Πανεπιστήμιο Δυτικής Αττικής

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Σωτήριος Σουκαράς, 21390206

Θεοφάνης Κουνιάκης, 21390103

Link του κώδικα:

A blue logo with text

Description automatically generated

Ανακτηση ΠΛηροφοριασ

ΤΕΛΙΚΗ ΈΚΘΕΣΗ ΑΝΑΦΟΡΑΣ

# Βήμα 1. Συλλογή δεδομένων:

Στο αρχείο **‘wiki\_scraper.py’** πραγματοποιείται συλλογή των δεδομένων από άρθρα του Wikipedia.

Η βασική συνάρτηση του αρχείου είναι η **‘fetch\_wikipedia’** η οποία δέχεται ως όρισμα τα **URL**, **depth\_limit** και **depth: (fetch\_wikipedia(URL, depth\_limit, depth = 1))**

* Η μεταβλητή **URL** αντιστοιχεί στον αρχικό σύνδεσμο του άρθρου που θα εξεταστεί.
* Το **depth\_limit** δηλώνει το μέγιστο βάθος της αναδρομής.
* Η μεταβλητή **depth** είναι το τρέχον επίπεδο της αναδρομής και έχει default τιμή **1**.

Η συνάρτηση αρχικά στέλνει **HTTP** αίτημα από το **URL** και ανακτά to **HTML**. Στην συνέχεια κρατά το κείμενο που περιέχεται στις παραγράφους του και αφαιρεί τα **html tags** τους.

Κάθε παράγραφος τοποθετείται στην λίστα **parsed\_paragraphs**.

Έπειτα γίνεται έλεγχος αν το **depth** είναι μεγαλύτερο ή ίσο του **depth\_limit**. Αν η συνθήκη ισχύει τότε η συνάρτηση επιστρέφει την λίστα **parsed\_paragraphs**.

Στην περίπτωση που η συνθήκη είναι ψευδής τότε γίνεται αναδρομή για κάθε link **(href)** καλώντας την **wiki\_scraper** με όρισμα **depth+1**. Η επιστροφή κάθε αναδρομής αποθηκεύεται στην λίστα **new\_paragraphs** η οποία χρησιμοποιείται για την ανανέωση της λίστας **parsed\_paragraphs**.

Με το τέλος των αναδρομών η συνάρτηση επιστρέφει την λίστα **parsed\_paragraphs**.

Για την συλλογή των δεδομένων η συνάρτηση κλήθηκε με το **URL** της σελίδας της Wikipedia του δευτέρου παγκοσμίου πολέμου με βάθος **depth = 2**. Η έξοδος της συνάρτησης τοποθετήθηκε στην λίστα **results**.

Τέλος η λίστα αυτή αποθηκεύεται στο αρχείο **wiki\_scrape.json** . Κάθε στοιχείο του αρχείου είναι ένα λεξικό που περιέχει:

* Το **URL** του άρθρου (με το κλειδί **website\_url**).
* Μια λίστα με παραγράφους του κειμένου του άρθρου (με το κλειδί **content**).

\* Στο Wikipedia, το **‘mw-content-text’** είναι το κύριο container (HTML div) που περιέχει το κείμενο του άρθρου. Επομένως η εντολή **‘soup\_responce.find(id="”w-content-text”)’** βρίσκει το τμήμα HTML id mw-content-text.

**Έξοδος:**

**A screenshot of a computer program

Description automatically generated**

wiki\_scrape.json

# Βήμα 2. Προεπεξεργασία κειμένου (Text Processing):

Στο αρχείο **‘preprocessor.py’** γίνεται η προεπεξεργασία του κειμένου που συλλέχθηκε από το **wiki\_scraper.**

Βασική συνάρτηση του αρχείου είναι η ‘**clear\_special\_char’** και αφαιρεί τα εξής:

* χαρακτήρες αναφοράς του τύπου ‘[1][2]’
* μη αλφαριθμητικούς χαρακτήρες
* τον χαρακτήρα ‘-’

Τα παραπάνω πραγματοποιούνται με την χρήση της συνάρτησης **re.sub** της βιβλιοθήκης **re**,η οποία κάνει αντικατάσταση επιλεγμένων χαρακτήρων σε μια συμβολοσειρά. Για τους χαρακτήρες αναφοράς και τους μη αλφαριθμητικούς χαρακτήρες χρησιμοποιήθηκε **regex**.

