ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

Εργασία : Μηχανή αναζήτησης ταινιών

Τελική Αναφορά

Μέλη Ομάδας

Ονοματεπώνυμο: Ευάγγελος Ηλιάδης ΑΜ: 3117

Ονοματεπώνυμο: Βασίλης Νίκας ΑΜ: 3143

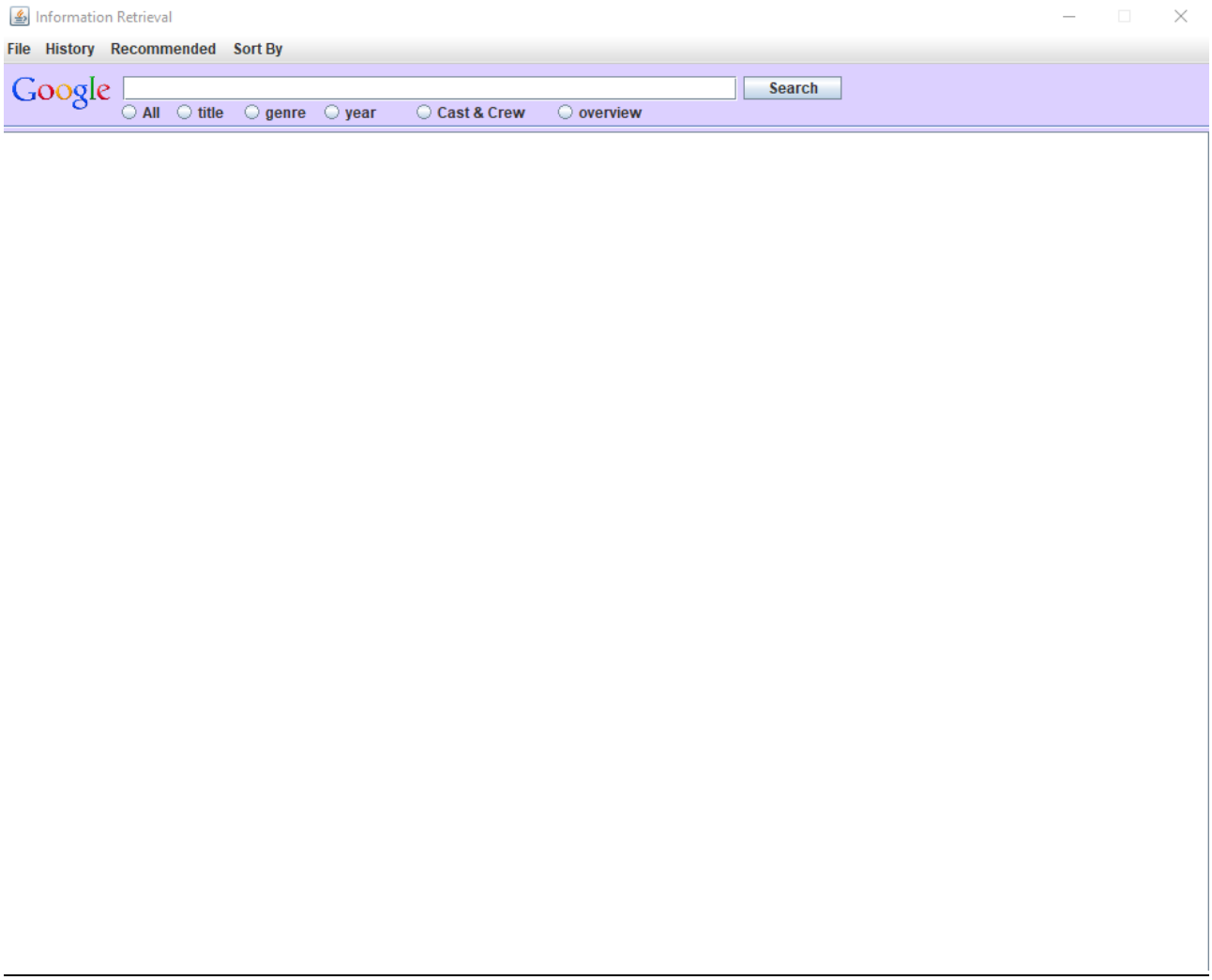
Εισαγωγή

Στα πλαίσια του μαθήματος “Ανάκτηση Πληροφορίας” κληθήκαμε να δημιουργήσουμε μηχανή αναζήτησης ταινιών. Η μηχανή αυτή έχει θα πρέπει να έχει ως σκοπό της την “ανάκτηση πληροφορίας”.

