Πανεπιστήμιο Ιωαννίνων Ανάκτηση Πληροφορίας

Τμήμα Μηχανικών Η/Υ & Πληροφορικής Ακαδημαϊκό Έτος 2022-2023

Μηχανή αναζήτησης τραγουδιών - Φάση 2η

Κωνσταντίνος Δεδικούσης - 2962
Δημητρόπουλος Δημήτριος - 4352

ΣΥΛΛΟΓΗ ΕΓΓΡΑΦΩΝ

Για τη συλλογή των εγγράφων επιλέξαμε να χρησιμοποιήσουμε ένα έτοιμο dataset σε μορφή .csv από από την online κοινότητα data science του Kaggle.

Συγκεκριμένα επιλέξαμε το songs.csv από το οποίο χρησιμοποιούμε τα πεδία title, description, appears on, artist, writers, producer και released.

ΤΡΟΠΟΠΟΙΗΣΗ ΑΡΧΕΙΟΥ

Για την απαραίτητη επεξεργασία του αρχείου δημιουργήσαμε ένα python αρχείο με το όνομα scrapper το οποίο διαβάζει το .csv και για κάθε τραγούδι δημιουργεί και ένα διαφορετικό .txt αρχείο με τις πληροφορίες του τραγουδιού.

Με μια σταθερά N καθορίζουμε το πόσα αρχεία τραγουδιών θα δημιουργηθούν από τον scrapper. Μέγιστος αριθμός αρχείων που μπορεί να δημιουργηθεί είναι 500 καθώς τόσος είναι και ο αριθμός των τραγουδιών που υπάρχουν στο .csv αρχείο.

ΚΑΤΑΣΚΕΥΗ ΕΥΡΕΤΗΡΙΟΥ

Για να μπορέσουμε να «διαβάσουμε» και να αναλύσουμε τα αρχεία που παρήγαγε ο scrapper δημιουργήσαμε την κλάση Indexer.

Η παραπάνω κλάση υλοποιεί έναν ευρετήριο για αρχεία κειμένου χρησιμοποιώντας τη Lucene.

Αρχικοποίηση Indexer - Αρχικά, δημιουργείται ένας Indexer με τη χρήση ενός directory όπου θα αποθηκευτεί το ευρετήριο και ενός άλλου που περιέχει το path των αρχείων που θα ευρετηριαστούν.

Δημιουργία ευρετηρίου - Μέσω της μεθόδου createIndexer( ), ο indexer δημιουργεί το ευρετήριο και εκτελεί τις παρακάτω δράσεις

- Διαγραφή υπάρχοντος ευρετηρίου για να ξεκινήσει ένα νέο index.

- Ανάκτηση όλων των αρχείων από το path που έχει οριστεί και επιλογή μόνο των .txt αρχείων.

- Για κάθε αρχείο, δημιουργείται ένα νέο έγγραφο (Document) στο ευρετήριο. Το περιεχόμενο του αρχείου διαβάζεται και αποθηκεύεται σε ένα πεδίο του εγγράφου, ενώ οι σχετικές πληροφορίες των επιμέρους πεδίων του εγγράφου όπως ο τίτλος, η περιγραφή, ο καλλιτέχνης κτλ. αποθηκεύονται σε επιμέρους αντίστοιχα πεδία.

Πιο αναλυτικά τα πεδια που χρησιμοποιούνται είναι τα ακόλουθα

- **contents** όπου αποθηκεύεται όλο το περιεχόμενου του εγγράφου

- **title** όπου αποθηκεύεται ο τίτλος του τραγουδιού

- **description** όπου αποθηκεύεται αποθηκεύεται μια σύντομη περιγραφή του τραγουδιού

- **appears** όπου αποθηκεύεται το album όπου ανήκει το τραγούδι

- **artist** όπου αποθηκεύεται το όνομα και το επώνυμο του καλλιτέχνη ή το όνομα του συγκροτήματος του τραγουδιού

- **writes** όπου αποθηκεύεται το όνομα και το επώνυμο του/των στιχουργού/ων του τραγουδιού

- **producer** όπου αποθηκεύεται το όνομα και το επώνυμο του παραγωγού του τραγουδιού

- **released** όπου αποθηκεύεται η ημερομηνία κυκλοφορίας του τραγουδιού

Τέλος το δημιουργηθέν έγγραφο προστίθεται στο ευρετήριο

Στο σημείο αυτό να σημειωθεί ότι με την κλάσης LuceneConsts ορίζουμε τα ονόματα των προαναφερθέντων πεδίων.

