεια, δημιουργούμε για κάθε γραμμή του αρχικού αρχείου ένα αντικείμενο Document στο οποίο αποθηκεύουμε τέσσερα πεδία (fields): artist, song, lyrics (text) και all. Το πεδίο all το χρησιμοποιούμε όταν ο χρήστης θέλει να κάνει αναζήτηση και στα 3 πεδία (artist, song, lyrics) ταυτόχρονα. Για όλα τα πεδία χρησιμοποιούμε την κλάση TextField της Lucene για να μπορούν τα Documents που δημιουργήσαμε να αναπαρασταθούν σαν text data, να γίνουν indexed, και να μπορούμε να εκτελέσουμε ερωτήματα με βάση αυτά.

4. Αναζήτηση

Για την αναζήτηση στο ευρετήριο που δημιουργήσαμε, κατασκευάστηκε η κλάση MainReader. Στα πεδία της κλάσης αποθηκεύονται τα αποτελέσματα της αναζήτησης αλλά και άλλες σχετικές πληροφορίες έτσι ώστε να μπορούμε να τα χρησιμοποιήσουμε εύκολα στην συνέχεια. Παρακάτω περιγράφονται αναλυτικότερα οι λειτουργίες της κλάσης MainReader αλλά και τα είδη ερωτημάτων που υποστηρίζει η εφαρμογή.

4.1 Είδη ερωτήσεων που υποστηρίζονται

- key-word queries : αναζήτηση με βάση κάποιες συγκεκριμένες λέξεις (π.χ. holy)
- boolean queries : αναζήτηση ερωτημάτων σε συνδυασμό λέξεων και λογικών τελεστών όπως το AND, OR και NOT.
  (π.χ. love AND cry) να περιέχονται ταυτόχρονα και οι δύο λέξεις
  (π.χ. love OR cry) να περιέχεται τουλάχιστον μία από τις δύο λέξεις
  (π.χ. love NOT cry) να περιέχεται η λέξη love όχι όμως η cry, για την χρήση του NOT θα πρέπει να συμπληρωθεί τουλάχιστον μία λέξη για αναζήτηση μπροστά.
- wildcard queries : αναζήτηση με λέξεις κλειδιά συνδυάζοντάς και ειδικούς χαρακτήρες όπως ?,*,+,-,~ (fuzzy character)
  - (π.χ. ho?y): αναζητα λέξεις μήκους 4 όπου περιέχουν στην θέση του ? οποιονδήποτε μονό χαρακτήρα, το ερωτηματικό μπορεί να συμπληρωθεί σε οποιαδήποτε θέση.
  - (π.χ. ho*y): αναζητά λέξεις που αρχίζουν με το πρόθεμα ho και τελειώνουν με το χαρακτήρα y. Το * υποδηλώνει πως ανάμεσα από αυτούς του χαρακτήρες μπορούμε να συμπληρώσουμε άπειρους

χαρακτήρες. Αντίστοιχα με το ? μπορεί να χρησιμοποιηθεί σε όλα τα σημεία μιας λέξης.
    - (π.χ. +love +cry) να περιέχονται και οι δύο λέξεις λειτουργεί σαν το AND
    - (π.χ. +love -cry) να περιέχεται η λέξη love και να μην περιέχεται η λέξη cry, όπως το NOT.
    - (π.χ. love~) να αναζητά λέξη παρόμοια με το love
    - (π.χ. love~N) να αναζητά λέξη με το πολύ N διαφορετικούς χαρακτήρες από το love (όπου N = ακέραιος θετικός αριθμός)
- field queries : να μπορεί ο χρήστης να αναζητά τα παραπάνω ερωτήματα σε όλα τα πεδία ή σε καθένα ξεχωριστά. Αυτό μπορεί να το κάνει μέσω του GUI με την επιλογή ενός εκ των τεσσάρων επιλογών που παρέχονται (artist, song, lyrics, all)