Για τους σκοπούς της εργασίας είχαμε την επιλογή να πάρουμε έτοιμο dataset ταινιών/ σειρών με την προϋπόθεση ότι αυτό θα περιέχει τα βασικά πεδία πληροφοριών(τίτλος, κατηγορία, περιγραφή κτλ.) . Το πρόγραμμά μας υποστηρίζει την επιλογή οποιουδήποτε csv. Γνωρίζουμε ότι το εσωτερικό format αυτών των αρχείων χρησιμοποιεί το κόμμα (“,”) για τον διαχωρισμό των επιμέρους κελιών/ στηλών της εκάστοτε γραμμής. Οι δε γραμμές χωρίζονται με αλλαγή γραμμής “\n”. Η πηγή από την οποία αντλήσαμε τα έγγραφα είναι https://www.kaggle.com/datasets . Πιο συγκεκριμένα το υποδειγματικό αρχείο είναι το “imdb_top_1000.csv”, το οποίο θα περιέχεται μεταξύ των υποβληθέντων αρχείων. Περιέχει όλες τις σημαντικές πληροφορίες τα οποία αναφέρθηκαν κ.α για τις top 1000 ταινίες του της ιστοσελίδας imdb . Εάν κάποιος επιθυμεί να το κατεβάσει από την εν λόγω ιστοσελίδα βρίσκεται μεταξύ των πρώτων αποτελεσμάτων για movie datasets. Εάν επιλεγεί κάποιο άλλο αρχείο κατά την εκτέλεση του προγράμματος ίσως και να χρειαστούν ελάχιστες τροποποιήσεις στον κώδικα του προγράμματος (περισσότερες λεπτομέρειες στο αντίστοιχο κομμάτι)[1].

Υλοποίηση (βήμα-βήμα)

1. Διεπαφή: Όπως με κάθε πρόγραμμα το οποίο απαιτεί την αλληλεπίδραση με τον χρήστη πρώτος μας στόχος ήταν η ανάπτυξη της γραφικής διεπαφής(GUI).

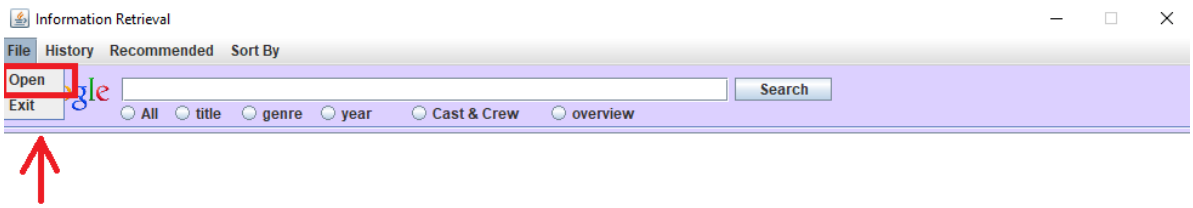


Σχήμα 1: Γραφική διεπαφή (GUI) του προγράμματος

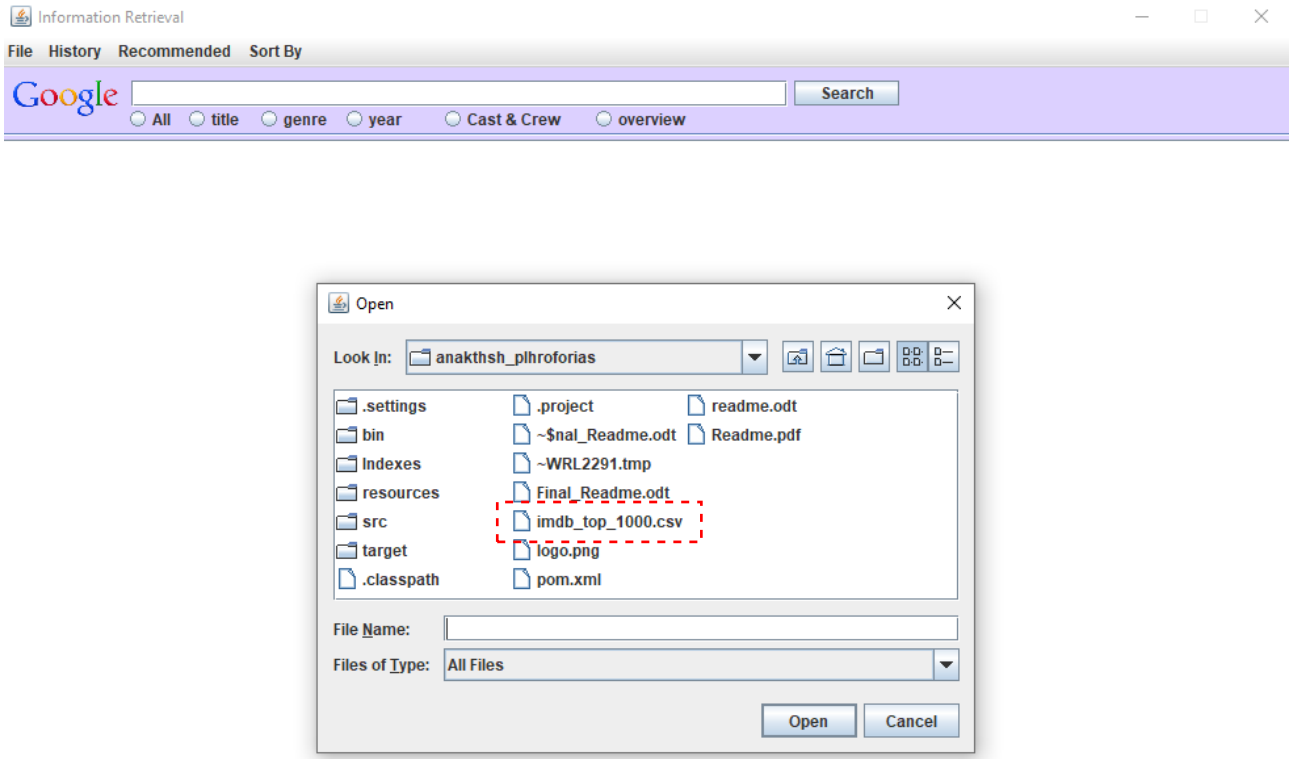
(Disclaimer: Καθαρά και μόνο για λόγους αισθητικής το λογότυπο το οποίο χρησιμοποιήσαμε είναι της Google, της πιο διαδεδομένης μηχανής αναζήτησης.).