Αποθήκευση και κλείσιμο ευρετηρίου - Αφού ολοκληρωθεί η διαδικασία του indexing για όλα τα αρχεία, το ευρετήριο αποθηκεύεται με την εντολή writer.commit( ), ενώ ο indexer και το directory τερματίζουν με τις εντολές writer.close( ) και dir.close( ) αντίστοιχα.

Επιστροφή αποτελέσματος - Τέλος, η μέθοδος createIndexer( ) επιστρέφει τον αριθμό των εγγράφων που έχουν προστεθεί στο ευρετήριο, ώστε να γίνεται έλεγχος για την επιτυχή ολοκλήρωση της διαδικασίας ευρετηρίασης.

ΑΝΑΛΥΣΗ ΚΕΙΜΕΝΟΥ

Για την ανάλυση του κειμένου επιλέξαμε την κλάση StandardAnalyzer καθώς θεωρήσαμε πως παρέχει μια ισορροπημένη προσέγγιση ανάλυσης και δεν εισάγει πολλές εξειδικευμένες παραμέτρους.

Η κλάση StandardAnalyzer εφαρμόζει μια σειρά από κανόνες επεξεργασίας του κειμένου, που περιλαμβάνουν την διαίρεση του κειμένου σε λέξεις (tokenization) με βάση τα κενά, την μετατροπή όλων των χαρακτήρων σε πεζά (lowercasing) και την αφαίρεση συμβόλων στίξης και συνημμένων σημείων (punctuation removal) γεγονός το οποίο τον καθιστά ιδανική περίπτωση για τις ανάγκες αυτής της εφαρμογής.

ΑΝΑΖΗΤΗΣΗ

Η αναζήτηση γίνεται με την κλάση Searcher η οποία διαβάζει το ευρετήριο από το δίσκο και με χρήση της μεθόδου search επιστρέφει τη λίστα με τα documents που αντιστοιχούν στην ερώτηση του χρήστη.

Κατά την αρχικοποίηση ενός αντικειμένου Searcher, δημιουργείται ένας IndexSearcher και ένας IndexReader για την περάτωση της αναζήτησης.

Η κλάση χρησιμοποιεί τους QueryParser που ορίζονται για κάθε πεδίο καθώς και για ολόκληρο το έγγραφο για να μετατρέψει τις αναζητήσεις σε αντικείμενα Query, τα οποία μπορούν να χρησιμοποιηθούν για την αναζήτηση στο ευρετήριο.

Οι QueryParser χρησιμοποιούν τον StandardAnalyzer για την ανάλυση των αναζητητικών ερωτημάτων.

Η μέθοδος search δέχεται ένα searchQuery και ένα field με βάση το οποίο θα γίνει η αναζήτηση.

Ανάλογα με το πεδίο που ορίζει ο χρήστης, επιλέγεται ο κατάλληλος QueryParser για τη μετατροπή του ερωτήματος σε Query αντικείμενο. Εφόσον ο χρήστης δεν επιλέξει πεδίο η αναζήτηση γίνεται σε ολόκληρο το έγγραφο με χρήση του αντίστοιχου QueryParser.

Στη συνέχεια, ο IndexSearcher εκτελεί την αναζήτηση και επιστρέφει έναν πίνακα ScoreDoc με τα αποτελέσματα.

Κάθε αποτέλεσμα αποθηκεύεται σε ένα αντικείμενο Document και προστίθεται στη λίστα resultDocs.