4.2 Δημιουργία και εκτέλεση ερωτήματος

Η είσοδος που παρέχει ο χρήστης κατά την εκτέλεση της εφαρμογής επεξεργάζεται απο την μέθοδο runQuery της MainReader. Σε αυτή χρησιμοποιώντας τον Standard Analyzer και τον QueryParser δημιουργούμε ένα αντικείμενο Query το οποίο θα χρησιμοποιηθεί για την αναζήτηση στο ευρετήριο. Μέσω του QueryParser δίνεται ακόμα η δυνατότητα επιλογής του πεδίου αναζήτησης. Με αυτό τον τρόπο κατα την εκτέλεση, ο χρήστης μπορεί να επιλέξει να αναζητήσει σε ένα από τα πεδία ή σε όλα ταυτόχρονα.

Μετά την δημιουργία του Query με την χρήση του IndexReader και IndexSearcher πραγματοποιείται η πρόσβαση και ανάγνωση του ευρετηρίου. Αφού ανακτηθούν τα σχετικά έγγραφα από το ευρετήριο χρησιμοποιείται ένα αντικείμενο της κλάσης TopDocs για την βαθμολόγηση τους έτσι ώστε να αποθηκευτούν σε σειρά από το περισσότερο συναφές στο λιγότερο. Η αποθήκευση τους γίνεται μέσω της μεθόδου fillFoundDocuments στην MainReader όπου κρατάει τα έγγραφα με την σειρά που αναφέρθηκε στο πεδίο της foundDocuments.

4.3 Τονισμός λέξεων κλειδιών

Για τον τονισμό των λέξεων χρησιμοποιούμε τον Highlighter που παρέχει η βιβλιοθήκη της Lucene. Για αυτή την λειτουργία έχουμε δημιουργήσει την μέθοδο highlightedHTMLResult στην κλάση MainReader η οποία τονίζει τις λέξεις κλειδιά και μετατρέπει τα έγγραφα του αποτελέσματος σε μορφή HTML για να είναι ευκολότερο να παρουσιαστούν.

Εντός της μεθόδου αρχικοποιείται ένα αντικείμενο της κλάσης SimpleHTMLFormatter και ένα της κλάσης QueryScorer που είναι απαραίτητα για την λειτουργία του Highlighter. Το πρώτο μετατρέπει σε έντονο κείμενο λέξεις των εγγράφων που περιέχονται στο ερώτημα που εκτελέστηκε. Το δεύτερο παρέχει πληροφορία σχετικά με το ποιοί όροι θα πρέπει να τονιστούν με βάση την συνάφια τους με το ερώτημα. Στην συνέχεια μπορούμε να πλέον να ορίσουμε τον Highlighter δίνοντας του ώς παραμέτρους τα δύο αντικείμενα που ορίσαμε. Επιπρόσθετα θέτουμε στον Highlighter το μέγεθος Fragment χρησιμοποιώντας τον SimpleFragmenter. Αυτό είναι το κομμάτι

κειμένου που θα επιστρέψει στο τέλος της εκτέλεσης του και επιλέγεται ανάλογα με το πόσους όρους περιέχει από το ερώτημα που εκτελέστηκε και πόσες φορές.

