ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μηχανή αναζήτησης ακαδημαϊκών εργασιών

Εργασία εργαστηρίου ανάκτηση πληροφορίας

ΔΗΜΗΤΡΙΟΣ ΧΟΡΕΒΑΣ (ΑΜ: 20390261)

Ημερομηνία διεξαγωγής: 27/11/23
Ημερομηνία παράδοσης: 29/1/24

ΑΘΗΝΑ 2024

Περιεχόμενα

Εισαγωγή

Στην εποχή της πληροφορίας (information age[1]) πολλοί είναι οι άνθρωποι οι οποίοι επιλέγουν να παίρνουν τις πληροφορίες που χρειάζονται μέσω του διαδικτύου από συστήματα ανάκτησης πληροφοριών. Ενδεικτικά, η εταιρία-κολοσσός διαδικτυακών υπηρεσιών Γκουγκλ (Google), η οποία παρέχει -μεταξύ άλλων- δωρεάν υπηρεσία ανάκτησης πληροφοριών, ανακοίνωσε το 2012 ότι εκείνον τον χρόνο έγιναν 1,2 τρισεκατομμύριο αναζητήσεις με χρήση αυτής της υπηρεσίας [3]. Συνεπώς, γίνεται αντιληπτό ότι η ανάκτηση πληροφορίας αποτελεί σημαντικό στοιχείο του σύγχρονου τρόπου διαβίωσης.

Στο παρόν έγγραφο θα αναλυθεί ο σχεδιασμός και η αξιολόγηση μιας απλής μηχανής αναζήτησης ακαδημαϊκών εργασιών. Αναλυτικότερα, θα επεξηγηθεί η διαδικασία ανάπτυξης του λογισμικού, η λειτουργία των αλγόριθμων οι οποίοι χρησιμοποιήθηκαν και η μεθοδολογία αξιολόγησης ενός συστήματος ανάκτησης πληροφοριών (ΣΑΠ) σε θεωρητικό επίπεδο.

Σταχυολογητής

Αρχικά, πρέπει να συλλεχθούν τα έγγραφα προς χρήση. Αυτο επιτυγχάνεται με τη χρήση ενός σταχυολογητή. Ο σταχυολογητής συλλέγει υλικό από το αποθετήριο ιατρικών ακαδημαϊκών εργασιών “PubMed”[2] με χρήση της υπάρχουσας μηχανής αναζήτησης.

Στην συνέχεια, ξεκινά η συλλογή των δεδομένων. Συγκεκριμένα, καταγράφεται ο αναγνωριστικός αριθμός (PMID), ο τίτλος, οι συγγραφείς, η ημερομηνία δημοσίευσης και η περίληψη της εκάστοτε εργασίας. Τα δεδομένα εμφανίζονται σε μορφή απλού κειμένου (plain text) με την χρήση της επιλογής `&format=pubmed`, η οποία είναι πιο εύκολη στην επεξεργασία σε σχέση με την προεπιλεγμένη μορφή “HTML”. Επιπλέον, χρησιμοποιούνται οι παράμετροι `&page` για τον καθορισμό της σελίδας αποτελεσμάτων της αναζήτησης και `&size` για τον καθορισμό της ποσότητας των αποτελεσμάτων σε κάθε σελίδα.

Τα δεδομένα συλλέγονται με μαζικό τρόπο και αποθηκεύονται τοπικά σε μορφή “JavaScript Object Notation” (JSON)[3]. Με αυτόν τον τρόπο, γίνεται εύκολη η δόμηση και η ανάγνωση των δεδομένων. Αξίζει να σημειωθεί η αναγκαιότητα ανάπτυξης αποδοτικού αλγόριθμου αποθήκευσης των δεδομένων. Λαμβάνοντας υπόψη ότι χρειάζεται να αποθηκευτεί μεγάλος όγκος δεδομένων, η μετατροπή σε μορφή JSON και η αποθήκευση σε αρχείο γίνεται με σταδιακό τρόπο. Έτσι, επιτυγχάνεται η λιγότερη κατανάλωση μνήμης από το πρόγραμμα.

