Αλέξανδρος Ξιάρχος 1059619

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ · ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ · 2023 – 2024

ΠΕΡΙΕΧΟΜΕΝΑ

1 ΕΙΣΑΓΩΓΗ

1.1 ΕΠΙΛΟΓΗ ΣΥΣΤΗΜΑΤΩΝ ΣΤΑΘΜΙΣΗΣ

Καταρχάς πρέπει να επιλέξουμε δύο συστήματα στάθμισης των βαρών για τους όρους των εγγράφων και των ερωτημάτων.

1.1.1 ΠΡΩΤΟ ΣΥΣΤΗΜΑ ΣΤΑΘΜΙΣΗΣ

Το πρώτο σύστημα στάθμισης θα είναι μια παραλλαγή του προτεινόμενου ως καλύτερου συστήματος σύμφωνα με τους Salton-Buckley[1] (best fully weighted system). θα χρησιμοποιήσουμε τη **απλή συχνότητα εμφάνισης** (raw term frequency) για το TF βάρος των εγγράφων:

$$\textbf{Σύστημα #1:}\ \text{TF βάρος}_{\text{εγγράφων}} = f_{i,j}$$

όπου f_{ij} οι φορές που ο όρος εμφανίζεται σε ένα έγγραφο, την **διπλή 0,5 κανονικοποίηση** για το TF βάρος των ερωτημάτων (augmented normalized TF):

$$\textbf{Σύστημα #1:}\ \text{TF βάρος}_{\text{ερωτημάτων}} = 0.5 + 0.5\frac{f_{i,j}}{max_i\ f_{i,j}}$$

και τέλος την **απλή ανάστροφη συχνότητα εμφάνισης** για το IDF βάρος και των εγγράφων και των ερωτημάτων:

$$\textbf{Σύστημα #1:}\ \text{IDF βάρος}_{\substack{\text{εγγράφων}\\ \text{ερωτημάτων}}} = \log\frac{N}{n_i}$$

όπου N το πλήθος των εγγράφων και n_i ο αριθμός των εγγράφων στα οποία εμπεριέχεται ο όρος.[2]

1.1.2 ΔΕΥΤΕΡΟ ΣΥΣΤΗΜΑ ΣΤΑΘΜΙΣΗΣ

Στο δεύτερο σύστημα στάθμισης (best weighted probabilistic weight) θα χρησιμοποιήσουμε την **διπλή 0,5 κανονικοποίηση** για το TF βάρος των εγγράφων:

$$\textbf{Σύστημα #2:}\ \text{TF βάρος}_{\text{εγγράφων}} = 1 + \log f_{i,j}$$

το **μοναδιαίο** σύστημα για το IDF βάρος των εγγράφων και την **απλή λογαριθμική κανονικοποίηση** για το IDF βάρος των ερωτημάτων:

$$\textbf{Σύστημα #2:}\ \text{IDF βάρος}_{\text{ερωτημάτων}} = \log(1 + \frac{N}{n_i})$$

Συνολικά έχουμε τα παρακάτω συστήματα:

Σύστημα στάθμισης	Βάρος όρου εγγράφου	Βάρος όρου ερωτήματος
1	$f_{i,j} \times \log\frac{N}{n_i}$	$(0.5 + 0.5\frac{f_{i,j}}{max_i\ f_{i,j}}) \times \log\frac{N}{n_i}$
2	$1 + \log f_{i,j}$	$\log(1 + \frac{N}{n_i})$

[1] Gerard Salton, Christopher Buckley, Term-weighting approaches in automatic text retrieval, Information Processing & Management, Volume 24, Issue 5, 1988, Pages 513-523, ISSN 0306-4573

[2] Το σύστημα αναφέρεται ως παραλλαγή των Salton-Buckley για το λόγο ότι δεν έχει συμπεριληφθεί κάποιος παράγοντας κανονικοποίησης, μιας και τα έγγραφα είναι *περίπου* ισομεγέθη (μέσος όρος 350 λέξεις).

2 ΥΛΟΠΟΙΗΣΗ

2.1 ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΕΓΓΡΑΦΩΝ & ΒΟΗΘΗΤΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Το αρχείο `tools.py` περιλαμβάνει βοηθητικές συναρτήσεις για κάποιες επαναλαμβανόμενες διαδικασίες της υλοποίησης. Περιλαμβάνονται οι συναρτήσεις **`get_docs`**`()` και **`get_queries`**`()`.

