Ανάκτηση Πληροφορίας

Παπαποστόλου Βασίλειος 3176

Κώστογλου Σοφία 3114

Στην εργασία υλοποιείται μία μηχανή αναζήτησης, η οποία αποτελείται από τα υποσυστήματα του Crawler όπου γίνεται scrape στο web και εξάγονται κάποια urls με βάση το αρχικό που έχει δώσει ο χρήστης, του Indexer ο οποίος επεξεργάζεται το vocabulary των εγγράφων, δουλεύει για την αγγλική γλώσσα, και κατασκευάζει τον ανεστραμμένο κατάλογο, και τέλος του Query Processor ο οποίος είναι αρμόδιος για τα ερωτήματα του χρήστη στη μηχανή, και υλοποιεί το cosine similarity ώστε να επιστραφούν τα πιο σχετικά urls.

Crawler: Η κλάση Crawler αρχικοποιεί το url με το οποίο θα ξεκινήσουμε, το πλήθος των threads, το πλήθος των σελίδων που θέλουμε να κάνει crawl και μία τιμή flag η οποία διαγράφει ή διατηρεί τα έγγραφα που έχουμε κάνει ήδη scrape. Πρέπει να τονιστεί ότι όταν καλέσουμε για πρώτη φορά τον Crawler μας τότε flag = True δηλαδή «διατηρεί» τα δεδομένα καθώς δεν υπάρχει κάτι να διαγράψει.

Η πρώτη λειτουργία του Crawler είναι να κάνει scrape το πρώτο link που μας έδωσε ο χρήστης και μέσα από αυτό θα κινηθούμε στα υπόλοιπα links τα οποία τα εισάγουμε σε μια δομή δεδομένων Queue η οποία υλοποιεί FIFO τεχνική. Μπορούμε να πούμε ότι ψάχνουμε κατά βάθος καθώς κινούμαστε από σελίδα σε σελίδα και το πρώτο link που βρίσκουμε το βάζουμε στην σειρά ώστε να το κάνουμε scrape .

Το scrape των links και το scrape του περιεχομένου της σελίδας γίνεται παράλληλα. Για να γίνει κατανοητό χρονικά, αρκεί να σκεφτούμε ότι σε μία σελίδα υπάρχουν πολλά links αλλά ένα μόνο περιεχόμενο. Τις δύο αυτές λειτουργίες (parse_links, parse_info) τις ενώνουμε σε μία συνάρτηση (post_callback) που είναι η callback function του Thread pool μας. Επίσης για κάθε σελίδα που κάνουμε scrape έχουμε sleep για 0.1 δευτερόλεπτα και κατά την ολοκλήρωση έχουμε επιπλέον καθυστέρηση 5 δευτερολέπτων για να γραφτούν στην μνήμη όλα τα αποτελέσματα.

Τα περιεχόμενα των σελίδων μαζί με τα url τους και την αρίθμησή τους αποθηκεύονται σε μία λίστα αντικειμένων Docs όπου η κλάση Docs έχει 3 ορίσματα: url, περιεχόμενο, αρίθμηση. Στην αρχή τα γράφαμε σε ένα αρχείο txt αλλά ήταν κατά 10-15 δευτερόλεπτα πιο αργό.

Τέλος στην κλάση ελέγχουμε αν όλες οι σελίδες έχουν response, αποφεύγουμε τις φωτογραφίες ή οποιοδήποτε φίλτρο καθώς μία σελίδα μπορεί να κάνει scrape των εαυτό τις 100 φορές με διαφορετικές παραμέτρους φίλτρων.

Indexer: Στην κλάση του indexer δημιουργείται ο ανεστραμμένος κατάλογος, με βάση τα έγγραφα που έχουμε αποθηκεύσει στο txt αρχείο, που αποτελείται από το κείμενο των σελιδών που κάναμε crawling. Αρχικά, στη συνάρτηση tokenize, παίρνουμε σαν string το περιεχόμενο κάθε εγγράφου, που αντιστοιχεί σε μία σειρά του αρχείου, και αφαιρούμε τα σημεία στίξης. Έπειτα αφαιρούμε όλα τα ψηφία, μετατρέπουμε όλα τα γράμματα σε μικρά και τέλος κάνουμε tokenize κάθε λέξη.

Στη συνάρτηση delete_stop_words σβήνουμε όλα τα stop words (ισχύει για την αγγλική γλώσσα) με βάση ένα λεξικό που υπάρχει στην python.

Στη συνάρτηση lemmatization, προσπαθούμε να μετατρέψουμε κάθε λέξη στη ρίζα της, ώστε να αποφύγουμε να χρησιμοποιηθούν στο λεξικό ίδιες λέξεις απλά σε άλλο αριθμό, πτώση, χρόνο ή πρόσωπο.

Στη συνέχεια χρησιμοποιείται η συνάρτηση get_tf, ώστε να υπολογιστεί ο όρος tf του κάθε token, δηλαδή λέξης, σύμφωνα με τον τύπο. Όσον αφορά τη συνάρτηση get_idf, υπολογίζεται ο όρος idf, και έπειτα υπολογίζεται ο tfidf, ο οποίος θα χρησιμοποιηθεί στο cosine similarity στον Query processor.

