Τμήμα Μηχ. Η/Υ & Πληροφορικής **Εαρινό εξάμηνο 2023**

Πολυτεχνική σχολή, Πανεπιστήμιο Ιωαννίνων **Διδάσκων: Ευαγγελία Πιτουρά**

ΜΥΕ003 :ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

Στοιχεία ομάδας:

Χαράλαμπος Θεοδωρίδης: 4674 cs04674@uoi.gr

Γιώργος Σιδηρόπουλος: 4789 cs04789@uoi.gr

Σκοπός της εργασίας :

Σκοπός της εργασίας είναι η χρήση βιβλιοθηκών της Lucene για την ανάπτυξη ενός μηχανισμού αναζήτησης που επιτρέπει στους χρήστες να εξερευνήσουν μια συλλογή τραγουδιών. Στόχος της είναι η δημιουργία ενός αποτελεσματικού και αποδοτικού συστήματος που θα επιτρέπει στους χρήστες να βρίσκουν εύκολα σχετικές πληροφορίες μέσα στην συλλογή των τραγουδιών.

GitHub Link:

https://github.com/HarrisTheo/LuceneProject.git

Link For CSV,Embeddings,Video:

https://drive.google.com/drive/folders/1JG1f61ryFg1ir_58QSVBu9TIKwI-Ljen?usp=share_link

Περιεχόμενα :

Συλλογή δεδομένων

Η συλλογή αποτελείται από ένα αρχείο CSV που περιλαμβάνει 643 δημοφιλείς καλλιτέχνες και 44.824 μοναδικά τραγούδια. Αυτό το αρχείο CSV έχει 4 στήλες όπου : 1)Η πρώτη στήλη περιέχει το όνομα του καλλιτέχνη (η συγκροτήματος) 2)Η δεύτερη στήλη περιέχει τον τίτλο του τραγουδιού 3)Η τρίτη στήλη περιέχει τον σύνδεσμο του τραγουδιού 4)Η τέταρτη στήλη περιέχει τους στίχους του τραγουδιού.

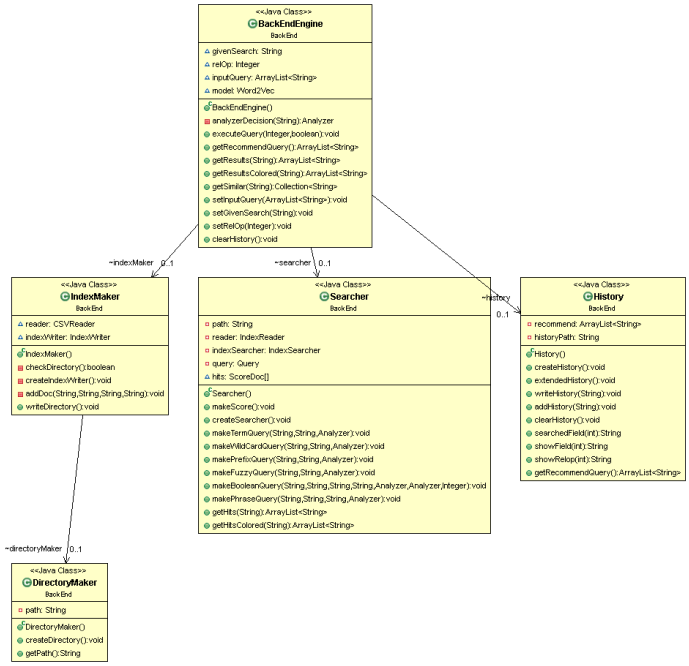
artist	song	link	text
ABBA	Ahe's My Kind Of Girl	/a/abba/ahes+my+kind+of+girl_20598417.html	Look at her face, it's a wonderful face
ABBA	Andante, Andante	/a/abba/andante+andante_20002708.html	Take it easy with me, please

Το csv αρχείο κατέβηκε από την ιστοσελίδα :

https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset

BackEnd

Αρχικά παρουσιάζεται το διάγραμμα κλάσεων του Backend.