Η βασικότερη συνάρτηση είναι η **‘preprocess\_text’** και εκτελεί τα εξής:

* **Tokenization**: Χώρισμα κειμένου σε λέξεις.
* **Αφαίρεση stop-words** όπως ‘the’, ‘Is’, ‘or’
* **Lemmatization:** Μετατροπή λέξεων σε λεξικογραφική μορφή.

Το αρχείο ‘**wiki\_scrape.json’** ανοίγεται με όνομα **data**. Στην συνέχεια αρχικοποιείται το λεξικό **parsed\_data**. Για κάθε άρθρο του **data** γίνονται τα εξής:

* Ορίζεται μία καινή λίστα η **parsed\_content**.
* Αποθηκεύεται το **URL** του **data** στην μεταβλητή **link**.
* Για κάθε παράγραφο του άρθρου **paragraph** καλούνται οι συναρτήσεις **clear\_special\_char()** και **preprocess\_text().** Το τελικό αποτέλεσμα γίνεται append στην **parsed\_content.**
* Αποθηκεύεται το **parsed\_content** στο λεξικό **parsed\_data** με κλειδί το **link**.

Τέλος αποθηκεύονται τα στοιχεία του **parsed\_content** στο αρχείο **‘parsed\_scrape.json’**.

**Έξοδος:**

A screen shot of a computer

Description automatically generated

parsed\_scrape.json

# Βήμα 3. Ευρετήριο (Indexing):

Στο αρχείο **‘indexer.py’** δημιουργείται το αντεστραμμένο ευρετήριο με την συνάρτηση **crate\_inverted\_index()**. Η συνάρτηση δέχεται ως όρισμα μία λίστα από λεξικά την **data**. Στην συνάρτηση μετά από την αρχικοποίηση του λεξικού **inverted\_index**, για κάθε άρθρο του data γίνονται τα εξής:

* Αποθήκευση του **URL** στο **link**.
* Μετατροπή της παραγράφου **paragraph** σε πεζά γράμματα.
* Στην μεταβλητή **words** αποθηκεύεται ένα σύνολο (**set**) από μοναδικές λέξεις της παραγράφου, αγνοώντας πολλαπλές εμφανίσεις της ίδιας λέξης.
* Για κάθε λέξη στο σύνολο **words** γίνεται έλεγχος αν το **link** υπάρχει στην λίστα τιμών για την λέξη ‘**word’** στο inverted\_index. Εάν δεν υπάρχει γίνεται append στην λίστα.

Η συνάρτηση επιστρέφει το **inverted\_index**. Επομένως η έξοδος της συνάρτησης θα είναι ένα λεξικό με κλειδιά τις ξεχωριστές λέξεις και με τιμές τις λίστες από **URL** στα οποία αυτές εμφανίζονται.

Το αρχείο **‘parsed\_scrape.py’** δίνεται ως όρισμα στην συνάρτηση **create\_inverted\_index()** και εκχωρείται στο **inverted\_index** το οποίο στην συνέχεια γίνεται cast σε λεξικό. Το αποτέλεσμα αποθηκεύεται στο αρχείο **inverted\_index.json.**

**Έξοδος:**

A screenshot of a computer

Description automatically generated

inverted\_index.json

# Βήμα 4. Μηχανή αναζήτησης (Search Engine):

## α) Επεξεργασία ερωτήματος (Query Processing):

Στο αρχείο **‘search\_engine\_1.py’** γίνεται η επεξεργασία ερωτημάτων των χρηστών και εκτελούνται Boolean αναζητήσεις με λογικές πράξεις ώστε να βρεθούν τα σχετικά **URL**.

Η εξής λειτουργία πραγματοποιείται με την χρήση της συνάρτησης **request\_query**, η οποία δέχεται ως όρισμα την ερώτηση του χρήστη (**query**) και το αντεστραμμένο ευρετήριο (**index**).