Η συνάρτηση **`get_docs`**`()`, χρησιμοποιώντας την `os` βιβλιοθήκη διαβάζει το πλήθος των αρχείων της βιβλιοθήκης.[3] Η συνάρτηση δημιουργεί και επιστρέφει μια λίστα από tuples, με κάθε tuple να αντιστοιχεί σε κάθε αρχείο-έγγραφο. Τα tuples έχουν την δομή:

```
('docID', ['λήμμα_1', 'λήμμα_2' ...])
```

όπου `docID` η αρίθμηση του κάθε εγγράφου και `doc_term_n` η κάθε λέξη-λήμμα του εγγράφου. Η συνάρτηση `strip()` είναι απαραίτητη για την αφαίρεση των `\n` χαρακτήρων που προέκυψαν από την μορφολογία των εγγράφων (κάθε λέξη είναι σε νέα γραμμή). Αντίστοιχα η συνάρτηση **`get_queries`**`()` επιστρέφει τη λίστα με τα ερωτήματα της συλλογής.

Στις συναρτήσεις **`preprocess_collection`**`()` και **`preprocess_queries`**`()` πραγματοποιείται η προεπεξεργασία των εγγράφων, συγκεκριμένη η αφαίρεση των **stopwords** και το **stemming**.

Η αφαίρεση των stopwords και το stemming γίνεται με τη χρήση της `nltk` βιβλιοθήκης. Τα `doc_tuples` της `get_docs()` αφού περάσουν από τον `PorterStemmer` της `nltk` αποθηκεύονται σε μια λίστα, η οποία στη συνέχεια επιστρέφεται. Αντίστοιχη διαδικασία πραγματοποιείται και για την προεπεξεργασία των ερωτημάτων, στην `preprocess_queries()`.

2.2 ΑΝΕΣΤΡΑΜΜΕΝΟ ΕΥΡΕΤΗΡΙΟ

Το ανεστραμμένο ευρετήριο δημιουργείται στη συνάρτηση **`create_inverted_index`**`()` του αρχείου `inverted_index.py`. Στην συνάρτηση εισαγάγονται οι λίστες που δημιουργήθηκαν στις προηγούμενες συναρτήσεις.

Τα tuples που αντιστοιχούν σε αυτά αποθηκεύονται σε ένα dictionary που θα αποτελέσει το ανεστραμμένο ευρετήριο με την εξής δομή:

```
                    inverted_index['λήμμα'] =
{ ('docID στο οποίο εμφανίζεται' = <φορές εμφάνισης>), (···), ... }
```

Κάθε value του dictionary είναι ένα set[4] το οποίο περιλαμβάνει ένα ή περισσότερα tuples με το `docID` και τη συχνότητα εμφάνισης του λήμματος στο συγκεκριμένο έγγραφο. Η συχνότητα υπολογίζεται μέσω της `count()` σε όλο το έγγραφο ανά λήμμα. Αυτό είναι ένα παράδειγμα του τελικού ανεστραμμένου ευρετηρίου [5]:

```
inverted_index = {... 'coronari': ('01217', 2), ('00779', 1), ('00164', 1),
'graft': ('00164', 1), 'mobil': ('00673', 2), 'strain': ('00179', 7), ...}
```

[3] Να σημειωθεί ότι το πλήθος των εγγράφων διαφέρει από την αύξουσα αρίθμησή τους. Συγκεκριμένα έχουμε 1209 έγγραφα αριθμημένα από το `000001` ως `01239`. Με άλλα λόγια υπάρχουν αριθμοί στη συλλογή που δεν αντιστοιχούν σε έγγραφα. Συνεπώς δεν θα μπορούσαμε να χρησιμοποιήσουμε κάποια αριθμητική επανάληψη, για παράδειγμα, για την εισαγωγή των εγγράφων.

[4] Έχει επιλεχθεί set για εξοικονόμιση μνήμης, μας και δεν μας ενδιαφέρει η σειρά των tuples.

[5] Τα λήμματα έχουν τη stemming μορφή τους.

2.3 ΥΛΟΠΟΙΗΣΗ VECTOR SPACE ΜΟΝΤΕΛΟΥ

Η υλοποίηση του Vector Space μοντέλου πραγματοποιείται στο αρχείο `vsm.py`, στη συνάρτηση **`run_vsm`**`()`, η οποία εναλλάσσει τα ερωτήματα, τα οποία στη συνέχεια εισάγονται στην **`vsm`**`()`, όπου υπολογίζεται όντως η ομοιότητα του συνημιτόνου των διανυσμάτων μεταξύ εγγράφου και ερωτήματος.

Στόχος είναι να υπολογίσουμε τις tf τιμές από κάθε λήμμα του εκάστοτε ερωτήματος και στη συνέχεια, αν είναι εφικτό, να τα αντιστοιχήσουμε με τα λήμματα των εγγράφων χρησιμοποιώντας το ανεστραμμένο ευρετήριο που έχουμε δημιουργήσει. Συνεπώς αφού υπολογίσουμε ...