Υπάρχει ακόμη μεγαλύτερο περιθώριο βελτίωσης των επιδόσεων του προγράμματος. Αναλυτικά, μπορεί να γίνει και η φάση της συλλογής των δεδομένων με σταδιακό τρόπο. Έτσι, για κάθε εργασία που συλλέγεται, θα μετατρέπεται άμεσα σε μορφή JSON και έπειτα θα αποθηκεύεται σε αρχείο.

Προ-επεξεργασία κειμένου

Προτού χρησιμοποιηθεί το κείμενο από το ΣΑΠ, τροποποιείται με κατάλληλο τρόπο έτσι ώστε να είναι σε κανονική μορφή. Αυτή η διαδικασία αποτελείται από πολλαπλά στάδια όπου στο κάθε ένα από αυτά μετατρέπεται το εκάστοτε κείμενο σε τέτοια μορφή ώστε να γίνει ευκολότερη η διαδικασία της ανάκτησης πληροφορίας δοθέντος ενός ερωτήματος (query).

Η προ-επεξεργασία εκτελείται από την μέθοδο `TextProcessing.process` της βιβλιοθήκης `textprocessing`. Οι λειτουργίες της αναλύονται στις επόμενες παραγράφους.

[1] Είναι η σημερινή εποχή, στην οποία μεγάλος όγκος πληροφορίας είναι διαθέσιμος λόγω της ανάπτυξης της τεχνολογίας των υπολογιστών [1]. Ο όρος χρησιμοποιήθηκε πρώτη φορά από τον R. S. Leghorn το 1960 [2].

[2] https://pubmed.ncbi.nlm.nih.gov/

[3] https://www.json.org/json-en.html

Ορισμοί

Σύμβολο (token) Μία πεπερασμένη σειρά χαρακτήρων ενός κειμένου η οποία θεωρείται κατάλληλη για επεξεργασία.

Τύπος (type) Η τάξη όλων των όμοιων συμβόλων.

Λεξικό (dictionary) Σύνολο λέξεων.

Όρος (term) Ένας τύπος ο οποίος έχει υποστεί κανονικοποίηση και ανήκει στο λεξικό ενός ΣΑΠ.

Διαίρεση σε σύμβολα (tokenization) Η διαδικασία τεμαχισμού του κειμένου σε σύμβολα.

[4]

Διαίρεση σε σύμβολα

Αρχικά, διαιρείται το κείμενο σε σύμβολα. Στην προκειμένη περίπτωση τα σύμβολα είναι λέξεις οι οποίες χωρίζονται από κενά (" "). Αν και εκ πρώτης όψης, αυτό φαίνεται να επαρκεί, ο γραπτός λόγος πολλές φορές περιέχει σημεία στίξης (όπως κόμματα ή θαυμαστικά). Αν η διαίρεση περιορίζεται μόνο σε κενά, τότε τα σύμβολα ενδεχομένως να περιέχουν σημεία στίξης στην αρχή, στο τέλος, ή ακόμη και μέσα σε μία λέξη. Όμως, τα σημεία στίξης συνήθως δεν αποτελούν μέρος της λέξης και συνεπώς δεν πρέπει να συμπεριλαμβάνονται σε σύμβολο λέξης[4]. Συμπερασματικά, ένα ΣΑΠ πρέπει να αναγνωρίζει αυτούς τους χαρακτήρες και να τους απομονώνει ή να τους αφαιρεί όποτε κρίνεται απαραίτητο.

Η βιβλιοθήκη `nltk` διαθέτει την συνάρτηση `nltk.tokenize.word_tokenize` η οποία μετατρέπει ένα κείμενο σε μορφή `str` σε μορφή `list[str]`. Κάθε στοιχείο της λίστας αποτελεί ένα σύμβολο του κειμένου. Η χρήση αυτής της συνάρτησης είναι πιο αποτελεσματική από την `str.split()` της "Python", για τους λόγους που εξηγήθηκαν στην προηγούμενη παράγραφο.

Τέλος, η συνάρτηση `nltk.tokenize.word_tokenize` ενθυλακώνεται στην `TextProcessing.tokenize` της βιβλιοθήκης `textprocessing`. Η ενθυλάκωση συμβάλει στην ευκολότερη τροποποίηση και συντήρηση του κώδικα.

