

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ · ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ · 2023 – 2024

ΠΕΡΙΕΧΟΜΕΝΑ

1	ΕΙΣΑΓΩΓΗ	2
1.1	ΕΠΙΛΟΓΗ ΣΥΣΤΗΜΑΤΩΝ ΣΤΑΘΜΙΣΗΣ	2
1.1.1	ΠΡΩΤΟ ΣΥΣΤΗΜΑ ΣΤΑΘΜΙΣΗΣ	2
1.1.2	ΔΕΥΤΕΡΟ ΣΥΣΤΗΜΑ ΣΤΑΘΜΙΣΗΣ	2
2	ΥΛΟΠΟΙΗΣΗ	3
2.1	ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΕΓΓΡΑΦΩΝ & ΒΟΗΘΗΤΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ	3
2.2	ΑΝΕΣΤΡΑΜΜΕΝΟ ΕΥΡΕΤΗΡΙΟ	3
2.3	ΥΛΟΠΟΙΗΣΗ VECTOR SPACE ΜΟΝΤΕΛΟΥ	4

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 πλήρως σταθμισμένου συστήματος σύμφωνα με τους Salton-Buckley¹ (best fully weighted system). Θα χρησιμοποιήσουμε τη **απλή συχνότητα εμφάνισης** (raw term frequency) για το TF βάρος των εγγράφων:

$$\text{Σύστημα \#1: } TF_{\text{εγγράφων}} = f_{i,j}$$

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

$$\text{Σύστημα \#1: } TF_{\text{ερωτημάτων}} = 0.5 + 0.5 \frac{f_{i,j}}{\max_i f_{i,j}}$$

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

$$\text{Σύστημα \#1: } IDF_{\text{εγγράφων}} = \log \frac{N}{n_i}$$

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

Ως δεύτερο σύστημα στάθμισης θα χρησιμοποιήσουμε το καλύτερα σταθμισμένο πιθανολογικό σύστημα σύμφωνα με τους Salton-Buckley¹ (best weighted probabilistic weight) με

$$\text{Σύστημα \#2: } \text{βάρος όρου}_{\text{εγγράφων}} = 0.5 + 0.5 \frac{f_{i,j}}{\max_i f_{i,j}}$$

$$\text{Σύστημα \#2: } \text{βάρος όρου}_{\text{ερωτημάτων}} = \log \frac{N - n_i}{n_i}$$

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	$0.5 + 0.5 \frac{f_{i,j}}{\max_i f_{i,j}}$	$\log \frac{N - n_i}{n_i}$

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

²Το 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

Η εργασία υλοποιήθηκε σε Python χρησιμοποιώντας τις βιβλιοθήκες:

Βιβλιοθήκη	Περιγραφή
nlk	φιλτράρισμα stopwords, stemming
numpy	υπολογισμός ομοιότητας συνημιτόνου

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, συγκεκριμένα η αφαίρεση των **stopwords** και το **stemming**.

Η αφαίρεση των **stopwords** και το **stemming** γίνεται με τη χρήση της `nlk` βιβλιοθήκης. Τα `doc_tuples` της `get_docs()` αφού περάσουν από τον `PorterStemmer` της `nlk` 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

³ 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.

⁴ Έχει επιλεχθεί `set` για εξοικονόμηση μνήμης, μας και δεν μας ενδιαφέρει η σειρά των tuples.

5.

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

2.3 ΥΛΟΠΟΙΗΣΗ VECTOR SPACE ΜΟΝΤΕΛΟΥ

⁵Τα ρήγματα έχουν τη stemming μορφή τους.