ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

ΑΝΑΠΤΥΞΗ ΜΗΧΑΝΗΣ ΑΝΑΖΗΤΗΣΗΣ ΣΕ PYTHON

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΠΑΡΑΣΚΕΥΗ ΠΑΠΑΓΕΩΡΓΙΟΥ
ΑΕΜ: 3177
ΕΞΑΜΗΝΟ: 7^{Ο}
ΜΑΘΗΜΑ: ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

Αυτή η εργασία εκπονήθηκε στα πλαίσια του μαθήματος Ανάκτηση Πληροφορίας ατομικά. Η υλοποίηση της περιλαμβάνει μια μηχανή αναζήσης.

Για την εργασία έχει χρησιμοποιηθεί η γλώσσα Python 3.8.9 σε υπολογιστή με λειτουργικό σύστημα Windows 10. Με την εντολή pip install virtualenv εγκαθιστούμε τα πακέτα για τη δημιουργία virtual environment στη Python. Στη συνέχεια, τρέχουμε την εντολή & "c:project-folder/search-engine-assignment/env/Scripts/Activate.ps1" για την εκκίνηση του virtual environment.

.\env\Scripts\pip3.8.exe install -r .\requirements.txt

ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΚΑΙ ΤΩΝ ΑΠΑΙΤΗΣΕΩΝ

Σκοπός της εργασίας είναι ο σχεδιασμός και υλοποίηση μίας απλής μηχανής αναζήτησης, η οποία όμως θα έχει όλη τη βασική λειτουργικότητα μίας μηχανής αναζήτησης μεγάλης κλίμακας.
Η μηχανή αναζήτησης περιλαμβάνει τα έχει τα εξής υποσυστήματα:

- Crawler: αποτελεί το υποσύστημα που είναι υπεύθυνο για τη συλλογή των δεδομένων από το web. Ο crawler εκτελείται σαν ξεχωριστή process στο background και μαζεύει κείμενο από τις ιστοσελίδες. Στον crawler δίνουμε παράμετρο το URL αφετηρία και επίσης έναν ακέραιο αριθμό που δηλώνει πόσες σελίδες θέλουμε να κάνουμε crawling. Εάν ξεκινήσουμε πάλι τον crawler τα δεδομένα που είχαμε συλλέξει σε προηγούμενη εκτέλεση διατηρούνται εκτός και αν με μία παράμετρο δηλώσουμε ότι θέλουμε να ξεκινήσουμε από την αρχή. Επίσης, μία άλλη παράμετρος στον crawler θα μπορούσε να είναι το πλήθος των threads που θα χρησιμοποιηθούν. Προφανώς, μπορείτε να παραμετροποιήσετε τον crawler με επιπλέον παραμέτρους ανάλογα με τις ανάγκες σας (π.χ., αν θα εκτελέσει BFS, DFS ή κάποιον υβριδικό αλγόριθμο graph search). Ακολουθεί παράδειγμα εκτέλεσης του crawler ο οποίος ξεκινά από τη σελίδα http://mypage.gr θα συγκεντρώσει τα στοιχεία από 200 σελίδες, θα διατηρήσει τα προηγούμενα δεδομένα και θα λειτουργήσει με 8 threads:
  crawler http://mypage.gr 200 1 8
- Indexer: είναι το υποσύστημα που υλοποιεί τον αντεστραμμένο κατάλογο. Θα πρέπει να αποθηκεύετε όλες τις απαραίτητες πληροφορίες ώστε να μπορείτε να υπολογίζετε την ομοιότητα cosine με TF-IDF διανύσματα. Το λεξικό του καταλόγου αποτελείται από όλα τα μοναδικά tokens που συναντούμε μέσα στο κείμενο των σελίδων που κάνουμε crawling. Ο Indexer παίρνει τις πληροφορίες από τον Crawler.