Παράληψη τύπων

Ορισμός

Διακόπτουσα λέξη Η τετριμμένη λέξη. Λόγου χάρη: "ή", "και", "ένας" κτλ.

[4]

Ενίοτε, ορισμένοι τύποι συμβόλων δεν παρέχουν χρήσιμες πληροφορίες για ένα έγγραφο. Αυτοί οι τύποι μπορεί να είναι σημεία στίξης ή διακόπτουσες λέξεις. Αν ένα τέτοιο σύμβολο δεν είναι μείζονος σημασίας για ένα έγγραφο, τότε μπορεί να παραληφθεί με ασφάλεια. Με αυτόν τον τρόπο, δεν επιβαρύνεται με περιττούς όρους το λεξικό του ΣΑΠ [4].

Η βιβλιοθήκη `string` της "Python" διαθέτει μία συμβολοσειρά η οποία περιέχει συνηθισμένα σημεία στίξης τα οποία βρίσκονται στο πληκτρολόγιο. Με την μέθοδο `TextProcessing.is_special` εντοπίζονται εύκολα τα σύμβολα με ειδικούς χαρακτήρες.

[4] Παρόλα αυτά υπάρχουν εξαιρέσεις. Ενδεικτικά, ενα ΣΑΠ το οποίο διαχειρίζεται έγγραφα γραμμένα στα ελληνικά *δεν* πρέπει να χωρίσει την λέξη "ό,τι" (=οτιδήποτε) σε "ο", "," και "τι".

Η βιβλιοθήκη `nltk` διαθέτει μία λίστα από διακόπτουσες λέξεις της αγγλικής γλώσσας. Η μέθοδος `TextProcessing.is_stopword` εντοπίζει αν υπάρχει διακόπτουσα λέξη σε μια συμβολοσειρά με βάση την λίστα της `nltk`.

Μετατροπή σε πεζά

Ένα μέρος της προ-επεξεργασίας αποτελεί η μετατροπή των γραμμάτων ενός συμβόλου λέξης σε πεζά. Αυτό έχει ως αποτέλεσμα μια λέξη που βρίσκεται στην αρχή της πρότασης -η οποία ξεκινάει πάντα με κεφαλαίο γράμμα- να ταυτίζεται με την ίδια λέξη στην μέση μιας πρότασης η οποία ξεκινάει με πεζό γράμμα. Με αυτόν τον τρόπο επιτυγχάνεται η αποφυγή δημιουργίας διπλότυπων όρων στο λεξικό.

Αξίζει να σημειωθεί ότι υπάρχουν περιπτώσεις στις οποίες αυτή η πρακτική πιθανώς να δημιουργήσει προβλήματα. Λόγου χάρη, το σύμβολο "ΕΣΥ" (=Εθνικό Σύστημα Υγείας, αρχικά "Ε.Σ.Υ." πριν την παράληψη των τελειών σε προηγούμενο στάδιο) *δεν* πρέπει να μετατραπεί σε "εσυ" (προσωπική αντωνυμία, ονομαστική πτώση β' προσώπου), αφού αλλάζει το νόημα της λέξης. Επομένως, ορισμένες λέξεις πρέπει να διατηρούν την αρχική γραφή τους έτσι ώστε να μην συγχέονται με άλλες λέξεις της γλώσσας.

Στην εργασία επιλέχθηκε να χρησιμοποιηθεί η συνάρτηση `str.lower` της "Python", η οποία μετατρέπει οποιαδήποτε συμβολοσειρά σε πεζά γράμματα, λόγω της απλότητάς της.

Περιστολή και λημματοποίηση

Ορισμοί
Θέμα Το αρχικό μέρος της λέξης, το οποίο είναι σταθερό [5].
Κατάληξη Το τελικό μέρος της λέξης, το οποίο είναι μεταβλητό [5].
Λήμμα Μία λέξη μέσω της οποίας παράγονται άλλες λέξεις [6].