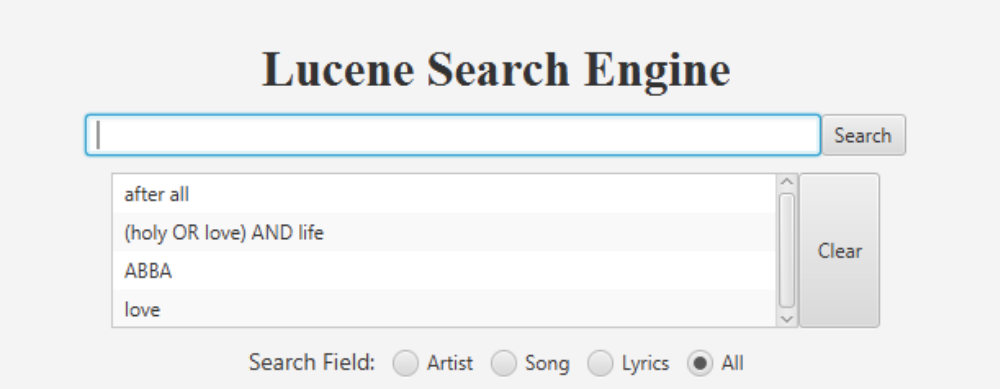
Με παρόμοιο τρόπο το κείμενο του πεδίου που θα επιλεχθεί επεξεργάζεται σε κομμάτια (tokens). Για τον λόγο αυτό χρησιμοποιούμε ένα αντικείμενο TokenStream της Lucene που χωρίζει το κείμενο συμφωνα με κάποιον analyzer, στην δική μας υλοποίηση τον StandardAnalyzer. Στην συνέχεια έχοντας δημιουργήσει το τονισμένο κείμενο του πεδίου αναζήτησης κατασκευάζουμε την HTML μορφή του εγγράφου ως ένα αντικείμενο StringBuilder. Όλα τα αντικείμενα StringBuilder που θα δημιουργηθούν για τα έγγραφα του αποτελέσματος αποθηκεύονται στο πεδίο της κλάσης MainReader htmlDocuments.

4.4 Χωρισμός αποτελεσμάτων σε σελίδες

Τα αποτελέσματα των ερωτημάτων εμφανίζονται στην εφαρμογή ανά 10. Για την υλοποίηση αυτής της λειτουργίας δημιουργήσαμε την μέθοδο createPages στην κλάση MainReader. Η μέθοδος διατρέχει τα αποτελέσματα της αναζήτησης στο πεδίο htmlDocuments και δημιουργεί λίστες με έγγραφα μήκους 10. Κάθε σελίδα που δημιουργείται αποθηκεύεται σε μία λίστα στο πεδίο htmlPages το οποίο μπορεί στην συνέχεια να προσπελαστεί και να χρησιμοποιηθεί στην προβολή.

4.5 Διατήρηση ιστορικού αναζητήσεων

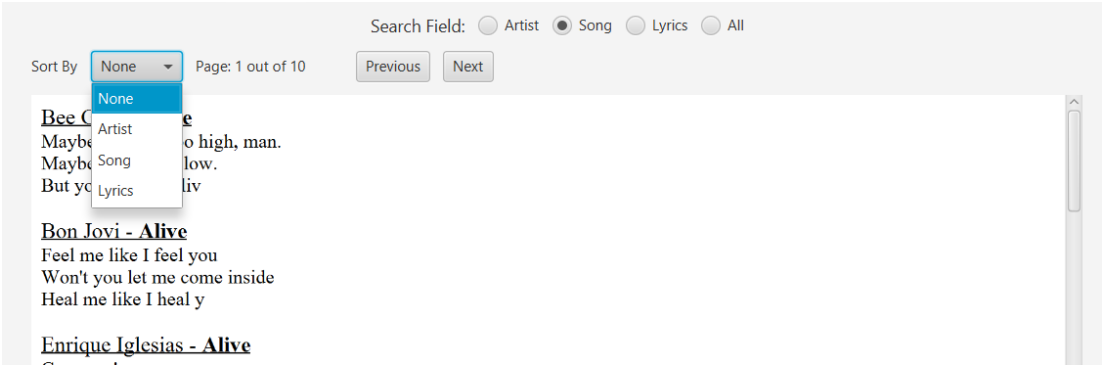
Όλες οι αναζητήσεις που πραγματοποιεί ο χρήστης αποθηκεύονται στην μνήμη κρατόντας έτσι το ιστορικό των αναζητήσεων. Στην συνέχεια, αυτές οι εγγραφές χρησιμοποιούνται έτσι ώστε κατά την πληκτρολόγηση του ερωτήματος να εμφανίζονται προτεινόμενα παρόμοια ερωτήματα τα οποία έχουν εκτελεστεί νωρίτερα. Η λίστα που εμφανίζεται είναι σε χρονολογική σειρά από το πιο νέο στο παλαιότερο ερώτημα και όσο ο χρήστης πληκτρολογεί εμφανίζονται μόνο οι λέξεις που μοιάζουν περισσότερο με αυτό που έχει πληκτρολογηθεί. Σε περίπτωση που ο χρήστης δεν έχει πληκτρολογήσει κάποιο ερώτημα εμφανίζονται όλα τα προηγούμενα όπως φαίνεται στην Εικόνα 4.5.1.