ΠΑΡΟΥΣΙΑΣΗ ΤΩΝ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Τα αποτελέσματα της αναζήτησης παρουσιάζονται ανα 10 (δέκα) στον χρήστη, με τα πιο σχετικά αποτελέσματα να εμφανίζονται πρώτα βάσει της συναφειας τους με το ερώτημα που τέθηκε.

Εφόσον τα αποτελέσματα της αναζήτησης είναι αριθμός μεγαλύτερος του 10 (δέκα) στην αρχική του UI που έχουμε υλοποιήσει εμφανίζονται μόνο οι δέκα πρώτες επιλογές. Εφόσον ο χρήστης επιθυμεί να δει και τις υπόλοιπες πρέπει να πατήσει το κουμπί “NEXT 10” όπου θα του παρουσιάζει τα υπόλοιπα αποτελέσματα ανα 10. Το κουμπί αφαιρείται από το UI εφόσον δεν υπάρχουν άλλα αποτελέσματα προς εμφάνιση.

Επίσης, ο χρήστης έχει την δυνατότητα να κατηγοριοποιήσει τα αποτελέσματα της αναζήτησης ανάλογα με το πεδίο σε αύξουσα σειρά πατώντας το κουμπί “ORDER BY”. Εφόσον το πεδίο προς κατηγοριοποίηση παραμείνει κενό το κουμπί “ORDER BY” δεν έχει καμία λειτουργία.

Επιπρόσθετα το application διατηρεί τις 10 (δέκα) κορυφαίες αναζητήσεις με βάση τη συχνότητα, χρησιμοποιώντας ένα ιστορικό των αναζητήσεων.

Το ιστορικό αποθηκεύεται σε ένα αρχείο με το όνομα "search-history.txt" στον δίσκο.

Πιο συγκεκριμένα, όταν ο χρήστης πατάει το κουμπί "Search", το πρόγραμμα ανοίγει το αρχείο και καταγράφει το ερώτημα που έχει εισαχθεί από τον χρήστη.

Αν το ερώτημα έχει ήδη εγγραφή στο αρχείο, αυξάνουμε τον αριθμό εμφανίσεων του ερωτήματος στο Map που χρησιμοποιούμαι για την διατήρησει των εν λόγω αποτελεσμάτων. Το Map είναι της μορφής <ερώτημα, αριθμός εμφανίσεων του ερωτήματος>.

Εφόσον το ερώτημα είναι νέο δημιουργούμε μια νεα εγγραφή στο Map με το ερώτημα και τον αριθμό 1 (ενα) ως αριθμό εμφανίσεων του ερωτήματος.

Εάν ο χρήστης κάνει double click σε ένα από τα αποτελέσματα της ερώτησης του, δημιουργείται και ανοίγει ένα νέο παράθυρο με τίτλο τον τίτλο του τραγουδιού και περιεχόμενο της πληροφορίες του τραγουδιού.

ΧΡΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Αρχικά, πρέπει να γίνει εκτέλεση του scrapper για να διαβαστούν και τροποποιηθούν τα δεδομένα από το αρχείο .csv ώστε να είναι έτοιμα προς ευρετηριοποίηση, ανάλυση και αναζήτηση. Η εκτέλεση του scrapper γίνεται με την εντολή ***python3 scrapper.py*** στο directory όπου βρίσκεται το python αρχείο.