Μία λέξη μπορεί να έχει διαφορετική μορφή ανάλογα με την θέση της σε μία πρόταση. Οι λέξεις αλλάζουν στην *κατάληξη* τους αφήνοντας το *θέμα* σταθερό. Στην ελληνική γλώσσα τα κλητά μέρη του λόγου[5] διαθέτουν αυτό το χαρακτηριστικό. Για παράδειγμα, ένα ουσιαστικό ή μια αντωνυμία είναι σε *ονομαστική πτώση* όταν αποτελεί το υποκείμενο μιας πρότασης, ενώ είναι σε *αιτιατική πτώση* όταν αποτελεί το (άμεσο) αντικείμενο μιας πρότασης.

Οι λέξεις πρέπει να καταχωρούνται με συγκεκριμένη μορφή έτσι ώστε να γίνεται ορθή η καταγραφή ενός όρου. Αυτό μπορεί να επιτευχθεί με χρήση μίας εκ των δύο παρακάτω τεχνικών: της *περιστολής* και της *λημματοποίησης*.

Ορισμοί
Περιστολή (stemming) Η διαδικασία της αποκοπής της κατάληξης μιας λέξης και διατήρησης μόνο του θέματός της.
Λημματοποίηση (lemmatization) Η διαδικασία μετατροπής μιας λέξης σε λήμμα.
[4]

Σε αρκετές περιπτώσεις προτιμάται η λημματοποίηση έναντι της περιστολής. Χρησιμοποιώντας λημματοποίηση διατηρείται η συγγένεια μιας λέξης με το λήμμα της. Για παράδειγμα, η λέξη "είπα" (αόριστος του ρήματος "λέω") μετατρέπεται στην λέξη "λέω". Εν αντιθέσει, με χρήση περιστολής προκύπτουν οι λέξεις "είπ" και "λέ" και κατά συνέπεια χάνεται η γλωσσολογική σύνδεσή τους. Για αυτόν τον λόγο, ένας αλγόριθμος λημματοποίησης παράγει αποτελέσματα

[5] Άρθρα, Ουσιαστικά, επίθετα και ρήματα.

μεγαλύτερης ακρίβειας σε σύγκριση με έναν περιστολής. Παρόλα αυτά, οι αλγόριθμοι περιστολής είναι συγκριτικά ταχύτεροι [7]–[10].

Η μέθοδος `TextProcessing.stem` ενθυλακώνει την μέθοδο `nltk.PorterStemmer.stem` η οποία εφαρμόζει τον αλγόριθμο του Πόρτερ [11] για περιστολή των λέξεων. Η μέθοδος `TextProcessing.lemmatize` ενθυλακώνει την μέθοδο `nltk.WordNetLemmatizer.lemmatize` η οποία χρησιμοποιεί την υπηρεσία "WordNet"[6] για λημματοποίηση των λέξεων.

Ευρετήριο

Το ευρετήριο λειτουργεί ως ένα λεξικό το οποίο ταυτίζει έναν όρο με ένα σύνολο εγγράφων. Συγκεκριμένα, το ευρετήριο διαβάζει τις (επεξεργασμένες) λέξεις ενός έγγραφου και για κάθε νέο σύμβολο το οποίο βρίσκει δημιουργεί μια νέα καταχώρηση με τον όρο και το έγγραφο στο οποίο συναντάται. Αν βρεθεί ο ίδιος όρος σε διαφορετικό έγγραφο, τότε προστίθεται το νέο έγγραφο στο σύνολο με τα υπόλοιπα. Με αυτόν τον τρόπο, εύκολα μαθαίνει κανείς σε *πόσα* και σε *ποια* έγγραφα βρίσκεται ένας όρος που ανήκει στο ευρετήριο.

Ένα ευρετήριο μπορεί να παρέχει παραπάνω πληροφορίες σχετικά με έναν όσο. Για παράδειγμα, θα μπορούσε να αποθηκεύονται και οι θέσεις στις οποίες βρίσκεται ένας όρος σε ένα έγγραφο. Έτσι, μπορούν να μελετηθούν και τα συμφραζόμενα ενός όρου. Προφανώς, όσο περισσότερες είναι οι πληροφορίες οι οποίες αποθηκεύονται, τόσο περισσότερη μνήμη χρειάζεται το ευρετήριο [4].