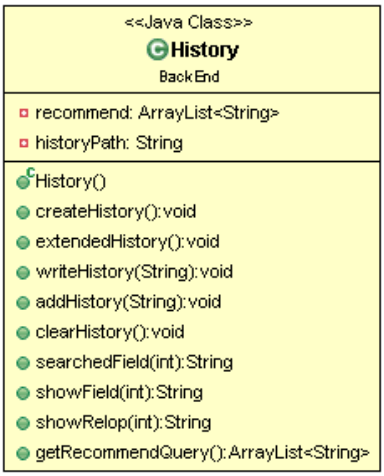


Το BackEnd της μηχανής αναζήτησης αποτελείται από τις κλάσεις :

- History.class : Είναι υπεύθυνη για την δημιουργία και διαχείριση του ιστορικού αναζητήσεων καθώς και για την δημιουργία προτεινόμενων ερωτημάτων αναζήτησης .
- DirectoryMaker.class : Είναι υπεύθυνη για την δημιουργία ενός directory στο path που ορίζεται ως πεδίο στην κλάση αυτή.
- IndexMaker.class : Είναι υπεύθυνη για την δημιουργία ενός ευρετηρίου(αν αυτό δεν υπάρχει ήδη) καθώς και για την εισαγωγή εγγράφων σε αυτό.
- Searcher.class : Είναι υπεύθυνη για την δημιουργία και εκτέλεση ερωτημάτων στο ευρετήριο καθώς και για την ανάκτηση των συναφών με το ερώτημα εγγράφων.

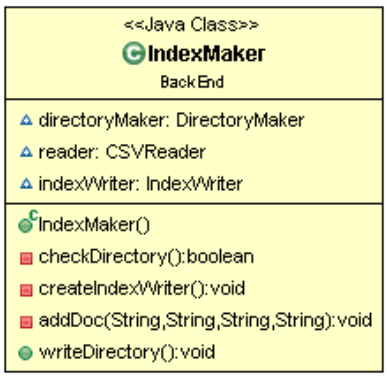
- BackEnd.class : Η κλάση αυτή αλληλεπιδρά με το FrontEnd της μηχανής αναζήτησης και καλεί μεθόδους των προαναφερθέν κλάσεων.

History.Class



Ο constractor αυτής της κλάσης δημιουργεί (αν δεν υπάρχει) ένα αρχείο με το ιστορικό αναζητήσεων (createHistory()) και διαβάζει από αυτό όλες τις αναζητήσεις , αν υπάρχουν(extendedHistory()) . Η εισαγωγή μιας αναζήτησης στο αρχείο γίνεται με την writeHistroy() ενώ η εκκαθάριση (διαγραφή) του ιστορικού γίνεται με την clearHistory(). Η μέθοδος getRecommendQuery() επιστρέφει μια λίστα με τα προτεινόμενα ερωτήματα που δημιουργήθηκαν με την χρήση των μεθόδων searchedField() , showRelop(), showField().

IndexMaker.Class

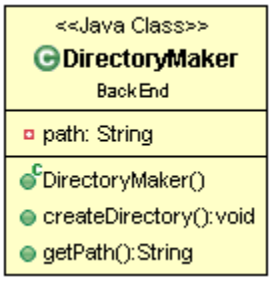


Ενδιαφέρον παρουσιάζει η μέθοδος createIndexWriter() η οποία δημιουργεί το ευρετήριο και καλείτε στην περίπτωση που το ευρετήριο δεν υπάρχει . Ο έλεγχος για την ύπαρξη του γίνεται με την checkDirectory() . Ο σχεδιασμός του index διαφέρει από αυτό που παρουσιάστηκε στην πρώτη φάση της εργασίας με μοναδική διαφορά την προσθήκη ενός synonymFilter στον StandardAnalyzer που χρησιμοποιείται στο πεδίο text(στίχοι). Η αλλαγή αυτή έγινε με σκοπό να κανονικοποιήσουμε τους όρους που εισάγονται στο πεδίο αυτό (τον τελικό σχεδιασμό αυτό μπορείτε να τον βρείτε σε αυτή την ενότητα Σχεδιασμός) . Το synonym filter αντλεί την πληροφορία για τα συνώνυμα από το αρχείο που βρίσκεται στο resources του project και έχει όνομα synonyms_en.txt . Η δομή του *.txt:

```
audio, sound
aureate, gilded, gilt, gold, golden
authority, authorization, authorisation, potency, dominance, say-so
award, accolade, honor, honour, laurels
```