Εικόνα 4.5.1: Προτάσεις ερωτημάτων από ιστορικό

4.6 Ομαδοποίηση αποτελεσμάτων

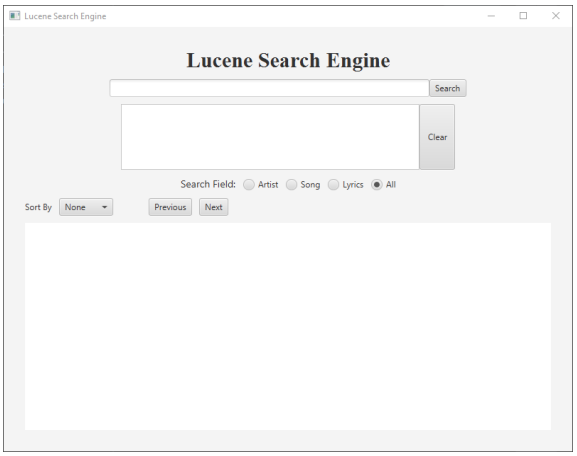
Η εφαρμογή δίνει ακόμα την δυνατότητα ομαδοποίησης των αποτελεσμάτων σύμφωνα με κάποιο πεδίο. Συγκεκριμένα, μετά την εμφάνιση των αποτελεσμάτων ο χρήστης μπορεί να να επιλέξει ένα από τα πεδία Artist, Song και Lyrics (Εικόνα 4.6.1) και το πρόγραμμα θα εμφανίσει τα αποτελέσματα σε αλφαβητική σειρά σύμφωνα με αυτό το πεδίο. Η λειτουργία αυτή υλοποιείται στην μέθοδο sortDocumentsBy στην κλάση MainReader και παιρνει ως όρισμα το όνομα του πεδίου που θέλουμε να γίνει η ομαδοποίηση. Η μέθοδος αρχικά κρατάει αντίγραφο των αποτελεσμάτων που έχουν βρεθεί και στην συνέχεια ταξινομεί τα αποτελέσματα με την μέθοδο sort που παρέχει η βιβλιοθήκη της Java Collections. Στην συνέχεια στα αποτελέσματα εφαρμόζεται ο τονισμός των λέξεων και ο χωρισμός σελίδων για την παρουσίαση τους. Τέλος, εάν απενεργοποιήσουμε την ομαδοποίηση (επιλέγοντας σαν πεδίο None) τα αποτελέσματα ανακτώνται από το αντίγραφο που είχε δημιουργηθεί και εμφανίζονται μη ταξινομημένα.



Εικόνα 4.6.1: Επιλογή ομαδοποίησης αποτελεσμάτων

5. Παρουσίαση Αποτελεσμάτων

Για την παρουσίαση των αποτελεσμάτων δημιουργήθηκε το γραφικό περιβάλλον που φαίνεται στην Εικόνα 5.1. Στα παρακάτω κεφάλαια περιγράφονται οι λειτουργίες του και ο συνδυασμός των αντικειμένων της βιβλιοθήκης JavaFX για την κατασκευή του.