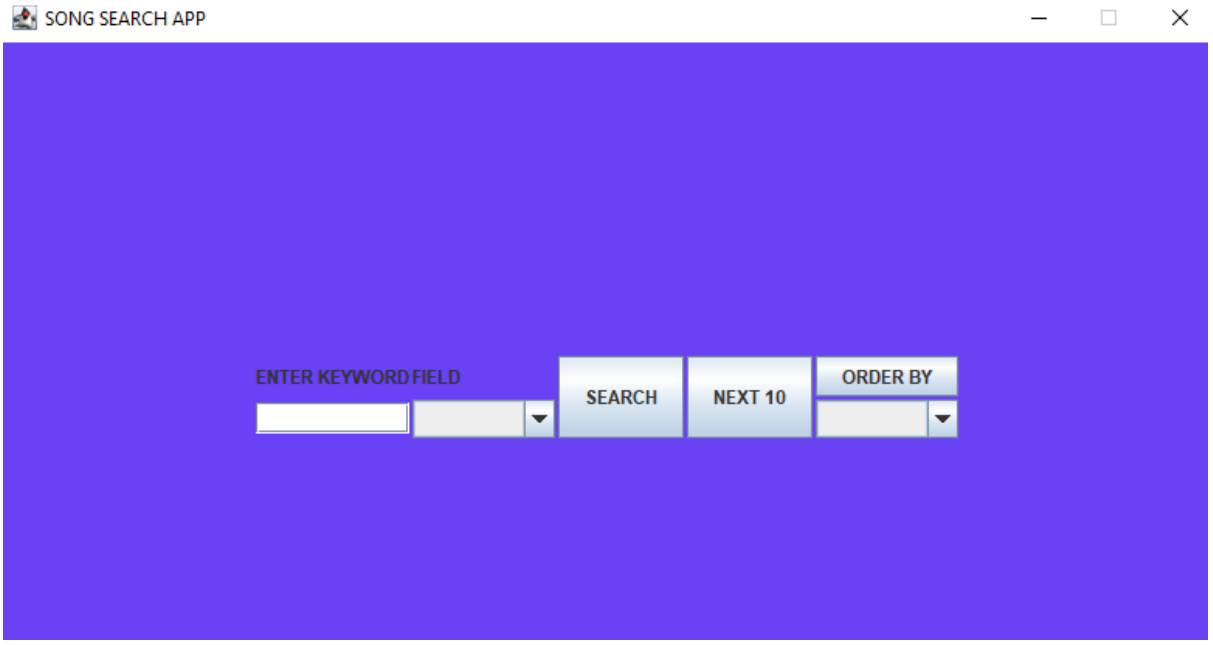
Ο αριθμός των τραγουδιών που θα διαβαστούν ορίζεται από την σταθερά N η οποία για τον παραδοτέο κώδικα έχει αρχικοποιηθεί στην τιμή 20.

Ωστόσο, έχουμε τη δυνατότητα να ορίσουμε τον αριθμό του N σε οποιαδήποτε θετική ακέραια τιμή μέχρι και το 500 όπου είναι ο μέγιστος αριθμός τραγουδιών που υπάρχουν στο .csv αρχείο.

Στη συνέχεια, για την επεξεργασία των αρχείων, εκτελούμε την IndexStart. Όπου αναμένουμε να τελειώσει για να πάρουμε ένα output της μορφής

```
Indexing: C:\Users\dimit\Desktop\Query-Searcher\src\main\files\Like a Rolling Stone.txt
Indexing: C:\Users\dimit\Desktop\Query-Searcher\src\main\files\London Calling.txt
Indexing: C:\Users\dimit\Desktop\Query-Searcher\src\main\files\Maybellene.txt
Indexing: C:\Users\dimit\Desktop\Query-Searcher\src\main\files\My Generation.txt
Indexing: C:\Users\dimit\Desktop\Query-Searcher\src\main\files\Purple Haze.txt
Indexing: C:\Users\dimit\Desktop\Query-Searcher\src\main\files\Respect.txt
Indexing: C:\Users\dimit\Desktop\Query-Searcher\src\main\files\Smells Like Teen Spirit.tx
Indexing: C:\Users\dimit\Desktop\Query-Searcher\src\main\files\What'd I Say.txt
Indexing: C:\Users\dimit\Desktop\Query-Searcher\src\main\files\What's Going On.txt
Indexing: C:\Users\dimit\Desktop\Query-Searcher\src\main\files\Yesterday.txt
Indexed files: 20
```