- Query Processor: είναι το υποσύστημα που είναι υπεύθυνο για την επεξεργασία ενός ερωτήματος. Ο χρήστης θα πρέπει να μπορεί να δώσει ένα ερώτημα και να ζητήσει τα top-k έγγραφα (webpages) που έχουν τη μεγαλύτερη ομοιότητα με το ερώτημα του χρήστη. Το ερώτημα διατυπώνεται χρησιμοποιώντας μία απλή html σελίδα. Τα αποτελέσματα εμφανίζονται στο χρήστη με φθίνουσα διάταξη ως προς το cosine similarity score, και περιλαμβάνουν τον τίτλο της σελίδας και το URL της σελίδας καθώς και όποια άλλη πληροφορία θέλετε να εμφανίσετε. Ο χρήστης στη συνέχεια έχει τη δυνατότητα να επιλέξει ποιά από τα αποτελέσματα είναι πιο σχετικά ώστε το σύστημα να υποστηρίξει ανάδραση

Η εργασία υλοποιηθήκε σε γλώσσα προγραμματισμού Python έκδοση 3.8.9. Για τη βάση δεδομένων NoSQL χρησιμοποιήθηκε το Docker με MongoDB.

Στα αρχεία crawler.py , query_handler και indexer.py αφορούν τα υποσυστήματα που περιγράφηκαν παραπάνω. Το αρχείο app.py περιλαμβάνει κώδικα που αφορά τον Flask server για την αλληλεπίδραση του χρήστη με τα υποσυστήματα και την υποβολή ερωτημάτων σε μια HTML σελίδα. Στο φάκελο templates βρίσκονται οι HTML σελίδες που φορτώνει ο σερβερ και στο φάκελο ./static/css βρίσκεται το αρχείο για τη μορφοποίηση των σελίδων.

ΠΕΡΙΓΡΑΦΗ ΥΛΟΠΟΙΗΣΗΣ

Η πρώτη ενέργεια που πρέπει να κάνουμε πριν τρέξουμε τη μηχανή αναζήτησης είναι να εγκαταστήσουμε τα απαραίτητα πακέτα που βρίσκονται στο αρχείο requirements.txt . Επίσης, πρέπει να δημιουργήσουμε με το Docker ένα container με τη MONGODB βάση δεδομένων που θα αποθηκεύει τα έγγραφα και τον αντεστραμμένο κατάλογο με την εντολή docker-compose up στο φάκελο που περιέχει το αρχείο για την αρχικοποίηση του docker.

Η μηχανή αναζήτησης πρέπει να αρχικοποιηθεί αφού τρέξουμε τον crawler και συλλέξει κάποια documents. Ο crawler ξεκινάει από ένα συγκεκριμένο link το οποίο θα επισκεφθεί και θα συλλέξει όλα τα link της σελίδας. Στην είσοδο, επίσης, πρέπει να δώσουμε ως ορίσματα τον αριθμό των σελίδων που θέλουμε να συλλέξει, τον αριθμό των thread που θα μπορούν να τρέχουν παράλληλα καθώς και αν θέλουμε να διαγραφούν τα παλιά documents που συλλέχθηκαν ή όχι. Σε όλα τα υποσυστήματα γίνεται χρήση Locker για την ανανέωση μεταβλητών που χρησιμοποιούνται από τα Threads για την ασφαλή ανανέωση τους. Η συνάρτηση parse θα κάνει τη πιο σημαντική εργασία του crawler. Δέχεται ως όρισμα το url και με τη χρήση της βιβλιοθήκης BeautifulSoup επιλέγουμε να κρατήσουμε τον τίτλο, το σώμα καθώς και όλους τους συνδέσμους που περιέχει η σελίδα με αναφορές σε άλλες σελίδες. Επίσης με τη χρήση της βιβλιοθήκης nltk αφαιρούμε

tokens και stopwords. Τελικά, θα μετατρέψουμε τις λέξεις στη ρίζα τους με τη χρήση της βιβλιοθήκης PorterStemmer. Εφόσον «σκανάρει» μια ιστοσελίδα, αποθηκεύονται στη βάση στη συλλογή των "crawled documents" την ηλεκτρονική διεύθυνση (url), τον τίτλο της ιστοσελίδας, καθώς και το σύνολο των λέξεων (εξαιρουμένων των σημείων στήξης και των stop words) το γνωστό bag of words όπως αναφέρεται στη βιβλιογραφία. Όταν δεν υπάρχουν άλλοι σύνδεσμοι ή έχει φτάσει ο μέγιστος αριθμός των ιστοσελίδων που ορίσαμε ότι πρέπει να κάνει crawl, τότε το πρόγραμμα του υποσυστήματος ολοκληρώνεται. Ένα παράδειγμα λειτουργίας του crawler δίνεται στην παρακάτω εικόνα.

```
PS C:\Users\VIVI PAPAGEORGIOU\Downloads\search-engine-assignment\search-engine-assignment> & "C:/Users/VIVI PAPAGEORGIOU/AppData/Local/Microsoft/WindowsApps/python.exe" "c:/Users/VIVI PAPAGEORGIOU/Downloads/search-engine-assignment/search-engine-assignment/crawler.py" https://skroutz.gr 10 1 8
Crawling...
Crawled 1 documents of 10...
Crawled 2 documents of 10...
Crawled 3 documents of 10...
Crawled 4 documents of 10...
Crawled 5 documents of 10...
Crawled 6 documents of 10...
Crawled 7 documents of 10...
Crawled 8 documents of 10...
Crawled 9 documents of 10...
Crawled 10 documents of 10...
Crawler total execution time: 6.91 secs
Creating inverted index...
Inverted Index is successfully created. Total time 54.1635188...
```

