Μηχανή Αναζήτησης

Βεσέλ Νταλίπαϊ

Link: https://github.com/veselDalipaj/IR

Ανάκτηση Πληροφοράς

Πανεπιστήμιο Δυτικής Αττικής

Μηχανή Αναζήτησης

Σαν πρώτο βήμα επέλεξα να αναπτύξω έναν web crawler χρησιμοποιώντας τη βιβλιοθήκη BeautifulSoup, όπως προτάθηκε από το μάθημα, για τη συλλογή δεδομένων από την Wikipedia. Για κάθε link που εντόπιζα, εξήγαγα τον τίτλο του άρθρου, το url του και φυσικά τα δεδομένα του άρθρου, παράγραφο προς παράγραφο. Ο τίτλος και το url εξήχθησαν με σκοπό τη χρήση τους στη διεπαφή χρήστη αργότερα στο project. Επέλεξα τη χρήση αρχείου JSON για την αποθήκευση των δεδομένων, καθώς υποστηρίζει την αποθήκευση πολλών πεδίων και είναι αρκετά εύκολο στη φόρτωση και αποθήκευση δεδομένων. Ο web crawler έχει παραμετροποιηθεί ώστε να μπορεί να αλλάξει εύκολα το αρχικό url από το οποίο ξεκινά η αναζήτηση, καθώς και τον αριθμό των άρθρων που συλλέγονται, μέσω απλής τροποποίησης του κώδικα. Στο τέλος του script αποθηκεύουμε τα δεδομένα μας για περαιτέρω χρήση.

Στη συνέχεια, στην προεπεξεργασία κειμένου πραγματοποιήθηκαν εργασίες για τον καθαρισμό των δεδομένων, τη μείωση του θορύβου και τη μείωση των παραλλαγών των λέξεων. Ξεκινήσαμε διαχωρίζοντας τις λέξεις κάθε άρθρου (tokenization) και μετατρέψαμε όλα τα γράμματα σε πεζά για να διευκολύνουμε την επεξεργασία. Αφαιρέσαμε όλα τα αγγλικά stopwords, τους αριθμούς και όλους τους χαρακτήρες που δεν είναι αλφαριθμητικοί για τη μείωση του θορύβου. Επιλέξαμε τη χρήση lemmatization αντί για stemming, λόγω της μεγαλύτερης ακρίβειάς του, καθώς μας διασφαλίζει ότι κάθε λέξη θα είναι στην πιο απλή και βασική της μορφή.

Για τη δημιουργία του ανεστραμμένου ευρετηρίου αποφάσισα να χρησιμοποιήσω ένα λεξικό για την αποθήκευση των δεδομένων, καθώς το λεξικό εγγυάται ότι κάθε λέξη του ευρετηρίου θα είναι μοναδική και προσφέρει μεγάλη ταχύτητα. Το ευρετήριο αποθηκεύει κάθε λέξη, το άρθρο στο οποίο εμφανίστηκε (ξεκινώντας την αρίθμηση από το 1 για να είναι πιο ευανάγνωστο) και πόσες φορές εμφανίστηκε σε κάθε άρθρο.

Στο τέταρτο βήμα ανέπτυξα πρώτα το backend κομμάτι της μηχανής αναζήτησης. Δημιουργήθηκε μια συνάρτηση για τη φόρτωση των τριών αρχείων που δημιουργήθηκαν στα προηγούμενα βήματα και τα επιστροφή στο interface για περαιτέρω χρήση. Επιπλέον, υλοποιήθηκε μια συνάρτηση για την προεπεξεργασία των ερωτημάτων του χρήστη, ακολουθώντας τη διαδικασία του βήματος 2, ώστε να είναι συμβατή με τα ίδη επεξεργασμένα δεδομένα. Στη συνέχεια, αναπτύχθηκε η λειτουργικότητα της Boolean αναζήτησης με τη χρήση του ανεστραμμένου ευρετηρίου και με βοήθεια συναρτήσεων της set για την επιστροφή σχετικών λιστών. Για την υλοποίηση του αλγόριθμου TF-IDF χρησιμοποιήθηκαν έτοιμες συναρτήσεις της βιβλιοθήκης sklearn για τη μορφοποίηση των δεδομένων και η numpy για τον υπολογισμό των scores και την ταξινόμηση. Τέλος, για την υλοποίηση του BM25, αξιοποιήθηκε η βιβλιοθήκη rank_bm25 σε συνδυασμό με την numpy για την ταξινόμηση πάλι των αποτελεσμάτων.

Για το interface αποφάσισα να αναπτύξω μια διεπαφή σε Python, χρησιμοποιώντας τη βιβλιοθήκη ipywidgets και τη γλώσσα HTML. Δημιουργήθηκε ένα text box, όπου ο χρήστης μπορεί να εισάγει τα ερωτήματά του, καθώς και δύο drop-down menus: το πρώτο για την επιλογή του operator στην περίπτωση Boolean αναζήτησης και το δεύτερο για την επιλογή του αλγορίθμου. Τέλος, προστέθηκε ένα button, το οποίο μόλις το πατήσουμε επεξεργάζετε το ερώτημα με την συνάρτηση που αναπτύξαμε στο backend και κάνει την αναζήτηση που ζητήθηκε. Στην περίπτωση που ο χρήστης επιλέξει Boolean αναζήτηση, τα αποτελέσματα

εμφανίζονται ως τίτλοι των άρθρων που εντοπίστηκαν, με δυνατότητα να μεταφερθεί στο άρθρο της wikipedia. Αντίστοιχα, για τους αλγορίθμους TF-IDF ή BM25, εμφανίζονται και τα scores των αποτελεσμάτων ταξινομημένα.

Λόγω της κακής διαχείρισης του χρόνου μου, δεν κατάφερα να τελειώσω το συγκεκριμένο και το έκανα κάπως πρόχειρα. Δεν πρόλαβα να σχολιάσω στο Jupyter Notebook το interface και δεν είμαι σίγουρος για τη λειτουργικότητα του βήματος 5, καθώς δεν πρόλαβα να το δοκιμάσω με αρκετά παραδείγματα. Ζητώ συγγνώμη εκ των προτέρων.