Η βιβλιοθήκη `index` διαθέτει την κλάση `InvertedIndex` η οποία επεκτείνει (κληρονομεί) την κλάση `nltk.text.TextCollection` η οποία υλοποιεί τις βασικές δυνατότητες του ευρετηρίου. Επιπλέον, διαθέτει την μέθοδο `contains`, η οποία δέχεται ένα σύμβολο επιστρέφει σύνολο έγγραφων στα οποία υπάρχει ως όρος, την μέθοδο `not_contains`, η οποία είναι το συμπλήρωμα της προηγούμενης, και την μέθοδο `average_length`, η οποία επιστρέφει τον μέσο όσο του μήκους όλων των κειμένων στο λεξικό. Οι μέθοδοι θα αξιοποιηθούν από τους αλγόριθμους αναζήτησης οι οποίοι θα αναλυθούν στις επόμενες παραγράφους.

Μηχανή αναζήτησης

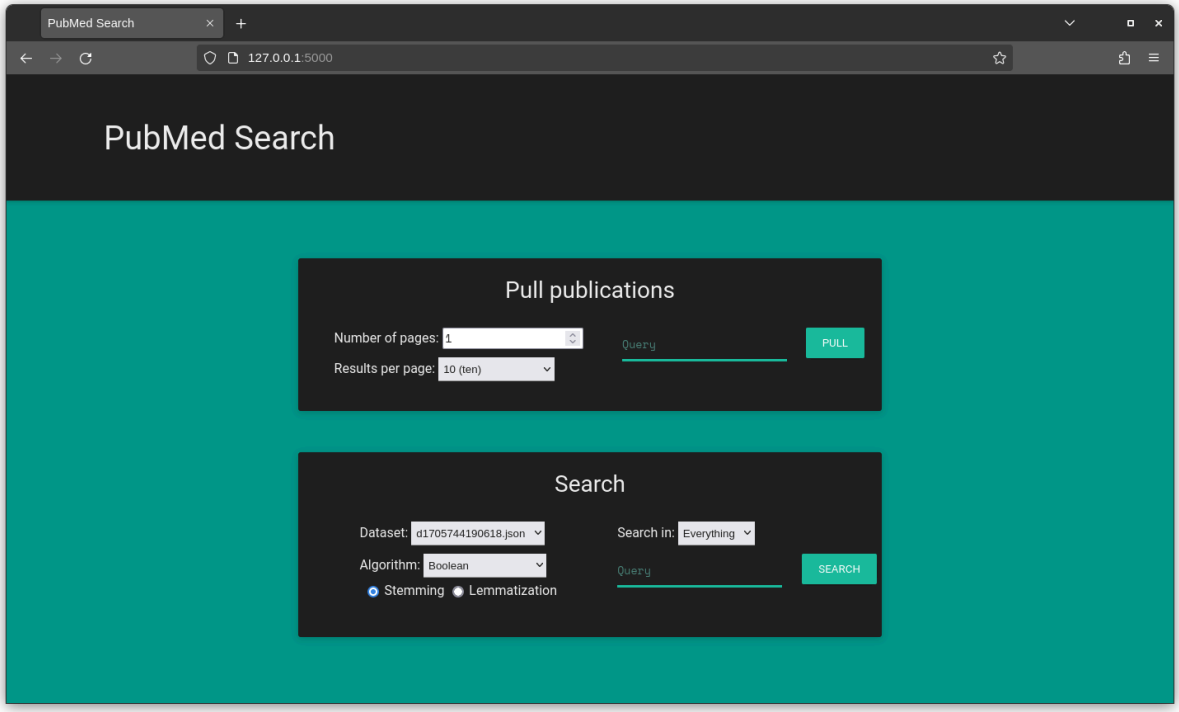
Η μηχανή αναζήτησης επιτρέπει την εύρεση ακαδημαϊκών εργασιών με βάση ένα ερώτημα (query). Αποτελείται από μία φιλική διεπαφή χρήστη, η οποία είναι υπεύθυνη για την εμφάνιση των αποτελεσμάτων, και τους αλγόριθμους αναζήτησης, οι οποίοι εκτελούν τα απαραίτητα βήματα για την εύρεση των σχετικών εγγράφων.