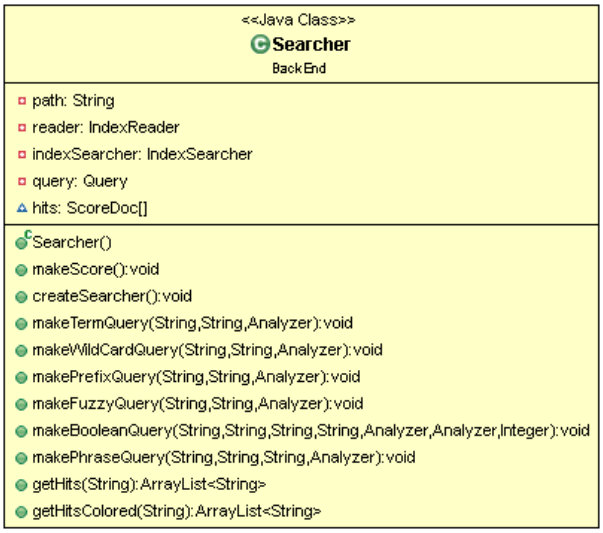
Η μέθοδος writeDirectory() και η addDoc() χρησιμοποιούνται για να διαβάσουν από το csv τις γραμμές και να εισάγουν στο ευρετήριο έγγραφα.

DirectoryMaker.Class



Αποτελεί την μικρότερη κλάση του συστήματος τόσο σε μέγεθος όσο και σε λειτουργικότητα. Η μόνη λειτουργεία της είναι η δημιουργία ενός directory στο path "C://Programming//Lucene//Directory".

Searcher.Class



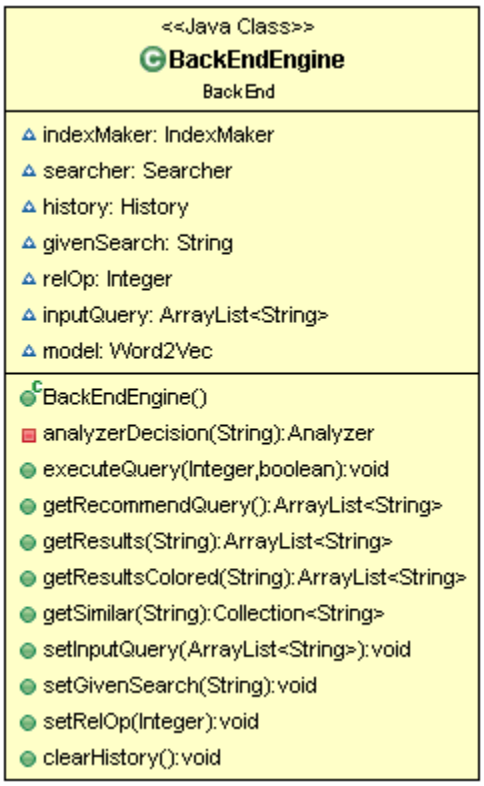
Οι μέθοδοι :

- makeTermQuery()
- makeWildCardQuery()
- makePrefixQuery()
- makeFuzzyQuery() (30% similarity)
- makeBooleanQuery() (2 slop)
- makePhraseQuery()

Υλοποιούν ένα είδος ερωτήματος και ύστερα καλούν την μέθοδο makeScore() με σκοπό να αρχικοποιήσουν το πεδίο hits που αποτελεί ένα array με τα έγγραφα διατεταγμένα σε φθίνουσα σειρά με βάση την συνάφεια τους . Η getHits() και η getHitsColored() αποτελούν getters για το array αυτό με μόνη

διαφορά πως η getHitsColored() χρησιμοποιεί έναν highlighter με σκοπό οι λέξεις κλειδιά να παρουσιάζονται τονισμένες στο αποτέλεσμα. Η κλάση searcher έχει τις μεθόδους αυτές οι οποίες διαχειρίζονται από την κλάση BackEndEngine().