Στη συνέχεια, ο crawler δημιουργεί ένα αντικείμενο του Indexer και ξεκινάει το χτίσιμο του αντεστραμμένου καταλάγου από τα έγγραφα που συλλέχθηκαν. Δέχεται ένα όρισμα κατά την εκκίνηση του που αφορά τον αριθμό των threads που θα τρέχουν παράλληλα για τη δημιουργία του καταλόγου.

Η δημιουργία του καταλόγου γίνεται σε δύο φάσεις. Στην πρώτη φάση βρίσκει κάθε όρο που περιλαμβάνεται μέσα στα έγγραφα και αποθηκεύεται στη συλλογή indexer_db η συχνότητα εμφάνισης (t_freq), η λίστα των εγγράφων (documents). Στη δεύτερη φάση υπολογίζεται το μέτρο του κάθε εγγράφου με τη χρήση των τύπων:

$$normalizedtf = \frac{f_{t,d}}{max_x f_{x,d}}$$

$$\text{normalized idf} = \frac{\ln\left(\frac{N}{n_t}\right)}{ln(N)}$$

$$w_{x,d} = tf_{x,d} idf_x$$

$$L_d = \sqrt{\sum_{x \in T_d} w_{x,d}^2}$$

Αν προσπαθήσουμε να τρέξουμε το server χωρίς να κάνουμε crawl κανένα document τότε φορτώνεται μια σελίδα λάθους που μας ενημερώνει ότι ο κατάλογος είναι κενός και πρέπει πρώτα να μαζέψουμε κάποια έγγραφα και να χτίσουμε τον κατάλογο και μετά να κάνουμε refresh

Oh no! It seems that the Inverted Index is empty and there are no documents to search.

This could mean that the index has not been initialized yet so you need to crawl some pages and create index.
If you are executing the create index for the first time and it is not initialized yet, please wait and refresh.

Please refresh your page and try again later!

Το υποσύστημα Query Handler δέχεται ένα όρισμα που αφορά των αριθμό των threads που θα τρέχουν παράλληλα για την επεξεργασία του ερωτήματος του χρήστη. Όταν ο

χρήστης υποβάλλει ένα ερώτημα και κάνει submit με το κουμπί search, καλείται η μέθοδος main() του Query Handler που δέχεται τις λέξεις κλειδιά που που πληκτρολόγησε ο χρήστης καθώς και τον αριθμό των top-k αποτελεσμάτων που θέλει να εμφανιστούν.
Όπως αναφέρεται και στις απαιτήσεις, για τον υπολογισμό της ομοιότητας των εγγράφων με το ερώτημα του χρήστη γίνεται χρήση του τύπου cosine similarity, ο οποίος είναι ο εξής:

$$S(q,d) = \frac{1}{\mathcal{L}_q \mathcal{L}_d} \sum_{t \in T_{q,d}} \left(1 + ln\left(f_{t,d}\right)\right) ln\left(1 + \frac{N}{n_t}\right)$$

q: διάνυσμα ερωτήματος του χρήστη στη μηχανή αναζήτησης
d: έγγραφο της συλλογής
N: το πλήθος των εγγράφων της συλλογής
$T_{q,d}$: οι κοινοί όροι μεταξύ ερωτήματος και εγγράφου
$f_{t,d}$: η συχνότητα εμφάνισης του όρου t στο έγγραφο d
n_t: ο αριθμός των εγγράφων που περιέχουν τον όρο t
$\mathcal{L}_q$: το μέτρο του διανύσματος του ερωτήματος όπου εδώ το θεωρούμε ότι ισούται με τη μονάδα αφού είναι το ίδιο για όλα τα documents.
$\mathcal{L}_d$: το μέτρο του διανύσματος ενός εγγράφου

