

# Ανάκτηση Πληροφορίας 2019-2020

## Εργασία: Σχεδιασμός και Υλοποίηση Μηχανής Αναζήτησης

Μια εργασία των:

Στέφανος Καραμπέρας – ΑΕΜ 2910

Γιάννης Μαυρουδόπουλος – ΑΕΜ 2770

### Εισαγωγή

Σκοπός της παρούσας εργασίας είναι η υλοποίηση μιας απλής μηχανής αναζήτησης, η οποία όμως θα έχει όλη τη βασική λειτουργικότητα μιας μηχανής αναζήτησης μεγάλης κλίμακας.

Η μηχανή αναζήτησης αποτελείται από τα εξής διακριτά υποσυστήματα:

- Σαρωτής (**Crawler**): Είναι το υποσύστημα της μηχανής αναζήτησης που είναι υπεύθυνο για τη σάρωση του παγκοσμίου ιστού και την εύρεση των ιστοσελίδων που θα αποτελέσουν το σύνολο των εγγράφων (documents) της μηχανής.
- Δεικτοδοτητής (**Indexer**): Είναι το υποσύστημα της μηχανής αναζήτησης που λαμβάνει το σύνολο εγγράφων (documents) από το σαρωτή και στη συνέχεια οργανώνει το περιεχόμενο με τέτοιο τρόπο που να επιτρέπει τη γρήγορη εκτέλεση αναζητήσεων σε αυτό.
- Επεξεργαστής Ερωτημάτων (**Query Processor**): Είναι το υποσύστημα της μηχανής αναζήτησης το οποίο λαμβάνει από τον χρήστη ένα ερώτημα που αποτελείται από λέξεις-κλειδιά και σε συνεργασία με τον δεικτοδοτητή (indexer), επιστρέφει στον χρήστη τα έγγραφα (ιστοσελίδες) που θεωρούνται σχετικά ως προς το υποβληθέν ερώτημα.

Τέλος, όπως ορίζουν οι προδιαγραφές της εκφώνησης για την παρούσα εργασία, όλα τα υποσυστήματα της μηχανής αναζήτησης εκμεταλλεύονται τον παραλληλισμό και επομένως είναι σε θέση να χρησιμοποιούν πολλαπλά νήματα (threads) για την εκτέλεση των ενεργειών τους.

## Τεχνολογίες που αξιοποιήθηκαν

Στο πλαίσιο υλοποίησης της μηχανής αναζήτησης, αξιοποιήθηκαν οι ακόλουθες τεχνολογίες προγραμματισμού και ανάπτυξης λογισμικών:

➤ **Python**

Η «Python» (έκδοση **3.6**) χρησιμοποιήθηκε ως γλώσσα προγραμματισμού για την υλοποίηση και των 3 υποσυστημάτων της μηχανής αναζήτησης, λόγω της δυνατότητας ταχείας συγγραφής ευανάγνωστου κώδικα που αυτή προσφέρει.

➤ **Flask**

Η «Flask» χρησιμοποιήθηκε ως framework για την κατασκευή της διεπαφής χρήστη με χρήση βασικής «**HTML/CSS**». Μέσω της διεπαφής, ο χρήστης υποβάλλει τα ερωτήματα του (**queries**) στη μηχανή αναζήτησης.

➤ **MongoDB**

Η «MongoDB» χρησιμοποιήθηκε για την κατασκευή της μη-σχεσιακής βάσης δεδομένων (**NoSQL Database**) στην οποία αποθηκεύονται οι πληροφορίες για τις ιστοσελίδες που συλλέγει ο «**Crawler**», ο κατάλογος που δημιουργεί ο «**Indexer**» αλλά και συλλογές δεδομένων απαραίτητες για την ανεξάρτητη λειτουργία (σε σχέση με την κατάσταση εκτέλεσης των υπόλοιπων υποσυστημάτων) του «**Query Processor**».

➤ **Docker**

Το εργαλείο «**Docker**» χρησιμοποιήθηκε για την εύκολη και γρήγορη αρχικοποίηση της βάσης δεδομένων «**MongoDB**» που αξιοποιεί το σύστημα μηχανής αναζήτησης, ανεξάρτητα από την τοπική παραμετροποίηση του λειτουργικού συστήματος στο οποίο εκτελείται η εφαρμογή.

## Συνοπτική περιγραφή των αρχείων του project

Ακολουθεί συνοπτική παρουσίαση των αρχείων του project και της συνεισφοράς τους στη λειτουργία του.