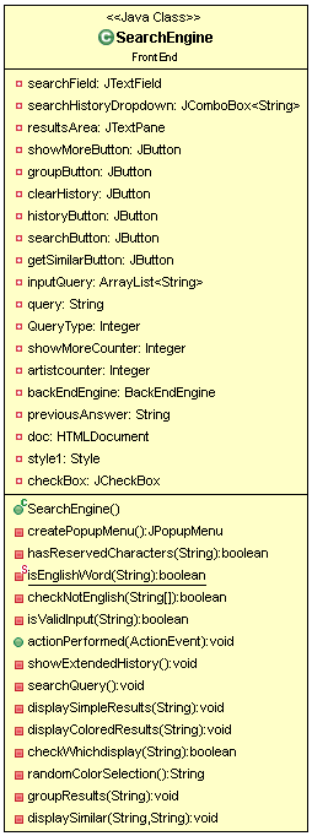
BackEndEngine.class



Η BackEndEngine αποτελεί τον διαχειριστή του συστήματος. Η μέθοδος executeQuery() εκτελεί το query που αντιστοιχεί στην ακέραια παράμετρο που δέχεται σαν όρισμα και αποθηκεύει την αναζήτηση στο ιστορικό αναζητήσεων. Η κλάση αυτή με τις μεθόδους getRecommendQuery(), getSimilar(), getResults() , getResultsColored() παρέχει στο FrontEnd τις απαντήσεις σε οποιαδήποτε απαίτηση του χρήστη. Η κλάση έχει ακόμα σαν πεδίο ένα model από embeddings με σκοπό να παρέχει την δυνατότητα σημασιολογικής αναζήτησης . Τα διανύσματα των λέξεων εισήχθησαν στο μοντέλο μας από ένα αρχείο που περιέχει 300.000 διανύσματα και αποτελούν μέρος μιας μεγαλύτερης συλλογής που αριθμεί πάνω από 300billion διανύσματα λέξεων. Η επιλογή των διανυσμάτων αυτών έγινε με ένα πρόγραμμα της python το οποίο επέλεξε τα embedding που είναι πλησιέστερα στους όρους που περιέχει το ευρετήριο μας. Η χρήση τους γίνεται με τρόπο που περιγράφεται στην ενότητα 2.

FrontEnd

SearchEngine.class

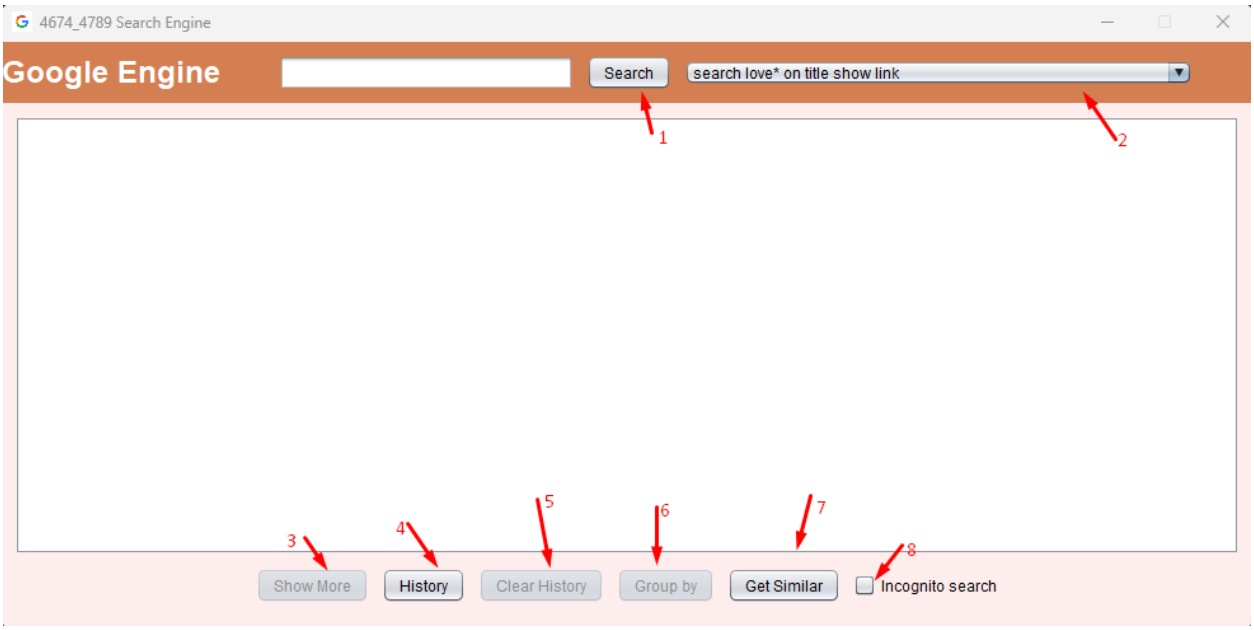


Η searchEngineClass() δημιουργεί το γραφικό περιβάλλον αλληλεπίδρασης του χρήστη με το πρόγραμμα. Το περιβάλλον δημιουργήθηκε με την χρήση της βιβλιοθήκης swing . Περιέχει μεθόδους :