Ο χρήστης έχει επίσης τη δυνατότητα να συμπληρώσει στο input text field τα document με τη σειρά εμφάνισης τους στα αποτελέσματα που εκείνος θεωρεί ως σχετικά (όσα δεν επιλεχθούν θεωρούνται άσχετα) και με τη χρήση του τύπου του Rocchio να βρεθεί ένα βέλτιστο ερώτημα (query) που θα επιστρέψει πιο σχετικά documents. Ο τύπος του Rocchio είναι ο εξής:

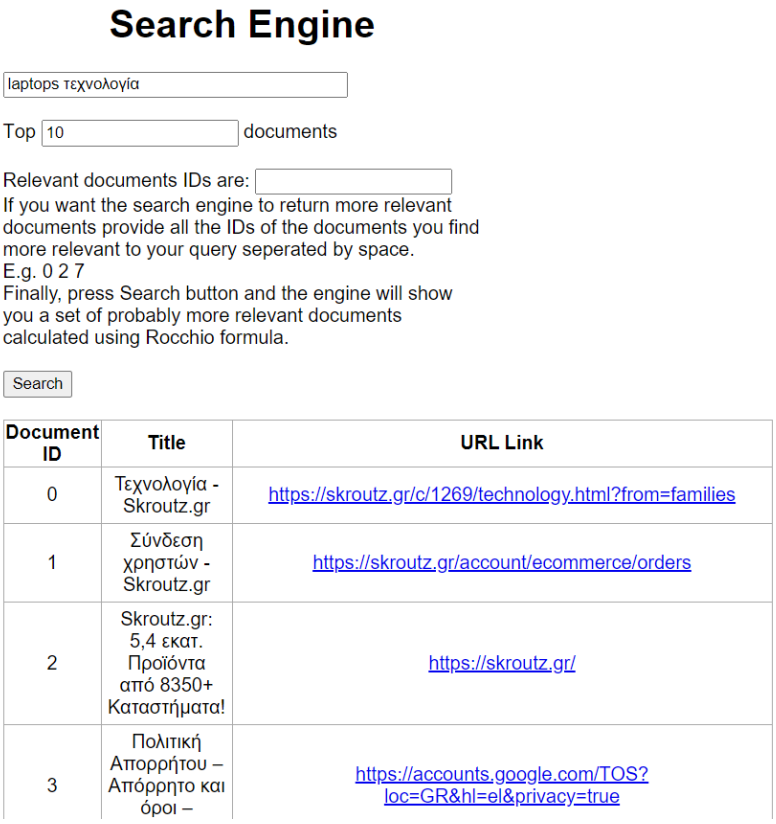
$$\mathbf{q}_m = \alpha \mathbf{q}_0 + \beta \frac{1}{|R|} \sum_{\mathbf{d}_j \in R} \mathbf{d}_j - \gamma \frac{1}{N - |R|} \sum_{\mathbf{d}_j} \mathbf{d}_j$$

$\mathbf{q}_m$: μετασχηματισμένο ερώτημα που θα είναι πιο κοντά στα σχετικά έγγραφα και πιο μακριά από τα μη σχετικά
$\mathbf{q}_0$: αρχικό ερώτημα που υπέβαλλε ο χρήστης
α, β, γ; Τα βάρη θέτουμε α=0.5, β=0.7 και γ=0.1
R: σύνολο των σχετικών εγγράφων
NR: σύνολο μη σχετικών εγγράφων
Αφού επιλεχθούν κάποια έγγραφα του αποτελέσματος και γίνει ξανά αναζήτηση πατώντας το κουμπί Search, η μηχανή θα επιστρέψει τα έγγραφα με τη μεγαλύτερη σχετικότητα υποβάλλοντας στο σύστημα το νέο μεταχηματισμένο ερώτημα που προκύπτει τρέχοντας τη συνάρτηση rocchio_relevance_feedback.