### ➤ Αρχεία κώδικα:

#### ◆ **app.py:**

Το αρχείο περιέχει το μέρος του κώδικα που αφορά τον «**Flask Server**», μέσω του οποίου είναι προσβάσιμη η HTML σελίδα διεπαφής της μηχανής αναζήτησης.

#### ◆ **mongodb.py:**

Το αρχείο περιέχει όλες τις μεθόδους που χρησιμοποιούνται από τα 3 υποσυστήματα της μηχανής αναζήτησης για την επικοινωνία τους με τη βάση δεδομένων της «**MongoDB**».

#### ◆ **crawler.py:**

Το αρχείο περιέχει το μέρος του κώδικα που αφορά το υποσύστημα του «**Crawler**».

#### ◆ **indexer.py:**

Το αρχείο περιέχει το μέρος του κώδικα που αφορά το υποσύστημα του «**Indexer**».

#### ◆ **query\_handler.py:**

Το αρχείο περιέχει το μέρος του κώδικα που αφορά το υποσύστημα του «**Query Processor**».

### ➤ Αρχεία HTML/CSS:

#### ◆ **./templates/base.html:**

Περιέχει τη βασική δομή της σελίδας διεπαφής χρήστη που αξιοποιεί ο «**Flask Server**».

#### ◆ **./templates/index.html:**

Επεκτείνει τη βασική δομή του «**base.html**» με τη δομή διεπαφής που εμφανίζει ο «**Flask Server**» στο χρήστη για την υποβολή ερωτημάτων (**queries**) και την εμφάνιση των αποτελεσμάτων αυτών στον τελευταίο.

- ◆ **./templates/loading.html:**

Επεκτείνει τη βασική δομή του «**base.html**» με τη δομή διεπαφής που εμφανίζει ο «**Flask Server**» στο χρήστη στο σενάριο κατά το οποίο η δόμηση του καταλόγου (**index**) βρίσκεται ακόμα υπό εξέλιξη. Το παραπάνω σενάριο αναλύεται περαιτέρω στη συνέχεια της παρούσας αναφοράς.

- ◆ **./static/css/main.css:**

Είναι το αρχείο «**css**» που αξιοποιείται από το «**base.html**» (και κατ' επέκταση από τα «**.html**» αρχεία που το επεκτείνουν) για τη μορφοποίηση του περιεχομένου του.

- Άλλα αρχεία:

- ◆ **requirements.txt:**

Περιέχει τις ονομασίες αλλά και τις εκδόσεις των απαιτούμενων βιβλιοθηκών που πρέπει να είναι εγκατεστημένες στο σύστημα για την επιτυχή εκτέλεση της μηχανής αναζήτησης. Το αρχείο μπορεί επίσης να αξιοποιηθεί με χρήση κατάλληλων εργαλείων για την αυτόματη λήψη των απαραίτητων βιβλιοθηκών.

- ◆ **docker-compose.yml:**

Πρόκειται για το αρχείο παραμετροποίησης που αξιοποιεί το «**Docker**» ώστε να αρχικοποιήσει τη βάση δεδομένων «**MongoDB**» που αξιοποιεί το σύστημα μηχανής αναζήτησης.

- ◆ **.env:**

Περιέχει τα στοιχεία σύνδεσης της βάσης δεδομένων «**MongoDB**», δηλαδή το όνομα της βάσης δεδομένων, το όνομα και τον κωδικό root καθώς και τη διεύθυνση σύνδεσης της βάσης δεδομένων.

- ◆ **README.md:**

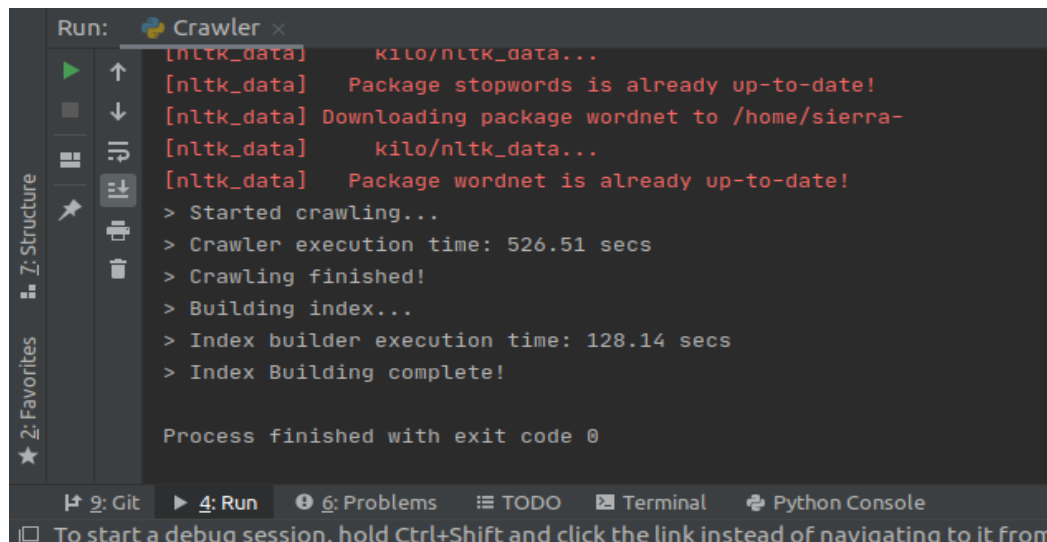
Περιέχει συνοπτικές οδηγίες εκτέλεσης του προγράμματος.