Εικόνα 5.1: Γραφικό περιβάλλον εφαρμογής

5.1 Τρόπος λειτουργίας GUI

- Ονομασία μηχανής αναζήτησης : Lucene Search Engine
- Λευκή μπάρα αναζήτησης : στο σημείο αυτό ο χρήστης συμπληρώνει την λέξη-λέξεις που θέλει να αναζητήσει.
- Search button : κουμπί αναζήτησης στο ευρετήριο.
- Λευκή μπάρα (history) : στο σημείο αυτό εμφανίζονται οι προηγούμενες αναζητήσεις του χρήστη. Όσο συμπληρώνει το επόμενο ερώτημα αν έχει ήδη ψάξει κάτι παρόμοιο θα εμφανιστεί μία πρόταση για αναζήτηση.
- Clear (history) : διαγράφει το ιστορικό του χρήστη.
- Search field : ο χρήστης επιλέγει σε ποιο πεδίο θέλει να κάνει αναζήτηση, by default δίνουμε την επιλογή all δηλαδή σε όλα τα πεδία.
- Sort By : άμα θέλει ο χρήστης να ομαδοποιήσει τα αποτελέσματα τα οποία πήρε από την αναζήτηση. Η ομαδοποίηση είναι αλφαβητική και δίνεται η δυνατότητα να την εκτελέσει σε διαφορετικό πεδίο από αυτό που έχει κάνει την αναζήτηση.
- Page 1-10 : μετά από την πρώτη αναζήτηση ο χρήστης μπορεί να δει σε ποια σελίδα των αποτελεσμάτων βρίσκεται. Μπορεί να αλλάξει σελίδα πηγαίνοντας στην επόμενη ή στην προηγούμενη. Όταν φτάσει στην τελευταία για διευκόλυνση το Next βάλαμε να τον μεταφέρει πίσω στην πρώτη, αντίστοιχα για το previous.
- Λευκή μπάρα (results) : στο μέρος αυτό ο χρήστης μπορεί να δει τα αποτελέσματα της αναζήτησης που έκανε και να περιηγηθεί μέσω της ροδέλας πιο πάνω ή πιο κάτω. Τα αποτελέσματα εμφανίζονται ανά 10 σε κάθε σελίδα και σε bolt μπορεί να εντοπίσει τη λέξη-λέξεις που αναζήτησε. Η ομαδοποίηση γίνεται ανάλογα με το πλήθος εμφανίσεων της λέξης.
  Σημείωση :
  α) όταν αναζητάμε σε όλα τα fields μια λέξη τότε εμφανίζουμε όλους τους στίχους του τραγουδιού μαζί με τον τίτλο. Γίνεται bolt η λέξη όπου εμφανίζεται.
  β) όταν αναζητάμε σε κάποιο συγκεκριμένο field κάποια λέξη τότε εμφανίζουμε το τραγουδιστή, το τίτλο του τραγουδιού και ένα μέρος από τα lyrics. Γίνεται bolt η λέξη που ΄ψαχνουμε μόνο στο πεδίο που την χρησιμοποιούμε.

5.2 Υλοποίηση Interface

Για την υλοποίηση του GUI (Graphical User Interface) χρησιμοποιήσαμε την βιβλιοθήκη JavaFX. Πιο συγκεκριμένα, δημιουργήσαμε ένα αντικείμενο Scene το οποίο περιέχει ένα αντικείμενο VBox(). Το πρώτο ορίζει το παράθυρο που θα εμφανιστεί, ενώ το δεύτερο δέχεται άλλα αντικείμενα της βιβλιοθήκης JavaFX τα οποία τοποθετεί κατακόρυφα το ένα κάτω από το άλλο. Έτσι, αποθηκεύουμε με κατακόρυφη σειρά όσα κουμπιά θέλουμε να προσθέσουμε στο GUI.