Το μεφαλύτερο κομμάτι πληροφοριών για τη γραφική διεπαφή θα βρει κανείς στην κλάση “MyFrame” του προγράμματός μας. Με μια πρώτη ματιά βλέπει κανείς μία γραμμή μενού JMenuBar. Η οποία περιέχει κάποιες από τις σημαντικές λειτουργίες του προγράμματός μας. Επίσης, κάποιος παρατηρεί το πεδίο αναζήτησης (αντικείμενο τύπου JTextField), το κουμπί “search” (JButton). Από κάτω βρίσκονται κουμπιά επιλογής τύπου JRadioButton. Αυτά εξυπηρετούν την αναζήτηση όρων σε συγκεκριμένα πεδία των ευρετηριοποιημένων εγγράφων[2]. Τέλος, υπάρχει ένα μεγάλο πεδίο JTextPane, το οποίο εμπεριέχεται σε ένα JScrollPane. Αυτά εξυπηρετούν στην προβολή των αποτελεσμάτων αναζήτησης.

2. Open (άνοιγμα αρχείου): Πρώτο βήμα είναι κάποιος να πλοηγηθεί στο άνοιγμα του αρχείου (File -> Open).



Έπειτα εμφανίζεται το παράθυρο επιλογής αρχείου.



Σχήμα 2: Στιγμιότυπο από το βήμα επιλογής csv αρχειου

Το τμήμα κώδικα που είναι υπεύθυνο για το άνοιγμα αρχείου είναι στην κλάση "OpenDocument".

```
fileChooser.setCurrentDirectory(new File("."); //Sets default folder for the file chooser to the one of the current program
```

Το πρόγραμμα υποστηρίζει μόνο άνοιγμα csv αρχείων.
Μετά το άνοιγμα γίνεται δημιουργία των αντικειμένων Indexer και QParser. Αντικείμενα υπεύθυνα για την δημιουργία ευρετηρίων και την αναζήτηση του query μέσα σε αυτά, αντίστοιχα.

3. Indexer: Η κλάση αυτή έχει σκοπό της τη δημιουργία ευρετηρίων. Ουσιαστικά πρόκειται για ένα "αντεστραμμένο ευρετήριο", δηλαδή ένας κατάλογο λέξεων. Κάθε μία από αυτές τις λέξεις έχει δίπλα της τη λίστα των ταινιών στις οποίες εμφανίζεται. Επομένως κατά την αναζήτηση με λέξεις κλειδιά από το χρήστη,

το αποτέλεσμα θα είναι οι λίστες ταινιών που περιέχουν τις λέξεις αυτές.

```
indexPath = new File(Paths.get(path).getParent().toString() + "\\Indexes");
```

Με αυτόν τον τρόπο τα ευρετήρια θα δημιουργούνται πάντα σε έναν φάκελο “Indexes” μέσα στον γονικό φάκελο του project.

Όσον αφορά την δημιουργία ευρετηρίων θα χρησιμοποιήσουμε τη βιβλιοθήκη lucene της apache. Διατρέχοντας τα αρχεία, κατασκευάζουμε Document αντικείμενα (org.apache.lucene.document.Document) τα οποία περιέχουν ως πεδία TextField τις σημαντικές πληροφορίες για της ταινίας. Έπειτα με τη βοήθεια της lucene κάνουμε parsing του κειμένου πληροφοριών για κάθε ταινία.Θα «σπάσουμε» το κείμενο σε κομμάτια (tokens). Σε πρώτη φάση για την κατασκευή των ευρετηρίων χρησιμοποιήσαμε αντικείμενα τύπων “IndexWriter”, analyzer, “IndexWriterConfig”. Ο “IndexWriter” είναι υπεύθυνος για τη δημιουργία και διατήρηση του index, ο analyzer είναι υπεύθυνος για το σπάσιμο του text σε κομμάτια και το “IndexWriterConfig” παραμετροποιεί τον “IndexWriter” με τον analyzer κατά τη δημιουργία του ευρετηρίου. Υπάρχουν πολλά είδη analyzer ανάλογα με τον τρόπο που επιθυμούμε να κάνουμε το “tokenization” .Εμείς χρησιμοποιήσαμε τον StandardAnalyzer.