## Παράδειγμα εκτέλεσης της μηχανής αναζήτησης

**Σημαντικό:** Πριν την εκτέλεση του συστήματος μηχανής αναζήτησης, πρέπει να έχει εγκατασταθεί στο σύστημα το εργαλείο «**Docker**» καθώς και το εργαλείο «**docker-compose**» με χρήση του οποίου γίνεται η παραμετροποίηση και εκτέλεση της εφαρμογής. Επιπρόσθετα, πρέπει να γίνει εγκατάσταση των απαραίτητων βιβλιοθηκών που αναγράφονται στο αρχείο «**requirements.txt**» με χρήση της εντολής «**pip3 install -r requirements.txt**». Τέλος, με την ολοκλήρωση των παραπάνω, πρέπει να γίνει εκτέλεση της εντολής «**sudo docker-compose up**» ώστε να γίνει εκκίνηση του «**docker container**» στο οποίο παραμετροποιείται αυτόματα η εφαρμογή. Στη συνέχεια, μπορούμε να προχωρήσουμε σε χρήση της εφαρμογής, όπως παρουσιάζεται παρακάτω.

Ακολουθεί σύντομο παράδειγμα ενδεικτικής εκτέλεσης του συστήματος της μηχανής αναζήτησης:

- 1) Εκτελούμε το υποσύστημα του «**Crawler**» προκειμένου να σχηματίσουμε τη συλλογή εγγράφων ιστοσελίδων. Για τον σκοπό αυτό εκτελούμε το αρχείο «**crawler.py**», δίνοντας τα απαιτούμενα ορίσματα. Για παράδειγμα, δίνοντας ως παραμέτρους τα «**http://mypage.gr 200 1 8**», ξεκινάμε την προσπέλαση ιστοσελίδων με σημείο εκκίνησης την ιστοσελίδα «**http://mypage.gr**», για βάθος (πλήθος) 200 ιστοσελίδων, ζητώντας από τον «**Crawler**» να μην διαγράψει τυχόν υπάρχουσες εγγραφές της συλλογής εγγράφων ιστοσελίδων (προσθέτοντας δηλαδή απλά τις εγγραφές για τις καινούργιες ιστοσελίδες που συγκεντρώνει κατά τη σάρωση). Τέλος, το όρισμα «**8**» δηλώνει ότι επιθυμούμε την αξιοποίηση 8 νημάτων (**threads**) από τον crawler για την εκτέλεση των ενεργειών του.
- 2) Με την ολοκλήρωση της σάρωσης (**crawling**), το υποσύστημα «**Crawler**» αρχικοποιεί αντικείμενο (instance) του υποσυστήματος Indexer και καλεί τη μέθοδο δημιουργίας καταλόγου του. Το υποσύστημα «**Indexer**» λαμβάνει ως όρισμα το πλήθος νημάτων (**threads**) που δύναται να αξιοποιήσει κατά την εκτέλεση των ενεργειών του. Στην περίπτωση μας, εφόσον γίνεται κλήση του «**Indexer**» από τον «**Crawler**», το όρισμα αυτό έχει την τιμή του αντίστοιχου ορίσματος που δώσαμε στον «**Crawler**» για την εκτέλεσή του, δηλαδή 8.