- για τον συντακτικό έλεγχο ενός ερωτήματος (isValidInput(), checkNotEnglish(), isEnglishWord()).
- για την εμφάνιση των αποτελεσμάτων στον χρήστη (displaySimilar(), displaySimpleResults() ,displayColoredResults(),chceckwhichdisplay())
- για την ομαδοποίηση των αποτελεσμάτων με βάση την αλφαβητική σειρά του πρώτου γράμματος κάθε αποτελέσματος.
- για την εμφάνιση λέξεων που είναι σημασιολογικά κοντά με έναν όρο.
- για την εμφάνιση του ιστορικού.

Αλληλεπίδραση με την μηχανή αναζήτησης

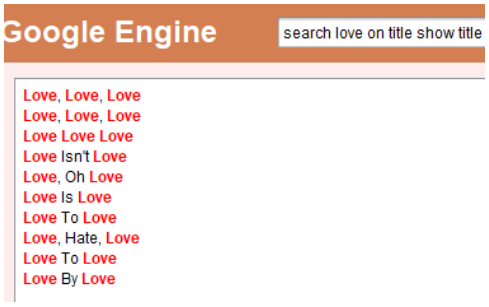
Όταν ο χρήστης εκτελέσει την main του προγράμματος θα του εμφανιστεί το παρακάτω παράθυρο αναζήτησης.



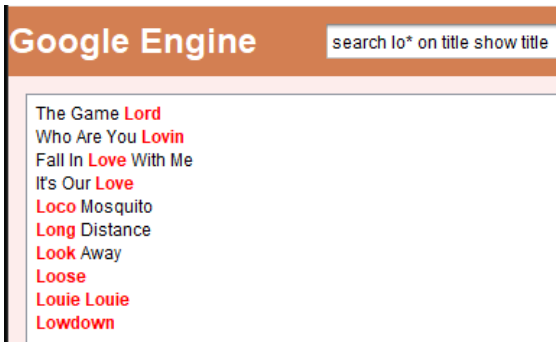
1)Search Button: Ο χρήστης μπορεί να διατυπώσει ένα ερώτημα στο box που βρίσκεται αριστερά του search και πατώντας το κουμπί αυτό να το εκτελέσει. Τονίζεται πως η πληροφορία που μπορεί να ανακτηθεί αφορά το πεδίο του link , ονόματος καλλιτέχνη(artist) και του τίτλου τραγουδιού(title) ενώ ο χρήστης μπορεί να αναζήτηση όρους στο πεδίο των στοίχων(text) του τίτλου(title) και του ονόματος του καλλιτέχνη(artist) . Τα ερωτήματα διατυπώνονται με τον εξής τρόπο :

- Term Query -> search term on field1 show field2 .

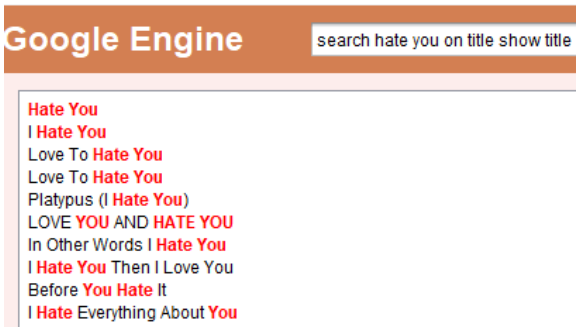
  Με αυτό τον τρόπο εκτελείτε ένα term query που αναζητά τον ορό term στο πεδίο field1 και εμφανίζει το πεδίο field2.