Φτάσαμε στο σημείο “απόσπασης πληροφορίας” από το αρχείο ώστε να χρησιμοποιηθεί για τη δημιουργία των εγγράφων Document που θα μπουν στο ευρετήριο. Ο κώδικας που χρησιμοποιήθηκε είναι ο εξής:

```
//Iterate through the opened file
Reader in = new FileReader(myFile);
Iterable<CSVRecord> records = CSVFormat.EXCEL.withHeader().parse(in);
/*Isolate the important csv columns and store them in TextFields.
*Then add said TextFields to the same Document obj.
*/

for(CSVRecord rec : records)
{
        doc = new Document();
        doc.add(new     TextField("title",rec.get("Series_Title"),Field.Store.YES));
        doc.add(new     TextField("rating",rec.get("IMDB_Rating"),Field.Store.YES));
        doc.add(new TextField("year",rec.get("Released_Year"),Field.Store.YES));
        doc.add(new TextField("genre",rec.get("Genre"),Field.Store.YES));
        doc.add(new TextField("director",rec.get("Director"),Field.Store.YES));
        doc.add(new TextField("star1",rec.get("Star1"),Field.Store.YES));
        doc.add(new TextField("star2",rec.get("Star2"),Field.Store.YES));
        doc.add(new TextField("star3",rec.get("Star3"),Field.Store.YES));
        doc.add(new     TextField("star4",rec.get("Star4"),Field.Store.YES));
        doc.add(new     TextField("overview",rec.get("Overview"),Field.Store.YES));

        indexWriter.addDocument(doc); //Adds the created doc obj to the index.
```

```
}
docFields = doc.getFields(); //List of the document fields. Will be used later for remote access to the document's field names.
indexWriter.commit(); //Commits all pending changes to the index.
indexWriter.close(); //Index changes have finished. Now index should be closed.
```

[1] Φτάσαμε στο σημείο της πρώτης υποσημείωσης περί μικρών τροποποιήσεων από μεριάς χρήστη (εάν χρειαστεί). Ο κώδικας είναι της μορφής:

```
doc.add(new TextField("title",rec.get("Series_Title"),Field.Store.YES));
```

Παρατηρούμε λοίπον ότι παίρνει πληροφορία από το κελί “Series_Title” του csv αρχείου και την βάζει στο πεδίο “title”. Αμέσως καταλαβαίνει κάποιος ότι το csv αρχείο που θα επιλέξει ο χρήστης πρέπει να έχει columns με ονόματα “Series_Title”, “IMDB_Rating”, “Released_Year”, “Genre”, “Director” , “Star1”, “Star2”, “Star3”, “Star4”, “Overview”. Εάν δεν ονομάζονται έτσι το πρόγραμμα δεν θα μπορέσει να αποσπάσει την απαραίτητη πληροφορία. Επομένως υπάρχουν δύο πιθανές επιλογές:

- Η τροποποίηση των ονομάτων για τα columns του csv αρχείου, ώστε να ταιριάζουν με τις απαιτήσεις του προγράμματος
- Η αλλαγή του κώδικα σύμφωνα με τις ανάγκες κάθε αρχείου

Μετά τη δημιουργία των αρχείων ευρετηρίου είναι σειρά του χρήστη να κάνει τη όποια αναζήτηση αυτός επιθυμεί.

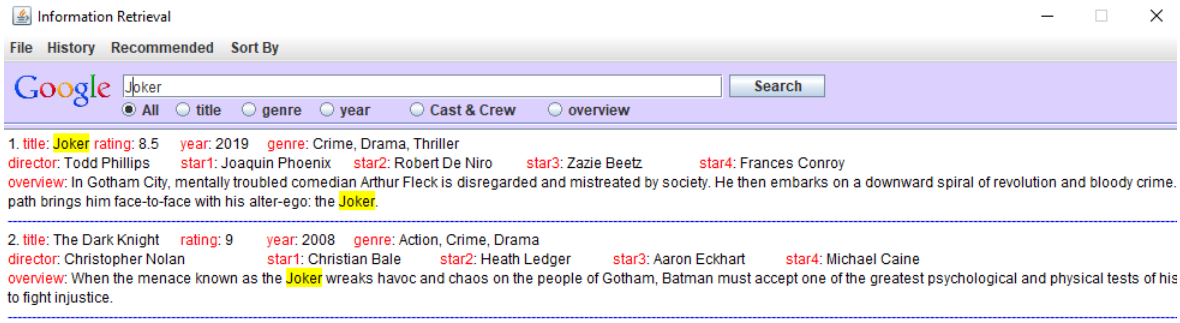
4. Αναζήτηση: Το στάσιο αυτό πυροδοτείται με το πάτημα του κουμπιού “search”. Πιο μπροστά ο χρήστης έχει πληκτρολογήσει το επιθυμητό query.