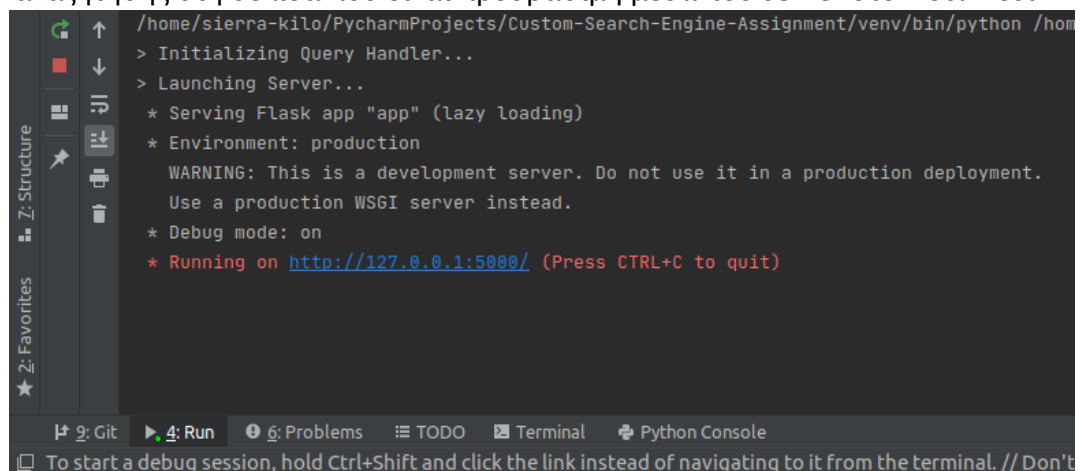


```
Run: Crawler x
[nltk_data] kilo/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/sierra-
[nltk_data] kilo/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
> Started crawling...
> Crawler execution time: 526.51 secs
> Crawling finished!
> Building index...
> Index builder execution time: 128.14 secs
> Index Building complete!

Process finished with exit code 0
```

Εικόνα 1: Μηνύματα εξόδου στην κονσόλα μετά το πέρας της εκτέλεσης του «**Crawler**» και του «**Indexer**».