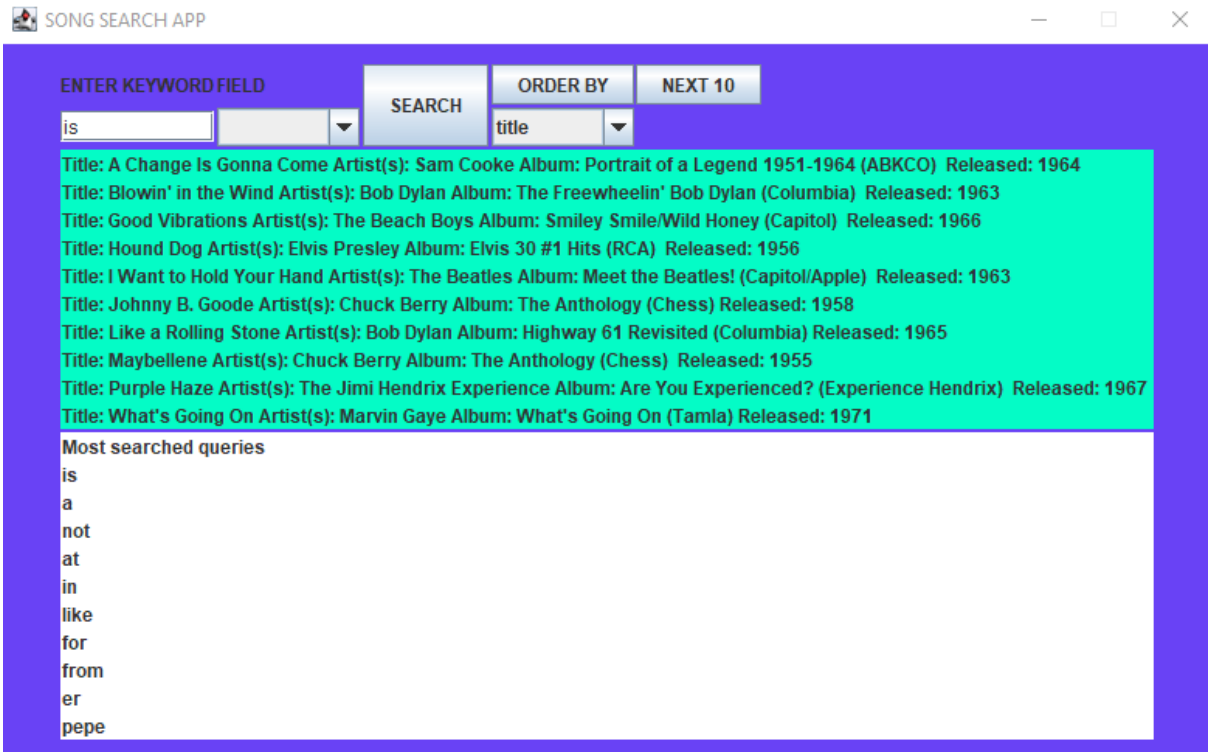
Τότε τερματίζουμε αυτό το configuration και τρέχουμε την InformationRetrievalApp όπου θα εμφανίσει ένα παράθυρο με την μηχανή αναζήτησης.



Για αναζήτηση του keyword **is** σε όλο το περιεχόμενο των αρχείων (αφήνοντας κενή την επιλογή πεδίου) η εφαρμογή παράγει τα ακόλουθα αποτελέσματα

Όπως μπορείτε να δείτε ο όρος **is** είναι ο πιο συχνά αναζητήσιμος όρος και το πιο σχετικό τραγούδι με αυτόν το keyword είανι το **Johnny B. Goode** στο οποίο αν κάνουμε double click μπορούμε να δούμε όλες τις πληροφορίες του

Εφόσον επιλέξουμε να κατηγοριοποιήσουμε τα αποτελέσματα με βάση τον **τίτλο** το αποτέλεσμα που παράγεται είναι το ακόλουθο



Όπως μπορείτε να δείτε τα αποτελέσματα έχουν κατηγοριοποιηθεί σε αύξουσα σειρά με βάση τον τίτλο τους.

ΣΧΟΛΙΑ

Ο σύνδεσμος για το github repository της εργασίας είναι ο ακόλουθος
https://github.com/dabouledidia/Information-Retrieval