   Η κλάση που είναι υπεύθυνη για την αναζήτηση είναι η “SearchCommand”. Μετά από κάθε αναζήτηση, μέσω αυτής της κλάσης γίνεται και η κλήση για ανανέωση του παραθύρου ιστορικού.

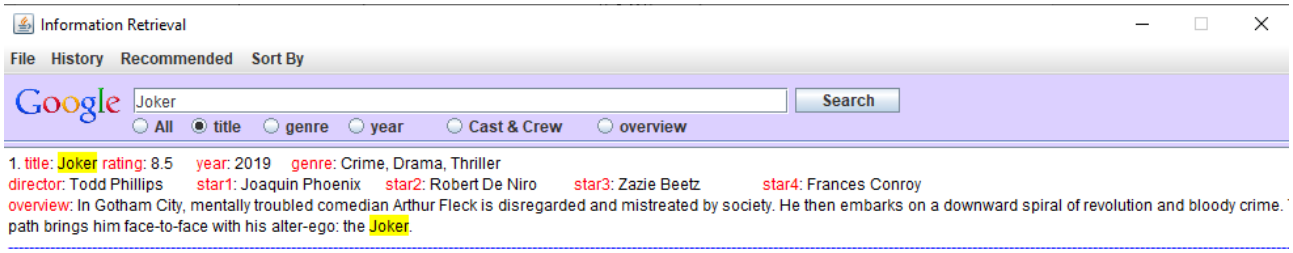
```
if(!myFrame.getHistory().contains(myFrame.getQ()))
{
        myFrame.setHistory(myFrame.getQ());
        myFrame.getWindow().refresh();
}else if(myFrame.getHistory().contains(myFrame.getQ()))
{
        myFrame.getHistory().remove(myFrame.getQ());
        myFrame.setHistory(myFrame.getQ());
        myFrame.getWindow().refresh();
}
```

   [2]Το κομμάτι της αναζήτησης λαμβάνει υπόψιν του το JRadioButton το οποίο είναι επιλεγμένο κάθε φορά. Εάν δεν είναι επιλεγμένο κανένα ή είναι επιλεγμένο το “All” η αναζήτηση θα γίνει σε όλα τα πεδία των Document του ευρετηρίου. Αλλιώς θα περιοριστεί στο πεδίο που έχει επιλεγεί κάθε φορά.

Π.χ. ας υποθέσουμε ότι έχουμε το query “Joker”. Θα πάρουμε διαφορετικά αποτελέσματα αναζήτησης ψάχνοντας στο “All” και διαφορετικό μόνο στο “Title” .



Σχήμα 3: Αποτελέσματα για το Query “Joker” σε όλα τα πεδία των Document του ευρετηρίου



Σχήμα 4: Αποτελέσματα για το Query “Joker” στο πεδίο “Title” των Document του ευρετηρίου

Παρατηρούμε λοιπόν ότι έχουμε μόνο ένα αποτέλεσμα ταίνιας με το όνομα “Joker”. Ενώ στην περίπτωση που δεν περιορίσουμε την εμβέλεια της αναζήτησής μας έχουμε δύο αποτελέσματα, καθώς ο όρος “Joker” εμφανίζεται και στην περίληψη της ταινίας “The Dark Knight”.

5. QParser: Η συγκεκριμένη κλάση είναι υπεύθυνη για την εύρεση των εγγράφων Document που περιέχουν τις λέξεις κλειδιά του query. Επιστρέφει μία λίστα από Documents που ανταποκρίνονται στο search. Στην περίπτωση που η λίστα αυτή είναι κενή, η “SearchCommand” θα κάνει κλήση στην “QParser” για “fuzzy search”. Δηλαδή μιας και δεν έχουμε Document που περιέχουν έστω και έναν από τους όρους του query αυτολεξεί, ψάχνουμε για Document που να περιέχουν όρους, των οποίων η “απόσταση” να είναι ένα (1 γράμμα διαφορά) από τον αρχίκο επιθυμητό όρο. Ο αλγόριθμος που χρησιμοποιήσαμε για την επίτευξη αυτής της “fuzzy” αναζήτησης είναι η απόσταση Levenshtein (Levenshtein distance).

```
Query fq = new FuzzyQuery(new Term(inField,query),1);
```

Το δεύτερο όρισμο του constructor new Term μπορεί να τροποποιηθεί ανάλογα με την απόσταση που επιτρέπουμε από την αρχική λέξη. Θεωρήσαμε ότι απόσταση ίση με 1 είναι υπερ- αρκετή για να προβλέψουμε τυχόν λάθη κατά την πληκτρολόγηση του query από τον χρήστη. Μεγαλύτερη απόσταση δίνει αποτελέσματα που αποκλίνουν πολύ από την αρχική αναζήτηση και η πιθανότητα συνάφειας με αυτήν είναι μικρή.