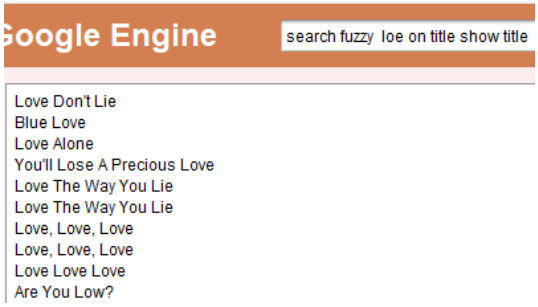
- WildCardQuery-> search wildCardTerm on field1 show field2 .
  Με αυτό τον τρόπο εκτελείται ένα wildCardQuery . Τονίζεται πως ο αστερίσκος μπορεί να βρίσκεται στην αρχή, στο τέλος αλλά και ενδιάμεσα της λέξεις πχ:
  - Lo*
  - *ve
  - o*v



- PhraseQuery-> search term1 term2 on field1 show field2.
  Με αυτό τον τρόπο εκτελείται ένα phraseQuery. Τονίζεται πως το μέγιστο πλήθος λέξεων που μεσολαβεί ανάμεσα στους term1 και term2 ορούς είναι 2. Πχ:

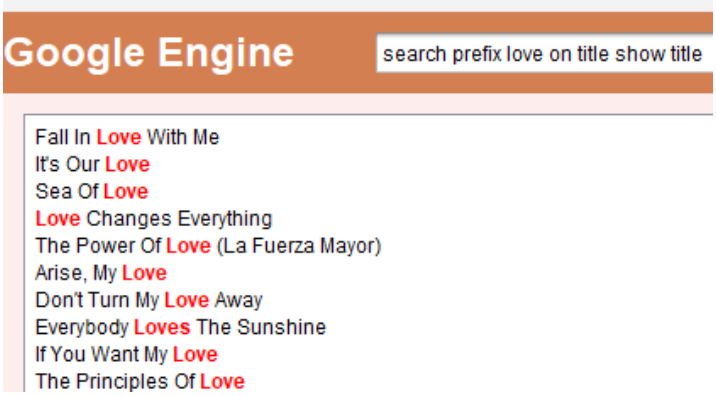


- FuzzyQuery->search fuzzy term on field1 show field2.
  Με αυτό τον τρόπο εκτελείται ένα fuzzy query. Τονίζεται πως ένα έγγραφο θα ανακτηθεί αν περιέχει κάποιον όρο με τουλάχιστον 30% ομοιότητα με τον ορό που αναζητείται. Πχ:

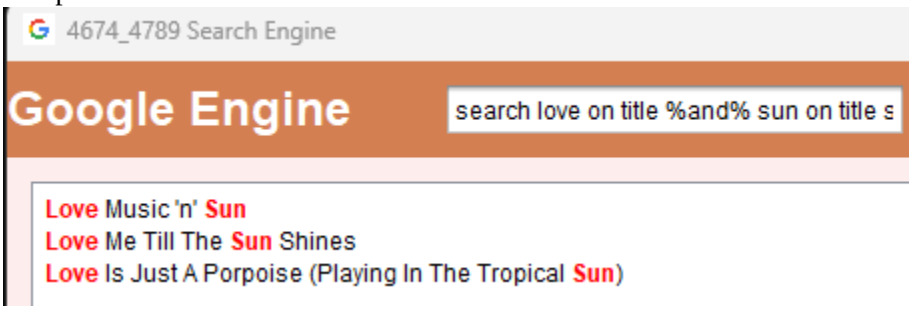


Οι λέξεις κλειδιά δεν είναι τονισμένες στο αποτέλεσμα καθώς η ομοιότητα του ορού loe με κάποιον όρο είναι μικρή και έτσι δεν είναι δυνατόν για τον highlighter να υπολογίσει την ομοιότητα των όρων.

- Prefix Query-> search prefix lo on title show title .
  Με αυτό τον τρόπο εκτελείται ένα prefix query . Πχ:



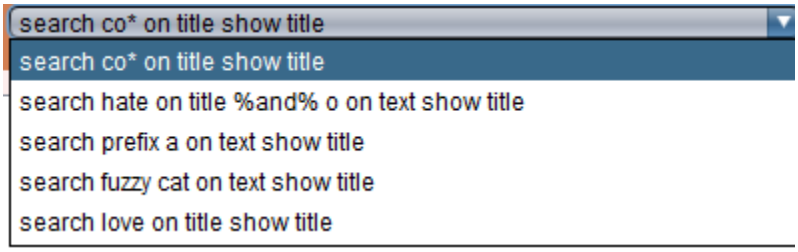
- Boolean Query-> search term1 on field1 %relOp% term2 on field2 show field3.
  Όπου relop : and , or , not . Το παραπάνω ερώτημα αναζητά τον term1 στο πεδίο field1 και τον term2 στο field2 . Τα αποτελέσματα που θα εμφανιστούν θα πρέπει να πληρούν την συνθήκη του relOp .