Στη συνέχεια, δημιουργούμε HBox αντικείμενα τα οποία τοποθετούν αντικείμενα οριζόντια το ένα δίπλα από το άλλο. Για κάθε οριζόντια γραμμή προσθέτουμε κουμπιά, μπάρα συμπλήρωσης ή μπάρα αποτελεσμάτων που θέλουμε να εμφανίσουμε στο GUI. Για την εμφάνιση του ιστορικού χρησιμοποιούμε ListView αντικείμενα τα οποία παρουσιάζουν στοιχεία σε λίστα. Τέλος, για το κομμάτι του Interface που

παρουσιάζονται τα αποτελέσματα έχουμε χρησιμοποιήσει ένα αντικείμενο WebView το οποίο μπορεί να διαβάσει ετικέτες, το κείμενο HTML και να το παρουσιάσει.

Για όλα τα στοιχεία του Interface που πραγματοποιούν κάποια λειτουργία έχουμε δημιουργηθεί EventHandlers που εκτελούν την αντίστοιχη μέθοδο.

6. Επέκταση Μηχανής Αναζήτησης

6.1 Δημιουργία ευρετηρίου embedding

Για την βέλτιστη χρήση των διανυσματικών αναπαραστάσεων των λέξεων δημιουργήσαμε ένα επιπλέον ευρετήριο. Η ανάγνωση των δεδομένων του μοντέλου και η παραγωγή του ευρετηρίου πραγματοποιείται στην κλαση NLPIndexCreator.

Πιο συγκεκριμένα, κατασκευάζουμε ένα ευρετήριο το οποίο αποτελείται από έγγραφα δύο πεδίων, “word” και “vec”. Στο πρώτο αποθηκεύεται η λέξη και στο δεύτερο η διανυσματική της αναπαράσταση σε μορφή String. Κατά την δημιουργία του ευρετηρίου λέξεις που δεν υπάρχουν στο σύνολο των δεδομένων με τα τραγούδια, αγνοούνται έτσι ώστε το τελικό ευρετήριο να μην περιέχει λέξεις που δεν θα χρησιμοποιηθούν. Με αυτό το τρόπο, αναζητώντας μια λέξη μπορούμε αποτελεσματικά να ανακτήσουμε την διανυσματική της αναπαράσταση.

Θα θέλαμε να επισημάνουμε, πως επειδή τα δεδομένα του embedding είναι λίγα σε όγκο το ευρετήριο αποθηκεύεται στην μνήμη για ταχύτερη προσπέλαση χρησιμοποιώντας αντικείμενο της κλάσης ByteBuffersDirectory της Lucene. Η κατασκευή του ευρετηρίου γίνεται πριν την έναρξη της εφαρμογής.

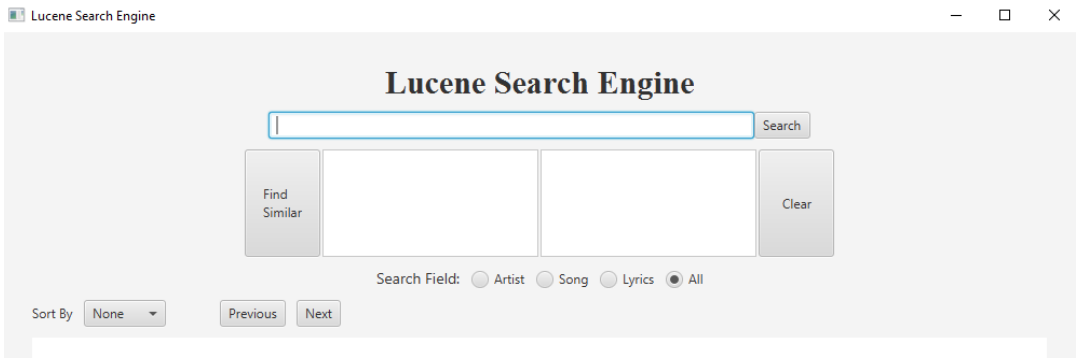
6.2 Επεξεργασία διανυσματικής αναπαράστασης ερωτήματος

Η χρήση του ευρετηρίου με τις διανυσματικές αναπαραστάσεις στην μηχανή αναζήτησης γίνεται μέσω της κλάσης NLPSearcher. Σε αυτή βασικό ρόλο έχει η μέθοδος searchSuggestions που για την είσοδο που έδωσε ο χρήστης επιστρέφει παρόμοιες σημασιολογικά λέξεις που μπορεί να αναζητήσει. Για την λειτουργία αυτή, αρχικά ανακτάται από το ευρετήριο του embedding το διάνυσμα του ερωτήματος που έδωσε ο χρήστης.

