

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ · ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ · 2023-2024

ΠΕΡΙΕΧΟΜΕΝΑ

1	ΕΙΣΑΓΩΓΗ	2
1.1	ΥΠΟΛΟΓΙΣΜΟΣ ΒΑΡΩΝ TF & IDF	2
2	ΥΛΟΠΟΙΗΣΗ	3
2.1	ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΕΓΓΡΑΦΩΝ & ΒΟΗΘΗΤΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ	3
2.2	ΑΝΕΣΤΡΑΜΜΕΝΟ ΕΥΡΕΤΗΡΙΟ	3
2.3	ΥΛΟΠΟΙΗΣΗ VECTOR SPACE ΜΟΝΤΕΛΟΥ	3

1 ΕΙΣΑΓΩΓΗ

1.1 ΥΠΟΛΟΓΙΣΜΟΣ ΒΑΡΩΝ TF & IDF

Καταρχάς πρέπει να επιλέξουμε την παραλλογή των βαρών TF και IDF για έγγραφα και ερωτήματα 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 F_{ij} οι φορές που ο όρος εμφανίζεται σε ένα έγγραφο και $\max_k F_{kj}$ 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**:

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

όπου N ο συνολικός αριθμός των εγγράφων και n_i ο αριθμός των εγγράφων στα οποία εμπεριέχεται ο όρος.

white	Βάρος εγγράφων	Βάρος ερωτημάτων
TF	$0.5 + 0.5 \frac{F_{ij}}{\max_k F_{kj}}$	$0.5 + 0.5 \frac{F_{ij}}{\max_k F_{kj}}$
IDF	$\log \frac{N}{n_i}$	$\log \frac{N}{n_i}$

¹Gerard 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:

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

2.3 ΥΛΟΠΟΙΗΣΗ VECTOR SPACE ΜΟΝΤΕΛΟΥ

² Να σημειωθεί ότι το πλήθος των εγγράφων διαφέρει από την αύξουσα αρίθμηση 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 επιλεγεί `set` για εξοικονόμηση μνήμης, μας και δεν μας ενδιαφέρει η σειρά των tuples.