- 3) Στη συνέχεια, προχωράμε στην εκκίνηση του «**Flask Server**» για την δημιουργία της HTML διεπαφής υποβολής ερωτημάτων χρήστη. Για τον σκοπό αυτό, εκτελούμε το αρχείο «**app.py**» δίνοντας ως όρισμα το πλήθος νημάτων (threads), ενώ στη συνέχεια επισκεπτόμαστε τη σελίδα υποβολής ερωτημάτων (**queries**) της μηχανής αναζήτησης στη σελίδα που είναι προσβάσιμη μέσω του server στον localhost.



```
/home/sierra-kilo/PycharmProjects/Custom-Search-Engine-Assignment/venv/bin/python /hom
> Initializing Query Handler...
> Launching 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
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Εικόνα 2: Μηνύματα εξόδου στην κονσόλα μετά το πέρας της εκκίνησης του **Flask Server**.

## Better than Google

Show top  results

Title	Url
Wikimedia Projects – Wikimedia Foundation	<a href="https://wikimediafoundation.org/our-work/wikimedia-projects/">https://wikimediafoundation.org/our-work/wikimedia-projects/</a>
Wikipedia, the free encyclopedia	<a href="https://en.wikipedia.org/">https://en.wikipedia.org/</a>
Wikimedia Commons	<a href="https://commons.wikimedia.org/">https://commons.wikimedia.org/</a>
Wikimedia Commons	<a href="https://commons.wikimedia.org/wiki/">https://commons.wikimedia.org/wiki/</a>

Εικόνα 3: Τα αποτελέσματα της αναζήτησης για τη λέξη-κλειδί “free”.

## Αναλυτική Περιγραφή Λειτουργίας Υποσυστημάτων

### Crawler

Ο «**Crawler**» δέχεται κατά την αρχικοποίησή του ως ορίσματα (κατά την αναφερόμενη σειρά):

1. Μία συμβολοσειρά που αντιπροσωπεύει τη διεύθυνση της ιστοσελίδας εκκίνησης (π.χ. <http://mypage.gr/>).
2. Έναν ακέραιο αριθμό που αντιπροσωπεύει το βάθος σάρωσης, δηλαδή το πλήθος των ιστοσελίδων που θα επισκεφτεί, ξεκινώντας από την ιστοσελίδα εκκίνησης.
3. Έναν ακέραιο αριθμό με τιμή 0 ή 1, ο οποίος καθορίζει αν τα δεδομένα που τυχόν υπάρχουν από προηγούμενες εκτελέσεις του «**Crawler**» θα διατηρηθούν με την προσθήκη των δεδομένων που θα προκύψουν από την τρέχουσα σάρωση (τιμή 1), ή θα διαγραφούν (τιμή 0) κατά την τρέχουσα εκτέλεση του «**Crawler**».
4. Έναν ακέραιο αριθμό μεγαλύτερο του 0, ο οποίος καθορίζει το πλήθος νημάτων επεξεργασίας (threads) που θέλουμε να αξιοποιήσει ο «**Crawler**» για την εκτέλεσή του.

Το πρόγραμμα ξεκινάει από την αρχική σελίδα, δημιουργεί ένα καινούριο νήμα και εκτελεί την συνάρτηση «**parse\_url**». Στην συνέχεια για κάθε νέα σελίδα που έχει εντοπίσει ξεκινάει ένα νέο νήμα. Υπάρχει μια επανάληψη η οποία ελέγχει πόσα ενεργά νήματα υπάρχουν και αν ο αριθμός είναι ίσος με τον αριθμό που έχει δώσει ο χρήστης ως όρισμα (πλήθος νημάτων επεξεργασίας), γίνεται αναστολή του προγράμματος για μισό δευτερόλεπτο και στη συνέχεια πραγματοποιείται εκ νέου έλεγχος των ενεργών νημάτων. Η λίστα με της επόμενες σελίδες που θα εξετάσει το πρόγραμμα («**head**») είναι «**thread-safe**», δηλαδή ένα νήμα για να κάνει κάποια τροποποίηση χρειάζεται να κλειδώσει (lock) τη δομή, να κάνει τις αλλαγές και στη συνέχεια να την ξεκλειδώσει. Κάθε επιτυχημένη σάρωση (parsing) μιας σελίδας, αυξάνει κατά 1 έναν μετρητή και το πρόγραμμα τερματίζει όταν αυτός ο μετρητής γίνει ίσος με το όρισμα που δόθηκε από τον χρήστη (**βάθος σάρωσης**).

### Συνάρτηση «**parse\_url**»:

Είναι η συνάρτηση που εκτελείται για την προεπεξεργασία κάθε νέας σελίδας. Αρχικά το κείμενο της σελίδας διαβάζεται με την βοήθεια της βιβλιοθήκης «**BeautifulSoup**» (html parser), ενώ μέσα από την «html» κρατάμε και τον τίτλο της σελίδας. Κάνουμε χρήση του τίτλου σελίδας και της διεύθυνσης σελίδας ώστε να ελέγξουμε αν η