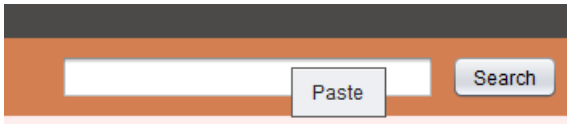
Η απάντηση στο ερώτημα search love on title %and% sun on title show title.

Τα αποτελέσματα θα περιέχουν τους όρους κλειδιά υπογραμμισμένους μόνο στην περίπτωση που το πεδίο στο οποίο αναζητείται κάποιος όρος ταιριάζει με το πεδίο των εγγράφων που θα εκτυπωθεί στην οθόνη. Εξαίρεση αποτελεί η περίπτωση που το query που διατυπώνεται είναι ένα fuzzy query.

2)Recommend drop Bar:
Όταν ο χρήστης πατήσει πάνω στην dropBar τότε του εμφανίζεται μια λίστα με προτεινόμενα ερωτήματα.

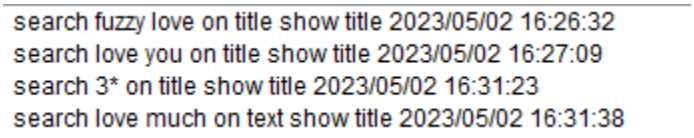


Τα προτεινόμενα ερωτήματα ανανεώνονται μετά από κάθε search. Επιλέγοντας ένα από αυτά , το ερώτημα αντιγράφεται απευθείας στο clipboard και είναι έτοιμο για επικόλληση στο search box.



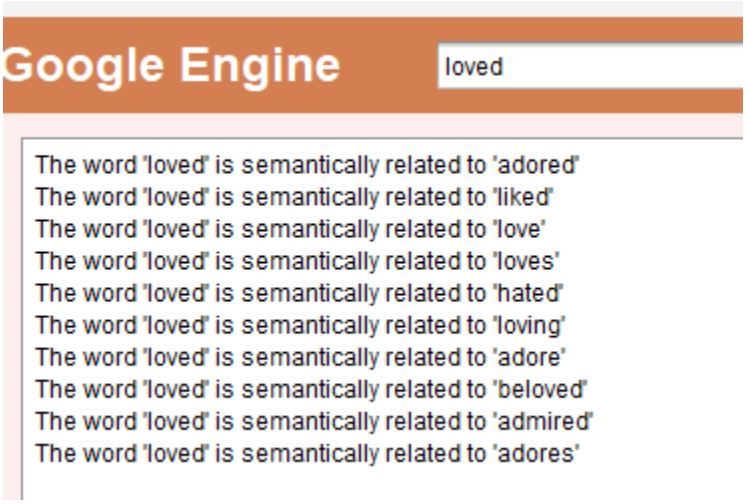
3) Show more button: Με το πάτημα του button ο χρήστης έχει την δυνατότητα να εμφανίσει ακόμα 10 αποτελέσματα . Ο χρήστης μπορεί να πατήσει το κουμπί όσες φορές επιθυμεί μέχρι να εξαντληθούν όλα τα αποτελέσματα.

4)History Button: Με το πάτημα του button εμφανίζεται το ιστορικό αναζητήσεων , όπου κάθε γραμμή του περιέχει την αναζήτηση που έγινε καθώς και την ώρα και την ημερομηνία που εκτελέστηκε .