6. Αποτελέσματα: Αφού πλέον επιστραφεί η λίστα για το search, εμφανίζονται τα αποτελέσματα στην προβλεπόμενη περιοχή του GUI. Υπεύθυνη για την λειτουργία αυτή είναι η μέθοδος "refRes" της "MyFrame". Ο κώδικας της οποίας είναι αρκετά μεγάλος για να συμπεριληφθεί στην αναφορά. Αξίζει όμως να αναφερθεί ότι έχουν ληφθεί μέτρα προς αποφυγή επαναλαμβανόμενης εμφάνισης του ίδιου document. Κάποιες ταινίες μπορεί να περιλαμβάνουν ένας όρο πολλές φορές. Ωστόσο κρίναμε πως δεν πρέπει να εμφανίζονται πολλαπλές φορές (όσες και ο αριθμός εμφάνισης του όρου μέσα στο document).

7. Ιστορικό: Το ιστορικό βρίσκεται σε ένα ξεχωριστό παράθυρο από αυτό του GUI. Η κλάση που απευθύνεται σε αυτό είναι η "HistoryWindow". Για να το δει κανείς αρκεί να πατήσει

   - History -> Show History

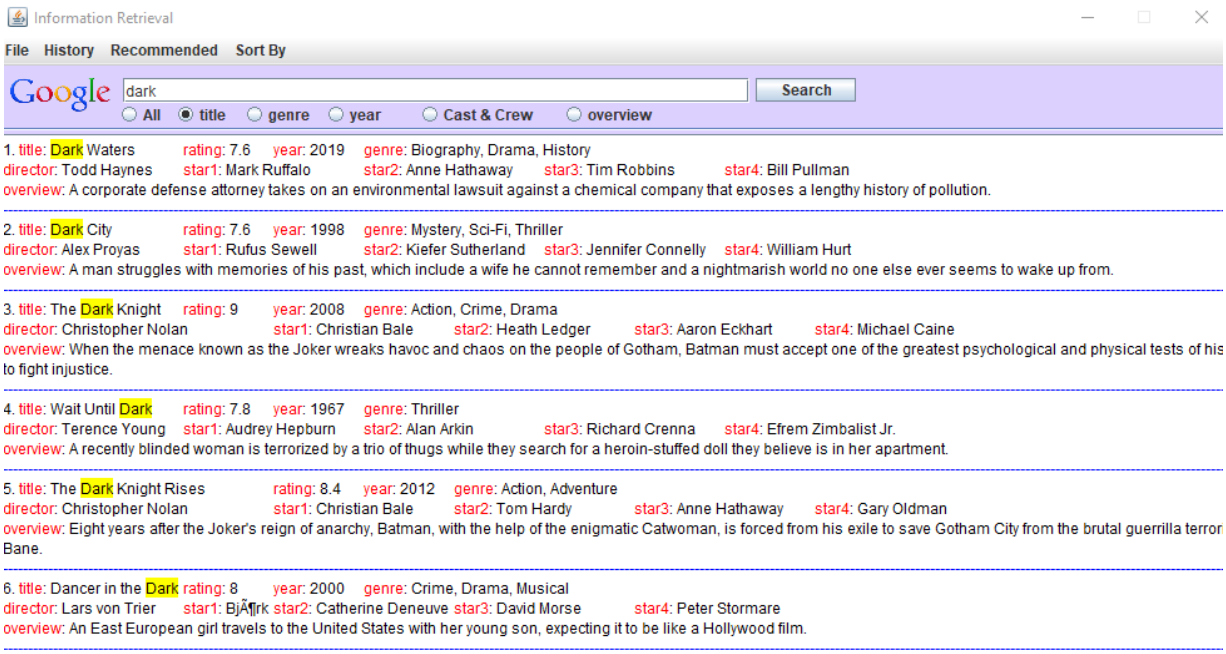
   Το παράθυρο αυτό ανανεώνεται σε πραγματικό χρόνο. Πολλαπλά επαναλαμβανόμενα searches για τον ίδιο όρο δεν καταγράφονται ξανά και ξανά. Ωστόσο εάν ένας όρος υπάρχει ήδη πιο παλιά, επανατοποθετείται στην κορυφή του ιστορικού. Την ιδέα αυτή την αντλήσαμε από τον τρόπο με τον οποίο το Google Chrome μεταχειρίζεται το ιστορικό επίσκεψης των ιστοσελίδων. Σε αυτό, επαναλαμβανόμενες επισκέψεις στην ίδια σελίδα, η μία μετά την άλλη, δεν καταγράφονται ξανά και ξανά.