Στην συνάρτηση γίνεται αρχικά δήλωση των λογικών τελεστών (**and,or,not**) στην μεταβλητή **logic\_operators**.

Στην συνέχεια διαχωρίζεται το ερώτημα σε λέξεις (**tokenization**), αφαιρούνται τα **stop words**, και μετατρέπεται το **query** σε λεξικογραφική μορφή (**lemmatization**).

Έπειτα αρχικοποιείται το **result** με ένα κενό **set**. Γίνεται επίσης αρχικοποίηση του λογικού τελεστή **op** με **or.**

Για κάθε στοιχείοτου **lemmatized\_query** γίνεται έλεγχος αν είναι λογικός τελεστής ή λέξη του αντεστραμμένου ευρετηρίου. Αν είναι τελεστής τότε εκχωρείται στην μεταβλητή **op**. Αν το στοιχείο **token** δεν περιέχεται στο αντεστραμμένο ευρετήριο η λούπα τερματίζει. Αν το στοιχείο υπάρχει τότε αποθηκεύονται τα **URL** του στην λίστα **url\_list** και γίνεται λογική πράξη ανάμεσα σε αυτή και την λίστα και του **result**.

Στο τέλος της συνάρτησης επιστρέφεται η λίστα **result**.

Το αρχείο **‘inverted\_index.py’** μετά το άνοιγμά του δίνεται ως όρισμα στην συνάρτηση **request\_query** η έξοδος της οποίας αποθηκεύεται στην λίστα **sites**.

**Ενδεικτική Έξοδος:**

**A screenshot of a computer program

Description automatically generated**

## β) Κατάταξη αποτελεσμάτων (Ranking):

Στο αρχείο **‘search\_engine\_2.py’** υλοποιείται ο αλγόριθμος κατάταξης **TF-IDF** και τα μοντέλα ανάκτησης **Boolean Retrieval** , **Vector Space Model (VSM),** και **Probabilistic Retrieval** (**Okapi BM25)**.

Αρχικά με την συνάρτηση **preprocess\_query** γίνεται επεξεργασία του ερωτήματος του χρήστη και συγκεκριμένα **lemmatization**, **tokenization**, αφαίρεση των **stop words**, και μετατρέπεται σε πεζά γράμματα.

Η συνάρτηση αυτή καλείται για όλα τα ερωτήματα των συναρτήσεων που υλοποιούν τους παραπάνω αλγορίθμους.

**TF-IDF:**

Ο αλγόριθμος υλοποιείται με την συνάρτηση **ranking\_TF\_IDF**. Στην συνάρτηση συνδυάζονται τα **URL** και οι παράγραφοι κάθε εγγράφου σε μία γραμμή.

Έπειτα δημιουργείται ο πίνακας **TF-IDF** των εγγράφων και το διάνυσμα **TF-IDF** του ερωτήματος με σκοπό να υπολογιστεί το **cosine similarity** τους. Με βάση αυτό ταξινομούνται τα έγγραφα και επιστρέφονται.

**Boolean Retrieval:**

Ο αλγόριθμος υλοποιείται με την συνάρτηση **boolean\_retrieval** η οποία δέχεται ως όρισμα την ερώτηση του χρήστη και το αντεστραμμένο ευρετήριο.

Στην συνέχεια γίνονται οι λογικές πράξεις που ορίζονται στο **query**, ανάμεσα στα σύνολα των **URL** κάθε **token** που αντιστοιχείται στο αντεστραμμένο ευρετήριο.

**Vector Space Model (VSM):**

Το μοντέλο ανάκτησης **VSM** υλοποιείται στην συνάρτηση **vsm\_retrieval** στην οποία αφού γίνει η προεπεξεργασία του ερωτήματος, καλείται η συνάρτηση **ranking\_TF\_IDF,** την έξοδο της οποίας και επιστρέφει.

**Probabilistic Retrieval** (**Okapi BM25):**

Ο αλγόριθμος του μοντέλου ανάκτησης υλοποιείται με την συνάρτηση **propabilistic\_retrieval**.