εξεταζόμενη σελίδα υπάρχει ήδη μέσα στη βάση δεδομένων (συλλογή «**crawler**» στη «**MongoDB**»). Στη συνέχεια, η «**parse\_url**» κρατάει όλες τις αναφορές που γίνονται σε άλλες σελίδες, κλειδώνει τη λίστα «**head**», προσθέτει τις αναφορές και τέλος απελευθερώνει τη δομή. Το κείμενο της σελίδας περνάει από «**tokenizer**» (γίνεται με την βοήθεια της βιβλιοθήκης **nlk**) και ακολούθως, με τη χρήση κανονικών εκφράσεων, απορρίπτονται «**tokens**» που ξεκινάνε με/ή περιέχουν «περίεργους» χαρακτήρες. Έπειτα, αφαιρούμε όποιες λέξεις ανήκουν στα «**stop words**» (τα οποία έχουμε βρει με την βιβλιοθήκη «**nlk**») και τέλος, μετατρέπουμε τις λέξεις σε μια βασική μορφή, ώστε να μην διαφοροποιείται μια λέξη αν είναι σε διαφορετικό χρόνο ή διαφορετική κλήση (διαδικασία «**lemmatization**» - βιβλιοθήκη «**WordNetLemmatizer**»).

Σε αυτό το σημείο, αξίζει να αιτιολογηθεί η προτίμηση της διαδικασίας «**lemmatization**» έναντι του «**stemming**». Η κύρια διαφορά μεταξύ των 2 τεχνικών είναι ότι το **stemming** δεν λαμβάνει υπόψιν το νόημα μίας λέξης στο ενιαίο κείμενο που την περιλαμβάνει (**context**), επιτυγχάνοντας λόγω αυτού χαμηλότερη ακρίβεια σε σχέση με το «**lemmatization**». Με δεδομένο το παραπάνω, θεωρήθηκε ωφέλιμο για την επίτευξη καλύτερων αποτελεσμάτων κατά την ανάκτηση πληροφοριών από το σύστημά μας, να προχωρήσουμε σε αξιοποίηση του «**lemmatization**» κατά την προεπεξεργασία των λέξεων που συλλέγει ο «**Crawler**».

Όταν ολοκληρωθεί η διαδικασία σάρωσης μίας σελίδας, δημιουργείται η συλλογή λέξεων της («**bag**»), η οποία για κάθε λέξη της περιέχει ένα στοιχείο «**JSON**» της μορφής: {λέξη : αριθμός εμφανίσεων στην σελίδα}.

Στη συνέχεια, τα δεδομένα της ιστοσελίδας που σαρώθηκε αποθηκεύονται στη βάση δεδομένων (συλλογή «**crawler**») με τη χρήση στοιχείου «**JSON**» που περιλαμβάνει τα ακόλουθα χαρακτηριστικά:

- **url**: η ηλεκτρονική διεύθυνση της ιστοσελίδας.
- **title**: ο τίτλος της ιστοσελίδας.
- **bag** : η «συλλογή» λέξεων που δημιουργήθηκε για την ιστοσελίδα.

Τέλος, αυξάνεται ο μετρητής των σελίδων που έχουν αποθηκευτεί.

Με το πέρας της εκτέλεσής του και έχοντας ολοκληρώσει τη συμπλήρωση της συλλογής «**crawler**» στη Βάση Δεδομένων με τις πληροφορίες των ιστοσελίδων που έχει διασχίσει, ο Crawler καλεί τη μέθοδο κατασκευής καταλόγου του υποσυστήματος «**Indexer**», προκειμένου να ξεκινήσει η διαδικασία σχηματισμού του αντεστραμμένου καταλόγου (**inverse index**).

## Indexer

Ο «**Indexer**» κατά την αρχικοποίησή του δέχεται ως όρισμα 1 ακέραιο αριθμό με τιμή μεγαλύτερη του 0, ο οποίος καθορίζει το πλήθος νημάτων επεξεργασίας (**threads**) που θέλουμε να αξιοποιήσει κατά την εκτέλεσή του. Σε περίπτωση που ο «**Indexer**» αρχικοποιηθεί από τον «**Crawler**» (με το πέρας της σάρωσης των ιστοσελίδων από τον τελευταίο) λαμβάνει ως όρισμα για το πλήθος νημάτων (**threads**) τον αριθμό που δόθηκε ως όρισμα από τον χρήστη στον «**Crawler**» για τη δική του εκτέλεση.

Ειδική περίπτωση λειτουργίας του «**Indexer**» αποτελεί το σενάριο κατά το οποίο όσο εκτελείται η μέθοδος κατασκευής καταλόγου του Indexer, ο χρήστης εκτελεί εκ νέου τον «**Crawler**». Το παραπάνω θα μπορούσε να οδηγήσει σε καταστροφική αστοχία του «**Indexer**», μιας και η συλλογή πληροφοριών «**crawler**» της βάσης δεδομένων που θα αξιοποιούσε ως πηγή εγγράφων για τη δημιουργία του index, πιθανώς να διαγραφεί κατά την νέα εκτέλεση του «**Crawler**».

Για την αντιμετώπιση αυτής της πιθανότητας, το πρώτο βήμα πριν ξεκινήσει η κατασκευή του καταλόγου index, είναι η αντιγραφή των εγγραφών της συλλογής «**crawler**» σε μία νέα συλλογή της Βάσης Δεδομένων, που ονομάζεται «**documents**». Με την αποθήκευση του πληροφοριακού στιγμιότυπου στη νέα συλλογή «**documents**», στην οποία έχει πρόσβαση μόνο το υποσύστημα «**Indexer**», διασφαλίζουμε την ορθή λειτουργία του τελευταίου, ακόμα και αν κατά τη διαδικασία δημιουργίας καταλόγου γίνει παράλληλα εκ νέου εκτέλεση του υποσυστήματος «**Crawler**».