Για την έναρξη του προγράμματος είναι αναγκαία η εγκατάσταση των προαπαιτούμενων βιβλιοθηκών. Σε "Unix-like" λειτουργικά συστήματα γίνεται με χρήση του `pip`[7] και συγκεκριμένα της εντολής `pip install -r requirements.txt`. Το αρχείο `requirements.txt` βρίσκεται στον φάκελο με τον κώδικα.

Το πρόγραμμα εκτελείται με την εντολή `python main.py` και η διεπαφή είναι προσβάσιμη στην διεύθυνση http://127.0.0.1:5000. Ενδείκνυται η χρήση ενός σύγχρονου περιηγητή ιστού (web browser).

[6] https://wordnet.princeton.edu/
[7] https://pypi.org/project/pip/

Διεπαφή χρήστη



Εικόνα 1: Διεπαφή χρήστη

Η διεπαφή χρήστη αποτελείται από δύο μέρη:

- τον σταχυολογητή (κάρτα "Pull publications"), ο οποίος είναι υπεύθυνος για την λήψη και αποθήκευση των εργασιών, και
- την μηχανή αναζήτησης (κάρτα "Search"), η οποία επιτρέπει την αναζήτηση ακαδημαϊκών εργασιών σε ένα σύνολο αποθηκευμένων εγγράφων

Κάρτα "Pull publications"

Η κάρτα "Pull publications" αποτελείται από τα εξής πεδία:

- Αριθμός σελίδων (**Number of pages**): όπου επιλέγεται ο αριθμός σελίδων με ακαδημαϊκές εργασίες
- Αποτελέσματα ανά σελίδα (**Results per page**) που θα έχει κάθε σελίδα *το πολύ*
- Πεδίο ερωτήματος (**Query**): όπου εισάγεται το ερώτημα

Όταν ο χρήστης πατά το κουμπί "Pull", γίνεται λήψη και αποθήκευση των εργασιών. Τα έγγραφα αποθηκεύονται στον υπο-φάκελο `data` σε μορφή αρχείου JSON με μοναδικό όνομα.

Κάρτα "Search"

Η κάρτα "Search" αποτελείται από τα εξής πεδία:

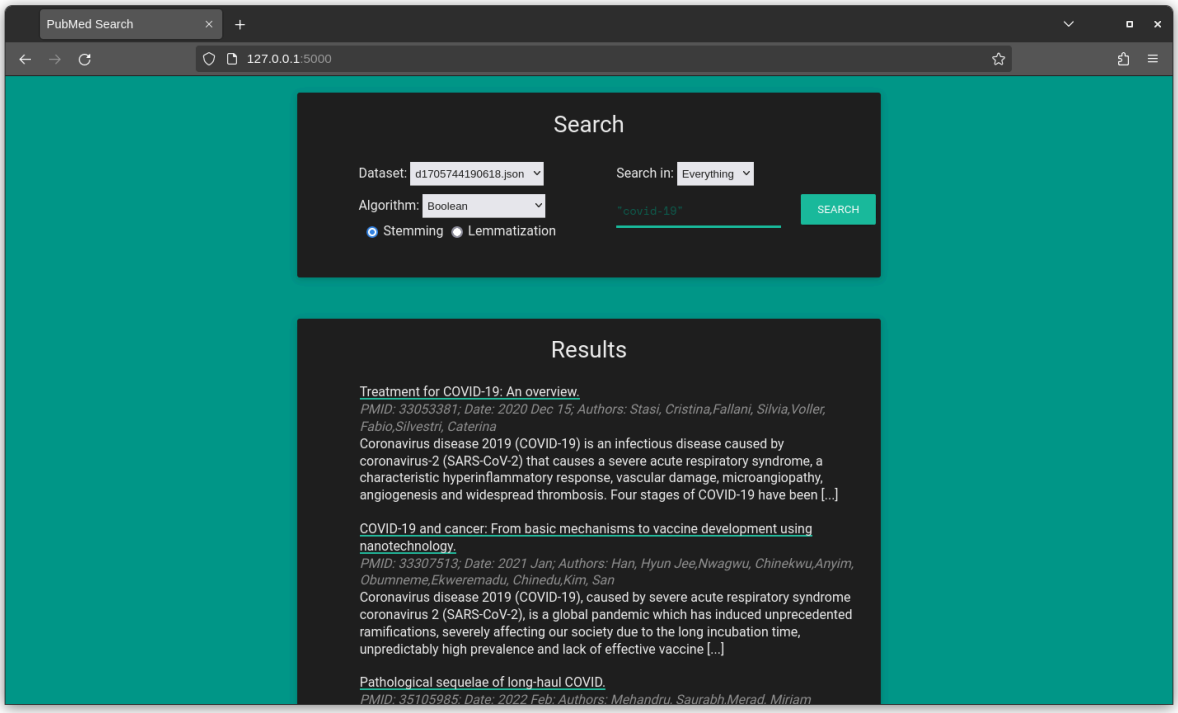
- Σύνολο δεδομένων (**Dataset**): όπου επιλέγεται το σύνολο των αποθηκευμένων δεδομένων
- Αλγόριθμος (**Algorithm**): όπου επιλέγεται ο αλγόριθμος αναζήτησης
- Περιστολή ή λημματοποίηση (**Stemming or Lemmatization**)
- Αναζήτηση σε πεδίο (**Search in field**): όπου επιλέγεται σε ποια πεδία των εργασιών θα γίνει αναζήτηση (φίλτρο)
- Πεδίο ερωτήματος (**Query**): όπου εισάγεται το ερώτημα