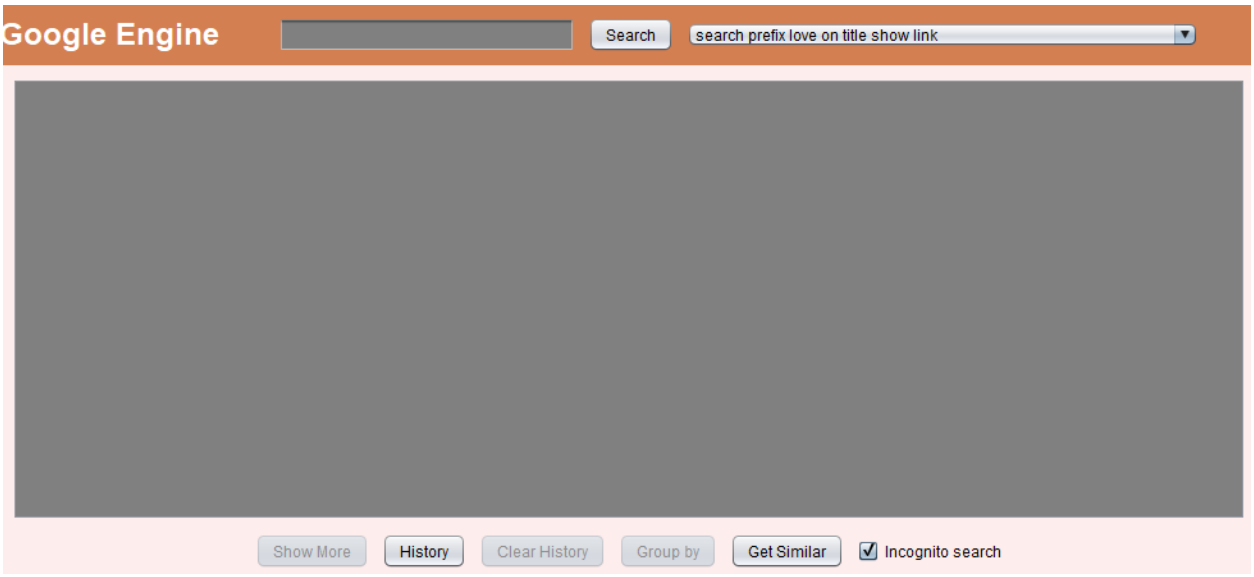
6)Group By : Με το πάτημα του button εμφανίζει στον χρήστη τα αποτελέσματα ομαδοποιημένα. Κάθε ομάδα έχει και ένα διαφορετικό χρώμα με σκοπό να είναι ευδιάκριτη.

7)Get Similar : Με το πάτημα του κουμπιού το σύστημα επιστρέφει στον χρήστη τις 10 λέξεις που είναι σημασιολογικά κοντά με την λέξη που βρίσκεται στο search box.



Το σύστημα μας χρησιμοποιεί τα διανύσματα λέξεων (embedding) για να υλοποίηση αυτή την αναζήτηση . Σκοπός είναι να βοηθήσει τον χρήστη να αναζητήσει κάποιον όρο που είναι σημασιολογικά κοντά με την λέξη που επιθυμεί . Ύστερα ο χρήστης μπορεί να αναζήτηση με οποιοδήποτε τρόπο μια από τις λέξεις που του εμφανίζονται και να ανακτήσει την επιθυμητή πληροφορία.

8)Incognito search check box: Επιλέγοντας το check box ο χρήστης ενεργοποιεί την ανώνυμη αναζήτηση και έτσι οποιαδήποτε αναζήτηση και να κάνει δεν θα αποθηκευτεί στο ιστορικό.



Σχεδιασμός

Αρχικά η μονάδα του εγγράφου που ορίσαμε είναι ένα τραγούδι. Τα πεδία της κάθε μονάδας είναι το link , το όνομα καλλιτέχνη, οι στίχοι και ο τίτλος.

Link	Not analyzed	Stored	Term_vector=No
Artist	Analyzed(Custom analyzer)	Stored	Term_vector=No
Song(Title)	Analyzed(Custom analyzer)	Stored	Term_vector=with_position_offset
Text	Analyzed(Standard analyzer)	Non-Stored	Term_vector=with_position_offset

1) Custom analyzer : Περιέχει τα παρακάτω φίλτρα :
   - StandardTokenizer
   - lowercase φίλτρο το οποίο μετατρέπει όλα τα tokens σε lowercase .

```
Analyzer customeAnalyzer = CustomAnalyzer.builder().withTokenizer( name: "standard").addTokenFilter( name: "lowercase").build();
```

2) Standard analyzer: Περιέχει τα παρακάτω φίλτρα :
   - StandardTokenizer
   - LoweCaseFilter
   - StopFilter
   - PorterStemFilter

```
Analyzer standardAnalyzer = CustomAnalyzer.builder().withTokenizer( name: "standard")
        .addTokenFilter(SynonymFilterFactory.class,  ...params: "synonyms", "synonyms_en.txt", "ignoreCase", "true")
        .addTokenFilter( name: "lowercase").addTokenFilter( name: "stop").addTokenFilter( name: "porterstem").build();
```