Ο «**Indexer**» αξιοποιεί τη δομή αντεστραμμένου καταλόγου. Για τον σχηματισμό της, από κάθε έγγραφο της συλλογής «**documents**», απομονώνεται το χαρακτηριστικό «**bag**». Πρόκειται για μία λίστα που περιέχει το σύνολο των λέξεων που έχουν συλλεχθεί από την ιστοσελίδα του τρέχοντος εγγράφου. Στη συνέχεια, για κάθε λέξη της λίστας «**bag**», εξετάζεται αν αυτή υπάρχει ήδη στον κατάλογο (συλλογή «**index**» της νάσης δεδομένων):

- Σε περίπτωση που η λέξη υπάρχει ήδη στον κατάλογο, ανανεώνεται η καταχώρησή της, με αύξηση κατά 1 του μετρητή συχνότητας εμφάνισης της (**word frequency**) και προσθήκη στη λίστα των ιστοσελίδων εμφάνισης της των πληροφοριών της ιστοσελίδας στην οποία εντοπίστηκε, με τη χρήση στοιχείου «**JSON**» που περιλαμβάνει τα ακόλουθα χαρακτηριστικά:
  - **\_id**: Το αναγνωριστικό του εγγράφου της ιστοσελίδας στη συλλογή «**documents**» της βάσης δεδομένων.
  - **title**: Ο τίτλος της ιστοσελίδας.
  - **url**: Η ηλεκτρονική διεύθυνση της ιστοσελίδας.

- **w\_d\_freq**: Η συχνότητα εμφάνισης της λέξης στο έγγραφο της ιστοσελίδας.
- Σε περίπτωση που η λέξη δεν υπάρχει ακόμα στον κατάλογο, τότε δημιουργείται μία νέα εγγραφή, στην λίστα των ιστοσελίδων εμφάνισης της οποίας προστίθεται ομοίως με πριν στοιχείο «**JSON**» που περιλαμβάνει τα χαρακτηριστικά της ιστοσελίδας στην οποία εντοπίστηκε η λέξη. Η διαφορά σε σχέση με την περίπτωση ενημέρωσης εγγραφής του καταλόγου, είναι ότι σε αυτή την περίπτωση, ο μετρητής συχνότητας εμφανίσεων της λέξης (**w\_freq**) αρχικοποιείται με την τιμή 1, αντί να αυξηθεί κατά 1.

Ο σχηματισμός/ενημέρωση του καταλόγου γίνεται με την τεχνική της αναδόμησης. Αυτό σημαίνει ότι κάθε φορά που καλείται η μέθοδος δημιουργίας καταλόγου του υποσυστήματος «**Indexer**», ο παλιός κατάλογος (συλλογή «**index**» στη βάση δεδομένων) διαγράφεται, ενώ τα αποθηκευμένα έγγραφα (εγγραφές ιστοσελίδων) στη συλλογή «**documents**» της βάσης δεδομένων αξιοποιούνται για τη δημιουργία ενός νέου αντεστραμμένου καταλόγου.

Με το πέρας της δόμησης του καταλόγου (index), γίνεται υπολογισμός των μηκών εγγράφων (**document length**). Τα μήκη εγγράφων είναι απαραίτητα λόγω των αναγκών της φόρμουλας βαθμολόγησης εγγράφων (**document score**) που χρησιμοποιείται για από τον επεξεργαστή ερωτημάτων (**Query Handler**) και παρουσιάζεται αναλυτικά στη συνέχεια της παρούσας αναφοράς. Το αριθμητικό αποτέλεσμα μήκους για κάθε διάνυσμα εγγράφου της συλλογής «**documents**» αποθηκεύεται εντός αυτής, ως νέο χαρακτηριστικό σε κάθε καταχώρηση εγγράφου (χαρακτηριστικό **length**).

Τόσο κατά την κατασκευή του αντεστραμμένου καταλόγου όσο και κατά τον υπολογισμό των μηκών εγγράφων αξιοποιούνται πολλαπλά threads, ανάλογα με το όρισμα πλήθους νημάτων επεξεργασίας που δόθηκε στον «**Indexer**».

Έχοντας ολοκληρώσει την κατασκευή του αντεστραμμένου καταλόγου αλλά και τον υπολογισμό μηκών εγγράφων, ο Indexer, ως τελευταίο βήμα, ενημερώνει τις συλλογές του «**Query Handler**» στη βάση δεδομένων (συλλογές «**query\_documents**» και «**query\_index**») σύμφωνα με το περιεχόμενο των δικών του συλλογών (συλλογές «**index**» και «**documents**»). Κατά αυτό τον τρόπο, δημιουργείται ένα «στιγμιότυπο» του πληροφοριακού περιεχομένου των συλλογών του «Indexer» στις συλλογές του «Query Handler», έτσι ώστε ακόμα και αν κατά την επεξεργασία ενός ερωτήματος του χρήστη προκύψει καταστροφική επαναδόμηση του καταλόγου (index) λόγω κάποιας εκ νέου εκτέλεσης του υποσυστήματος «**Indexer**», αυτό να μην επηρεάσει την ορθή λειτουργία του υποσυστήματος «**Query Handler**».

## Query Handler

Ο «**Query Handler**» κατά την αρχικοποίησή του δέχεται ως όρισμα 1 ακέραιο αριθμό με τιμή μεγαλύτερη του 0, ο οποίος καθορίζει το πλήθος νημάτων επεξεργασίας (**threads**) που θέλουμε να αξιοποιήσει κατά την εκτέλεσή του.

