ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

2020

1η φάση εργασίας

Ελένη Μουζάκη 3280

Παναγιώτης Παπαϊωάννου 3309

Συλλογή εγγράφων:

Σε αυτή την εργασία θα δημιουργήσουμε μια μηχανή αναζήτησης ταινιών. Θα συλλέξουμε άρθρα από τη Wikipedia σχετικά με ταινίες. Μέσω του Kaggle.com κατεβάσαμε το αρχείο 'wiki_movie_plots_deduped.csv' από το οποίο θα δημιουργήσουμε το database μας. Το αρχείο αυτό περιέχει πληροφορίες σχετικά με τις ταινίες (release year, title, origin/ethnicity, director, cast, genre, plot) αλλά και το Wikipedia URL τους.

Στο ίδιο αρχείο έχουμε προσθέσει 2 στήλες πληροφοριών για την κάθε ταινία η οποίες περιλαμβάνουν το Rating Value(αξιολόγηση στα 10) και Rating Count(αριθμός κριτικών) τους. Αυτές τις πληροφορίες τις συλλέξαμε από το IMDB.com μέσω Beautiful Soup και θα τις χρησιμοποιήσουμε στη συνέχεια.

Γράψαμε πρόγραμμα σε γλώσσα Python, το οποίο μας βοήθησε να αποθηκεύσουμε το περιεχόμενο των links στο δίσκο σε μορφή '.mhtml' . Έτσι όταν ο χρήστης θα επιλέγει ποιο άρθρο θέλει να διαβάσει, θα του ανοίγει ο browser στην συγκεκριμένη σελίδα της Wikipedia.

Έχουμε ήδη συλλέξει 10 άρθρα στο database μας:

Σκοπεύουμε μέχρι την παράδοση της εργασίας να έχουμε συλλέξει 5.000 τέτοια αρχεία.

Ανάλυση κειμένου και κατασκευή ευρετηρίου:

Η μονάδα εγγράφου μας θα είναι η γραμμή με βάση το αρχείο ‘wiki_movie_plots_deduped.csv’.

Τα fields κάθε εγγράφου θα είναι:

- Release Year
- Title
- Origin/Ethnicity
- Director
- Cast
- Genre
- Plot
- Rating Value.

Για κάθε field θα δημιουργηθεί ένα ανεστραμμένο ευρετήριο (inverted index), έτσι ώστε ο χρήστης να μπορεί να αναζητά άρθρα με βάση την ημερομηνία, τον τίτλο, την εθνικότητα, τον σκηνοθέτη, το είδος, τις λέξεις κλειδιά στην πλοκή και την αξιολόγηση, αλλά και συνδυασμό αυτών με Boolean queries.

Για να κατασκευάσουμε τα IndexDocuments θα χρησιμοποιήσουμε τον StandardAnalyzer έτσι ώστε να διατηρήσουμε τα links της Wikipedia αλλά ταυτόχρονα να μικρύνουμε όλα τα γράμματα και να διαγράψουμε τα stop words. Ακόμη θα αξιοποιήσουμε την δυνατότητα που προσφέρει η Lucene για stemming.

Επιπρόσθετα θα δημιουργήσουμε ένα λεξικό το οποίο θα περιέχει ως κλειδιά το docId κάθε document και ως τιμή το αντίστοιχο άρθρο στον δίσκο.

Αναζήτηση:

Ο χρήστης θα μπορεί να θέτει ερωτήματα στην εφαρμογή γράφοντας στο search box, δημιουργώντας έτσι ένα query από τον QueryParser το οποίο θα κληθεί να διαχειριστεί ο IndexSearcher, ο οποίος θα επιστρέφει τα αποτελέσματα – hits.

Η ερώτηση θα μπορεί να είναι οποιαδήποτε λέξη κλειδί, θα υποστηρίζεται όμως και η δυνατότητα αναζήτησης με βάση τα fields.

Παραδείγματα αναζητήσεων:

- Tom Hanks
- Cast: tom hanks
- Plot: moon
- Rating: >5
- Release date: 1999 + cast: tom hanks
- tom hanks AND Julia Roberts

Επιπλέον θα διατηρείται μία λίστα, η οποία θα αποτελεί το search history του χρήστη και θα ανανεώνεται με κάθε κλήση του συστήματος. Η λίστα αυτή θα χρησιμοποιηθεί για να ανακτηθούν οι συχνότερες ερωτήσεις FAQ τις οποίες θα μπορεί να δει ο χρήστης σε κάποιο μέρος της οθόνης.

Θα χρησιμοποιήσουμε επίσης embedding για τη βελτίωση της αναζήτησης.

Ακόμη θα υπάρχει ένα help button το οποίο ο χρήστης θα μπορεί να πατάει για να ενημερώνεται σχετικά με το πώς θα μπορεί να θέτει advanced ερωτήματα στην μηχανή αναζήτησης.

Παρουσίαση Αποτελεσμάτων:

Για να διατάξουμε τα αποτελέσματα της αναζήτησης στον χρήστη, δηλαδή να παρουσιάσουμε τα *TopDocs* θα αναθέσουμε σε κάθε ζεύγος ερωτήματος-εγγράφου ένα *score(d,q)*. Το score αυτό θα μετρά το πόσο συναφές είναι το έγγραφο με την ερώτηση. Το έγγραφο με το μεγαλύτερο score θα εμφανίζεται 1ο το αμέσως επόμενο 2ο κτλ.

Η μορφή στην οποία θα βλέπει ο χρήστης τα αποτελέσματα θα είναι μια λίστα από 10 links τα οποία θα τον οδηγούν στα αποθηκευμένα στον δίσκο άρθρα. Εάν θέλει να δει περισσότερα από 10 θα μπορεί να μεταβεί στα επόμενα 10 στην κατάταξη με κάποιο βελάκι.

Κάθε λέξη η οποία «ταίριαξε» με την αναζήτηση θα υπογραμμίζεται.

Θα υπάρχει επιλογή διαφορετικής διάταξης των αποτελεσμάτων: χρονολογική σειρά, με βάση το Rating Value και ο συνδυασμός των 2.