Στην συνάρτηση αφού γίνει συνδυασμός του κειμένου κάθε εγγράφου σε μία γραμμή, στην συνέχεια γίνεται **tokenization** του ερωτήματος και κάθε εγγράφου που δίνονται ως όρισμα.

Έπειτα υπολογίζονται οι **okapi** βαθμολογίες κάθε εγγράφου και τα έγγραφα ταξινομούνται βάσει αυτών. Η συνάρτηση επιστρέφει τα ταξινομημένα ερωτήματα.

Στην συνάρτηση **dataRetrival** γίνεται η επιλογή του μοντέλου ανάκτησης και επιστρέφεται η λίστα των **URL (result set)** της επιλεγμένης μεθόδου που δόθηκε ως όρισμα **(option)**.

Στο **main** με την χρήση μενού επιλογής ο χρήστης πληκτρολογεί το την μέθοδο ανάκτησης και το ερώτημά του στην γραμμή εντολών. Αυτά περνάνε ως ορίσματα στην **dataRetrival** με την έξοδο της να εκχωρείται στο **result\_set** το οποίο και εκτυπώνεται.

**Ενδεικτικές Έξοδοι:**

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Βήμα 5. Αξιολόγηση συστήματος:

Η αξιολόγηση του συστήματος πραγματοποιείται στο αρχείο **‘test.py’.**

**EvaluationValues:**

Στην κλάση **EvaluationValues** αποθηκεύονται και εμφανίζονται οι μετρικές αξιολόγησης **Precision**, **Recall**, **F1 Score**, **Mean Average Precision**.

**evaluate\_query:**

Η συνάρτηση **evaluate\_query** είναι υπεύθυνη για την αξιολόγηση των αποτελεσμάτων της αναζήτησης συγκρίνοντας με ένα σύνολο σωστών (**ground truth**) αποτελεσμάτων.

Αρχικά φιλτράρει από τα αποτελέσματα (**results**) κρατώντας τα έγγραφα που δεν έχουν μηδενική βαθμολογία. Αυτό στοχεύει στην απόρριψη των μη σχετικών αποτελεσμάτων. Τα αποτελέσματα αυτά αποθηκεύονται στο **relative\_results**.

Στην συνέχεια δημιουργείται η παράλληλη λίστα **y\_true** στην οποία για κάθε στοιχείο της, αν στοιχείο της αντίστοιχης θέσης του **relative\_results** δεν βρίσκεται στο **ground\_truth\_set**, τότε παίρνει τιμή **1** αλλιώς παίρνει τιμή **0**.

Η λίστα **y\_pred** δημιουργείται με παρόμοιο τρόπο, μόνο που σε αυτή την περίπτωση γίνεται έλεγχος αν το έγγραφο του **ground\_truth\_set** βρίσκεται στο **relative\_results**.

Τα μήκη των **y\_pred** και **y\_true** συγκρίνονται και αν είναι διάφορα προστίθενται μηδενικά στην **y\_pred** για να είναι ίσα.

Έπειτα δημιουργείται το αντικείμενο **results** της κλάσης **EvaluationValues** και συμπληρώνονται οι τιμές των μετρικών αξιολόγησης με τις έτοιμες συναρτήσεις της βιβλιοθήκης **sklearn.metrics**. To **results** το επιστρέφει η συνάρτηση.

**main:**

Στην **main** ανοίγονται τα αρχεία **inverted\_index.json, parsed\_scrape, ground\_truths.json**. Στην συνέχεια διαβάζεται από τον χρήστη με μενού επιλογής η μέθοδος ανάκτησης.

Στην συνέχεια διαβάζοντας από το **‘ground\_truths.json’** κάθε **query** καλείται η συνάρτηση **dataRetrival** του αρχείου **‘search\_engine\_2.py’** για την ανάκτηση των **URL**.

Τέλος καλείται η συνάρτηση **evaluate\_query** και το αποτέλεσμα της εκτυπώνεται.

**Έξοδος:**

**A screenshot of a computer program

Description automatically generated**