Κάθε φορά που ο χρήστης υποβάλλει κάποιο ερώτημα στη μηχανή αναζήτησης, καλείται η μέθοδος υπολογισμού ερωτήματος του «**Query Handler**». Η μέθοδος αυτή δέχεται ως ορίσματα τις λέξεις-κλειδιά του ερωτήματος που έχει υποβάλλει προς επεξεργασία ο χρήστης, καθώς και το πλήθος εγγράφων που πρέπει να επιστραφούν ως αποτέλεσμα.

Για τον υπολογισμό της ομοιότητας ενός εγγράφου σε σχέση με το υποβληθέν ερώτημα, αξιοποιείται μία συνάρτηση βαθμολόγησης βασισμένη στην ομοιότητα συνημίτονου (**cosine similarity score**). Συγκεκριμένα, αξιοποιείται η παρακάτω φόρμουλα:

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

όπου:

- $q$  είναι το υποβληθέν ερώτημα (*query*)
- $d$  είναι ένα έγγραφο της συλλογής
- $L_q$  είναι το μέτρο του διανύσματος του ερωτήματος  $q$
- $L_d$  είναι το μέτρο του διανύσματος του εγγράφου  $d$
- $N$  είναι το πλήθος των εγγράφων της συλλογής
- $n_t$  είναι το πλήθος των εγγράφων που περιέχουν τον όρο  $t$
- $T_{q,d}$  είναι το σύνολο των κοινών όρων του  $q$  και  $d$
- $f_{t,d}$  είναι η συχνότητα εμφάνισης του όρου  $t$  στο έγγραφο  $d$

Αξίζει να σημειωθεί ότι πρακτικά ο όρος  $L_q$  δε χρειάζεται να χρησιμοποιείται για τον υπολογισμό της ομοιότητας, καθώς έχει την ίδια επίδραση σε όλα τα έγγραφα.

Τέλος, θεωρείται δεδομένο ότι τα έγγραφα που επιστρέφονται από την μέθοδο υπολογισμού ερωτήματος επιστρέφονται ταξινομημένα από το περισσότερο προς το λιγότερο σχετικό ως προς το υποβληθέν ερώτημα (*query*).

### Ειδική περίπτωση εκτέλεσης του «Query Handler»

Κατά την αρχική εκτέλεση της εφαρμογής (πριν εκτελεστεί ο «**Crawler**» και ο «**Indexer**») είναι λογικό ότι οι συλλογές του «**Query Handler**» (συλλογές

«**query\_documents**» και «**query\_index**») δεν υπάρχουν ακόμα στη βάση δεδομένων. Σε αυτή την περίπτωση, αν ο χρήστης αποπειραθεί να εκκινήσει τον «**Flask Server**» και να ανοίξει μέσω αυτού τη σελίδα υποβολής ερωτημάτων, θα λάβει κατάλληλο μήνυμα προειδοποίησης. Το μήνυμα ενημερώνει τον χρήστη ότι οι συλλογές του «**Query Handler**» είναι κενές, πράγμα που υποδηλώνει είτε ότι ο «**Indexer**» δεν έχει εκκινηθεί ακόμα για πρώτη φορά, είτε ότι η δόμηση του καταλόγου για πρώτη φορά βρίσκεται σε εξέλιξη. Σε κάθε περίπτωση, το ιδιαίτερο αυτό σενάριο μπορεί να προκύψει μόνο κατά την πρώτη δόμηση του καταλόγου, μιας και αφού οι συλλογές του «**Query Handler**» (συλλογές «**query\_documents**» και «**query\_index**») ενημερωθούν για πρώτη φορά από τον «**Indexer**», στην επεξεργασία ερωτημάτων θα αξιοποιείται κάθε φορά το περιεχόμενο αυτών των ολοκληρωμένων (πλέον) συλλογών. Οι συλλογές του «**Query Handler**» ολοκληρώνονται στο τέλος κάθε δόμησης καταλόγου από τον «**Indexer**» με τα νέα δεδομένα που προκύπτουν, έτσι ώστε να περιέχουν σε κάθε περίπτωση την πιο πρόσφατη πλήρη μορφή του αντεστραμμένου καταλόγου.

### **Oops..! Query collections have not been initialized yet!**

This could mean that the index builder has not been executed yet!  
If that's the case, please execute the index builder first and try again!

It could also mean that the index builder is currently being executed for the first time!  
If that's the case, please wait while the index is being initialized!  
Good news, however! After the operation is complete, you will be able to execute queries **regardless of any future index rebuild operations being conducted concurrently.**

The required time for the initial index build depends on the size of the documents being processed.

Please refresh your page and try again later!

*Εικόνα 4: Το προειδοποιητικό μήνυμα που εμφανίζεται στον χρήστη, αν ο τελευταίος πραγματοποιήσει απόπειρα υποβολής ερωτήματος χωρίς να έχουν σχηματιστεί οι συλλογές του «**Query Handler**».*