8. Προτάσεις αναζήτησης:
   - Recommended -> Show Recommendations

   Προσεγγίσαμε τον τρόπο με τον οποίο δουλεύουν οι προτάσεις, που χρησιμοποιούν οι μεγάλες ιστοσελίδες όπως π.χ. Youtube. Ωστόσο σε αυτές υπάρχει κάποιο "confirmation" για το τι έψαχνε όντως ο χρήστης. Π.χ. στο Youtube η επιβεβαίωση, μετά την αναζήτηση, έρχεται με την επιλογή συγκεκριμένου βίντεο για προβολή. Για τα δεδομένα της άσκησης, σε εμάς η διαδικασία αυτή ολοκληρώνεται με την προβολή αποτελεσμάτων ταινίων. Δεν υπαρχεί δυνατότητα επιλογής της συγκεκριμένης ταινίας που είχε ο χρήστης στο μυαλό του κατά την αναζήτηση. Επομένως το δείγμα που χρησιμοποιούμε

για τις προτάσεις είναι ευρύ και σίγουρα όχι στοχευμένο.

Για παράδειγμα, έστω ότι ο χρήστης είχε στο μυαλό του την ταινία "The Dark Knight". Και κάνει ένα γρήγορο search του όρου "dark".



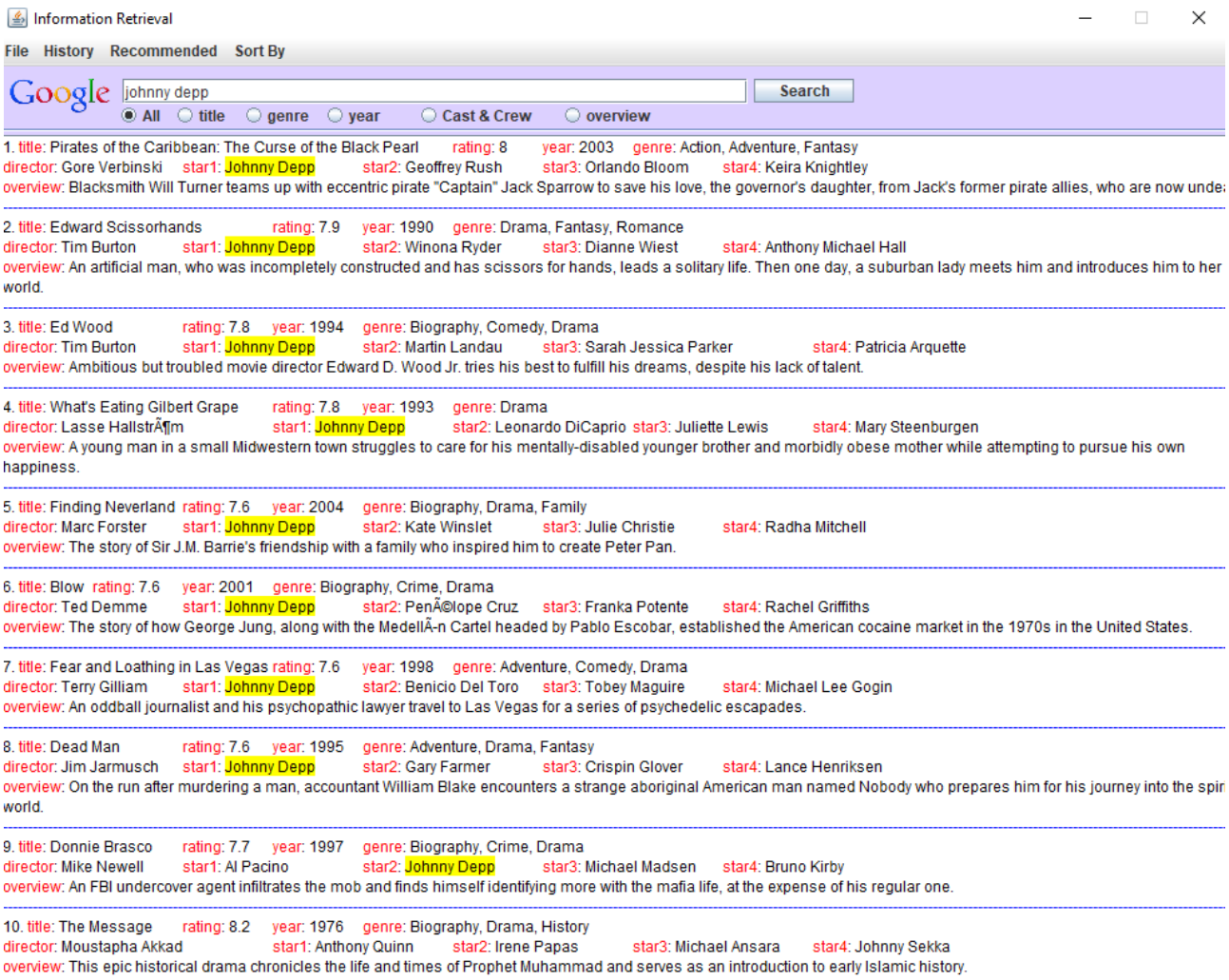
Σχήμα 5: Αποτελέσματα για το Query "dark"

Ωστόσο βλέπουμε ότι υπάρχουν αρκέτες ταινίες με τον όρο "dark" στο τίτλο τους. Στα πλαίσια του προγράμματός μας δεν υπάρχει τρόπος, να δοθεί feedback από τον χρήστη στο πρόγραμμα για τις προθέσεις του. Έτσι δεν γίνεται να περιοριστεί το δείγμα προτάσεων μόνο βασιζόμενο στην κατηγορία, ηθοποιούς κ.α. της ταινίας που είχε ο χρήστης στο μυαλό του.