Έτσι, όλα είναι έτοιμα για την κατασκευή του ανεστραμμένου καταλόγου, ο οποίος μας δείχνει για κάθε λέξη που βρίσκεται στο λεξικό μας, σε ποια έγγραγα εμφανίζεται. Πράγμα που καθιστάται πολύ χρήσιμο έπειτα, στον Query processor, όταν θα χρειαστεί να βρούμε εύκολα και γρήγορα ποια documents θα χρειαστούμε βάσει του ερωτήματος του χρήστη.

Query: Στον Query παίρνουμε όλα τα αποτελέσματα από τον Indexer. Αρχικά δεχόμαστε το input του χρήστη. Υπολογίζουμε το tf του για το οποίο δεν χρειαζόμαστε καμία πληροφορία. Για τον υπολογισμό του idf επικαλούμαστε τον πίνακα idf του Indexer. Στην συνέχεια ελέγχουμε ποια έγγραφα περιέχουν μία από τις λέξεις του query. Αν το πλήθος των λέξεων στο query είναι περισσότερες από μία τότε βρίσκουμε την ένωση των εγγράφων.

Τέλος είναι ο υπολογισμός του tf-idf και με τα τελικά διανύσματα υπολογίζουμε το cosine similarity του query με καθένα από τα σχετικά έγγραφα. Ο χρήστης έχει την επιλογή να δει τα k καλύτερα έγγραφα.

Ο τύπος της ανάδρασης δεν υλοποιήθηκε. Απλά μπορεί να επιλέξει ο χρήστης τα πιο σχετικά έγγραφα.

Ενδεικτικά κάποιοι χρόνοι εκτέλεσης:

	Χρόνος	Χαρακτήρες
Crawler	22 s	550.000
Indexer	30 s	550.000
Query	0.01s	100 docs

Παράδειγμα εκτέλεσης:

Κατά την ανάπτυξη του προγράμματος δουλεύαμε με τις παρακάτω παραμέτρους:

http://bbc.com 100 1 8

```
Give your crawl search: http://bbc.com 100 1 8
Scraping URL: 1 http://bbc.com
Scraping URL: 2 http://bbc.com/news/world-us-canada-56141673
Scraping URL: 3 http://bbc.com/news/world/us and canada
Scraping URL: 4 http://bbc.com/news/world-us-canada-56144794
Scraping URL: 5 http://bbc.com/news/world-europe-55986606
Scraping URL: 6 http://bbc.com/news/world/europe
Scraping URL: 7 http://bbc.com/sport/live/football/55864692
Scraping URL: 8 http://bbc.com/sport/football
Scraping URL: 9 http://bbc.com/news
Scraping URL: 10 http://bbc.com/news/live/uk-56144834
Scraping URL: 11 http://bbc.com/news/uk
Scraping URL: 12 http://bbc.com/news/world-asia-56144904
Scraping URL: 13 http://bbc.com/news/world/asia
Scraping URL: 14 http://bbc.com/news/world-middle-east-56147305
Scraping URL: 15 http://bbc.com/news/world/middle east
Scraping URL: 16 http://bbc.com/sport
Scraping URL: 17 http://bbc.com/sport/football/56146886
Scraping URL: 18 http://bbc.com/sport/football/european
Scraping URL: 19 http://bbc.com/sport/football/56033699
Scraping URL: 20 http://bbc.com/sport/football/56147763
Scraping URL: 21 http://bbc.com/reel
Scraping URL: 22 http://bbc.com/reel/video/p0976j3x/princess-latifa-the-dub
```

Ο χρήστης αναζήτησε τα εξής: «trump biden»

```
What do you want to search Sir? trump biden
```

Παρακάτω μπορούμε να δούμε δύο διανύσματα για δύο διαφορετικά έγγραφα. Στο πρώτο έγγραφο δεν περιέχεται η λέξη biden γι’ αυτό έχει και 0 ενώ στο δεύτερο περιέχεται.

```
[0.000973520959229775, 0] [0.37018134474712194, 0.6215190243431472]
[0.002399879058328181, 0.0020146483771252745] [0.37018134474712194, 0.6215190243431472]
```

Ο χρήστης επέλεξε τα 10 καλύτερα αποτελέσματα:

```
1 :  https://www.bbc.com/somali
2 :  https://www.bbc.com/mundo
3 :  https://www.bbc.com/news/world/us and canada
4 :  https://www.bbc.com/news/science and environment
5 :  https://www.bbc.com/swahili
6 :  https://www.bbc.com/azeri
7 :  https://www.bbc.com/news/world-us-canada-56141673
8 :  https://www.bbc.com/news/world-us-canada-56144794
9 :  https://www.bbc.com/news/world-europe-55986606
10 :  https://www.bbc.com/news/world-asia-56144904
You want to continue? Yes/No
```

Η μηχανή ρωτάει τον χρήστη αν θέλει να συνεχίσει το crawl/search προκειμένου να βάλει και άλλα έγγραφα.

Τα αποτελέσματα που βγάζει έχουν ελεγχθεί και περιέχουν όντως τις λέξεις που ρωτάει ο χρήστης στα πρώτα αποτελέσματα ενώ σε μερικά από το τέλος υπάρχει μόνο μία από τις 2.