Στην συνέχεια διατρέχονται μια-μια οι εγγραφές του ευρετηρίου υπολογίζοντας το cosine similarity τους με το διάνυσμα του ερωτήματος. Οι λέξεις που έχουν ομοιότητα μεγαλύτερη από μία τιμή, που ορίζουμε ως SIMILARITY THRESHOLD, προστίθενται σε αυτές που θα επιστραφούν. Όταν οι εγγραφές τελειώσουν ή όταν συμπληρώσουμε έναν αριθμό λέξεων που ορίζουμε ώς SIMILAR WORD COUNT η διαδικασία τερματίζεται. Οι δύο παράμετροι που αναφέρθηκαν μπορούν να ρυθμιστούν έτσι ώστε να έχουμε πιο σχετικές προτάσεις ή περισσότερες προτάσεις αντίστοιχα. Στο πρόγραμμα μας τις έχουμε επιλέξει έτσι ώστε ο χρόνος εκτέλεσης να είναι αποδεκτός, δηλαδή εντός μερικών δευτερολέπτων.

6.3 Παρουσίαση αποτελεσμάτων

Για την παρουσίαση των αποτελεσμάτων έχει προστεθεί στο γραφικό περιβάλλον της εφαρμογής αριστερά του ιστορικού αναζητήσεων μια νέα λίστα και ένα επιπλέον κουμπί “Find Similar” όπως φαίνεται στην Εικόνα 6.3.1. Έτσι, μετά την αναζήτηση ενός ερωτήματος με το πάτημα του κουμπιού εμφανίζονται στην αριστερή λίστα οι σχετικές προτάσεις. Επιλέχθηκε η αναζήτηση να γίνεται με κουμπί και όχι ταυτόχρονα με την αναζήτηση του ερωτήματος έτσι ώστε να μην επηρεάζεται ο χρόνος ανάκτησης των εγγράφων από το ευρετήριο με τα δεδομένα των τραγουδιών.



Εικόνα 6.3.1: Τροποποιημένο γραφικό περιβάλλον για την υποστήριξη σημασιολογικής αναζήτησης

6.4 Παρατηρήσεις

Στην επέκταση της εφαρμογής για την υποστήριξη της σημασιολογικής αναζήτησης εκτός των λειτουργιών που αναφέρθηκαν δοκιμάστηκαν κάποιες επιπλέον τεχνικές για πιο ακριβή αποτελέσματα. Μια τεχνική ήταν να διατρέχεται ολόκληρο το σύνολο των λέξεων του embedding και από αυτές να αποθηκεύονται οι k καλύτερες ως προς το cosine similarity. Μια ακόμα ήταν τα αποτελέσματα των λέξεων που προτείνονται να ταξινομούνται ως προς την ομοιότητα τους στο ερώτημα που έκανε ο χρήστης. Με αυτό τον τρόπο θα εμφανίζονταν πρώτα αυτές που ήταν σημασιολογικά πιο όμοιες. Ωστόσο, αυτές οι δύο τεχνικές και ιδίως η πρώτη, απαιτούσαν υψηλότερο χρόνο εκτέλεσης, της τάξης αρκετά λεπτα, με αποτέλεσμα να επηρεάζεται η λειτουργία της εφαρμογής. Για τον λόγο αυτό δεν χρησιμοποιήσαμε τις τεχνικές αυτές και χρησιμοποιούμε τις παραμέτρους που αναφέραμε στο Κεφάλαιο 6.2.