Αλέξανδρος Ξιάρχος 1059619

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ · ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ · 2023 – 2024

ΠΕΡΙΕΧΟΜΕΝΑ

1 ΕΙΣΑΓΩΓΗ

1.1 ΥΠΟΛΟΓΙΣΜΟΣ ΒΑΡΩΝ TF & IDF

Καταρχάς πρέπει να επιλέξουμε την παραλλαγή του συστήματος αρίθμησης των βαρών TF και IDF που είναι καταλληλότερη για τη συλλογή μας.

Όσον αφορά τα **έγγραφα**: Μιας και η συλλογή αφορά βάση δεδομένων για την Κυστική Ίνωση, δηλαδή πρόκειται για συλλογή με τεχνικές –ιατρικές συγκεκριμένα– ορολογίες (technical vocabulary and meaningful terms [MED collections]) [1], θα χρησιμοποιήσουμε τη **διπλή 0,5 κανονικοποίηση** (augmented normalized TF):

$$0.5 + 0.5\frac{F_{ij}}{max_k F_{kj}}$$

για το βάρος που αφορά τα έγγραφα, όπου $\boldsymbol{F_{ij}}$ οι φορές που ο όρος εμφανίζεται σε ένα έγγραφο και $\textbf{max}_{\boldsymbol{k}}\ \boldsymbol{F_{kj}}$ το μεγαλύτερο πλήθος εμφανίσεων κάποιου όρου σε ένα έγγραφο.

Όσον αφορά τα **queries**: κάθε λήμμα από τα ερωτήματα είναι σημαντικό (σχεδόν κάθε λέξη είναι ιατρική ορολογία), άρα θα χρησιμοποιήσουμε πάλι τη **διπλή 0,5 κανονικοποίηση** για το TF βάρος.

Για το IDF βάρος και σε έγγραφα και σε ερωτήματα, χρησιμοποιούμε την **απλή ανάστροφη συχνότητα εμφάνισης**:

$$\textbf{log}\ \frac{\boldsymbol{N}}{\boldsymbol{n_i}}$$

όπου $\boldsymbol{N}$ το πλήθος των εγγράφων και $\boldsymbol{n_i}$ ο αριθμός των εγγράφων στα οποία εμπεριέχεται ο όρος.

	Βάρος όρου εγγράφου	Βάρος όρου ερωτήματος
TF	$\boldsymbol{0.5 + 0.5\frac{F_{ij}}{max_k F_{kj}}}$	$\boldsymbol{0.5 + 0.5\frac{F_{ij}}{max_k F_{kj}}}$
IDF	$\textbf{log}\ \frac{\boldsymbol{N}}{\boldsymbol{n_i}}$	$\textbf{log}\ \frac{\boldsymbol{N}}{\boldsymbol{n_i}}$

Τέλος δεν έχει προστεθεί κάποιος παράγοντας κανονικοποίησης των εγγράφων, αφού τα έγγραφα είναι περίπου ισομεγέθη (μέσος όρος 350 λέξεις).

[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 ΥΛΟΠΟΙΗΣΗ

Η υλοποίηση έχει χωριστεί στα εξής αρχεία:

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.[2] 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 ????????????

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 αποθηκεύεται ως ένα `.json` αρχείο. 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 τα `.json` αρχεία 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[3] 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[4]:

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

[2] 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.

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

[4] Τα λήμματα έχουν τη stemming μορφή τους.

2.3 ΥΛΟΠΟΙΗΣΗ VECTOR SPACE ΜΟΝΤΕΛΟΥ