(Για τα δεδομένα του παραδείγματος, βασιζόμενοι στο ότι ο χρήστης έψαχνε την ταινία "The Dark Knight", οι προτάσεις θα έπρεπε να είναι "Action", "Crime", "Drama" για τις κατηγορίες και "Christian Bale", "Heath Ledger", "Aaron Eckhart", "Michael Caine" για τους ηθοποιούς). Ωστόσο η υλοποίηση μας περιλαμβάνει όλες τις κατηγορίες/ ηθοποιούς για τα αποτελέσματα του query "dark" καθώς όπως αναφέρθηκε δεν υπάρχει το feedback "αυτό έψαχνα" από τον χρήστη όπως σε σελίδες σαν το Youtube. Η κλάση που

συμπεριλαμβάνει τις λεπτομέρειες για το κομμάτι των προτάσεων αναζήτησης είναι η “RecommendationsWindow”.

9. Highlight: Τονισμός των λέξεων κλειδιά στα αποτελέσματα. Η μέθοδος υπεύθυνη είναι η “highlightTxt” της κλάσης “MyFrame”. Λαμβάνει υπόψιν της τα αποτελέσματα που υπάρχουν στο JTextPane και υπογραμμίζει με κίτρινο χρώμα τους αυτολεξεί όρους, όπως στο query. Για παράδειγμα:



Σχήμα 6: Αποτελέσματα για το Query “johnny depp” σε όλα τα πεδία των Document του ευρετηρίου

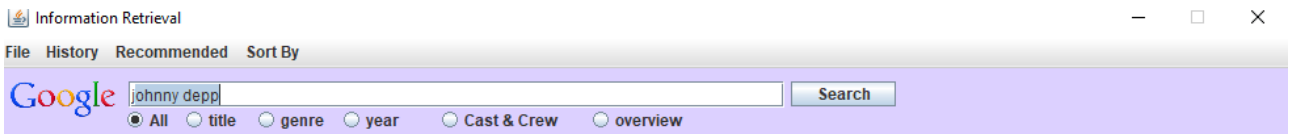
Βλέπουμε ότι μόνο οι ταινίες που έχουν ολόκληρο το query “johnny depp” (ανεξαρτήτως κεφαλαίων/ μικρών γραμμάτων) έχουν υπογράμμιση. Ενώ στο 10ο αποτέλεσμα ο ηθοποιός “Johnny Sekka” δεν έχει υπογράμμιση παρόλο που μοιράζονται το ίδιο όνομα “johnny”.

10. Sorting: Ταξινόμηση αποτελεσμάτων με βάση το έτος κυκλοφορίας/ βαθμό κριτικών.
    - Sort By -> Year
    - Sort By -> Rating

Η μέθοδος υπεύθυνη είναι η sortRes της κλάσης “MyFrame”.

Extra

Το πρόγραμμα έχει δυνατότητες auto-completion. Ωστόσο αυτό βασίζεται μόνο στο ιστορικό όρων που έχουν αναζητηθεί, όχι σε όλους τους όρους που υπάρχουν μέσα στο ευρετήριο. Για παράδειγμα έχοντας ψάξει ήδη μία φορά “Johnny Depp”, από τη δεύτερη και μετά γράφοντας ‘J’ στο search box το πρόγραμμα θα έχει την τάση για auto- completion.



Σχήμα 7: Στιγμιότυπο auto-complete

Παρατηρήσεις

- Προκειμένου να έχουμε τον απόλυτο έλεγχο πάνω στην εμφάνιση του GUI θέσαμε το layout του “null”.

```
this.setLayout(null); //Setting this to null disables any layout "templates". We will have to give each frame component its coordinates manually.
```

Με αυτό τον τρόπο βάλαμε όπου θέλαμε έμεις τα components του. Ωστόσο αυτό είχε ως αποτέλεσμα οι μπάρες του JScrollPane να μην εμφανίζονται. Παρ’ όλα αυτά είναι και πάλι δυνατό να διατρέξει κάποιος τα αποτελέσματα κάνοντας click μέσα στην περιοχή του JTextPane και σύροντας το ποντίκι προς την κατεύθυνση που επιθυμεί.

Επίσης κάποιες φορές ίσως να μην εμφανίζεται η μπάρα JMenu των βασικών λειτουργίων του GUI. Δοκιμάζοντας να επανεκινήσουμε το πρόγραμμα αυτό διορθώνεται.

Για περισσότερες λεπτομέριες για την υλοποίηση του προγράμματος παρέχονται αναλυτικά σχόλια στον κώδικα.