Για τον flask server χρησιμοποιήθηκε κώδικας από το tutorial του freeCodeCamp που δίνεται στις πηγές. Μπορούμε να τρέξουμε το server δίνοντας ως όρισμα τον αριθμό των threads όπως φαίνεται στην εικόνα παρακάτω.

```
PS C:\Users\VIVI PAPAGEORGIOU\Downloads\search-engine-assignment\search-engine-assignment> & "C:/Users/VIVI PAPAGEORGIOU/AppData/Local/Microsoft/WindowsApps/python.exe" "c:/Users/VIVI PAPAGEORGIOU/Downloads/search-engine-assignment/search-engine-assignment/app.py" 2
Starting Flask Server...
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
Starting Flask Server...
 * Debugger is active!
 * Debugger PIN: 316-216-615
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Στη συνέχεια παραθέτουμε κάποιες εικόνες που δείχνουν τη λειτουργία της μηχανής αναζήτησης σε κάποια ερωτήματα. Στην πρώτη εικόνα υποβάλλουμε το ερώτημα με keywords laptops τεχνολογία. Στη δεύτερη εικόνα δίνουμε σχετικά έγγραφα τα 0 και 4 και βλέπουμε τα νέα έγγραφα που μας επιστρέφει η μηχανή αναζήτησης για το νέο βελτιστοποιημένο ερώτημα. Στη τρίτη εικόνα θέτουμε το ερώτημα gmail και επιστρέφονται έγγραφα σχετικά με τη google.



Search Engine

laptops τεχνολογία

Top 10 documents

Relevant documents IDs are:

If you want the search engine to return more relevant documents provide all the IDs of the documents you find more relevant to your query seperated by space.
E.g. 0 2 7
Finally, press Search button and the engine will show you a set of probably more relevant documents calculated using Rocchio formula.

Search

Document ID	Title	URL Link
0	Τεχνολογία - Skroutz.gr	https://skroutz.gr/c/1269/technology.html?from=families
1	Σύνδεση χρηστών - Skroutz.gr	https://skroutz.gr/account/ecommerce/orders
2	Skroutz.gr: 5,4 εκατ. Προϊόντα από 8350+ Καταστήματα!	https://skroutz.gr/
3	Πολιτική Απορρήτου – Απόρρητο και όροι –	https://accounts.google.com/TOS?loc=GR&hl=el&privacy=true

Search Engine

laptops τεχνολογία

Top 10 documents

Relevant documents IDs are: 0 4

If you want the search engine to return more relevant documents provide all the IDs of the documents you find more relevant to your query seperated by space.
E.g. 0 2 7
Finally, press Search button and the engine will show you a set of probably more relevant documents calculated using Rocchio formula.

Search

Document ID	Title	URL Link
0	Σύνδεση χρηστών - Skroutz.gr	https://skroutz.gr/account/ecommerce/orders
1	Τεχνολογία - Skroutz.gr	https://skroutz.gr/c/1269/technology.html?from=families
2	Skroutz.gr: 5,4 εκατ. Προϊόντα από 8350+ Καταστήματα!	https://skroutz.gr/
3	Laptops - Skroutz.gr	https://skroutz.gr/c/25/laptop.html?from=families
4	Μόδα - Skroutz.gr	https://skroutz.gr/c/274/fashion.html?from=families

Search Engine

gmail

Top 10 documents

Relevant documents IDs are:

If you want the search engine to return more relevant documents provide all the IDs of the documents you find more relevant to your query seperated by space.
E.g. 0 2 7
Finally, press Search button and the engine will show you a set of probably more relevant documents calculated using Rocchio formula.

Search

Document ID	Title	URL Link
0	Πολιτική Απορρήτου – Απόρρητο και όροι – Google	https://accounts.google.com/TOS?loc=GR&hl=el&privacy=true
1	Όροι Παροχής Υπηρεσιών της Google – Απόρρητο και όροι – Google	https://accounts.google.com/TOS?loc=GR&hl=el
2	Gmail	https://accounts.google.com/AccountChooser?continue=https%3A%2F%2Fmail.google.com%2Fmail%2F%3Ftab%3Dwm&service=mail&rm=false&ltmpl=default&scc=1&osid=1&emr=1

ΠΗΓΕΣ

- https://www.youtube.com/watch?v=Z1RJmh_OqeA&ab_channel=freeCodeCamp.org
- https://github.com/jakerieger/FlaskIntroduction
- https://www.digitalocean.com/community/tutorials/how-to-set-up-flask-with-mongodb-and-docker
- https://www.python.org/
- https://www.w3schools.com/python/python_regex.asp