Όταν ο χρήστης πατά το κουμπί "Search", φορτώνεται στην μνήμη το σύνολο δεδομένων, γίνεται η επεξεργασία των δεδομένων και του ερωτήματος, βρίσκει ο εκάστοτε αλγόριθμος τα κατάλληλα αποτελέσματα και παρουσιάζονται στην κάρτα "Results".

Η αναζήτηση μπορεί να γίνει στο τίτλο, στους συγγραφείς, στην ημερομηνία, στην περίληψη ή σε όλα τα προαναφερθέντα πεδία.

Προφανώς, είναι αναγκαίο να υπάρχει ήδη κάποιο σύνολο δεδομένων προτού γίνει αναζήτηση.

Κάρτα "Results"



Εικόνα 2: Κάρτα "Results"

Σε αυτήν την κάρτα εμφανίζονται τα αποτελέσματα της αναζήτησης. Για κάθε αποτέλεσμα εμφανίζεται ο σύνδεσμος προς τον ιστότοπο PubMed και τα μετα-δεδομένα της εργασίας. Δεν εμφανίζεται ολόκληρη η περίληψη επειδή σε ορισμένες περιπτώσεις οι περιλήψεις είναι αρκετά εκτενείς.

Η κάρτα εμφανίζεται μόνο αν έχει γίνει επιτυχής αναζήτηση.

Αλγόριθμοι αναζήτησης

Χρησιμοποιούνται τρεις αλγόριθμοι αναζήτησης, ονομαστικά:

- Ανάκτηση Boole
- Ανάκτηση διανυσματικού χώρου
- Ανάκτηση Okapi BM25

Πριν την εκτέλεση ενός αλγορίθμου, τα έγγραφα έχουν φορτωθεί στην μνήμη και έχει εμπλουτιστεί το λεξικό με όρους.

Ανάκτηση Boole

Στην ανάκτηση Boole επιστρέφονται τα έγγραφα στα οποία βρίσκονται οι όροι του ερωτήματος. Αν ένας όρος ανήκει σε ένα έγγραφο, τότε προστίθεται στο σύνολο των αποτελεσμάτων χωρίς να ληφθεί υπόψη κάποιος άλλος παράγοντας. Επιπλέον, τα ερωτήματα έχουν την μορφή λογικής έκφρασης, δηλαδή μπορεί να περιέχουν πράξεις όπως "και" (and, intersection), "ή" (or, union) και "όχι" (not, difference)[4].

Το ΣΑΠ υποστηρίζει όλες τις βασικές λογικές πράξεις, ονομαστικά:

- πράξη "και" με χρήση του τελεστή &

- πράξη “ή” με χρήση του τελεστή `|`
- πράξη “όχι” με χρήση του τελεστή `!`

και παρενθέσεις `(`, `)`.

Οι λέξεις πρέπει να βρίσκονται μέσα σε εισαγωγικά, είτε `""` είτε `''`. Αν υπάρχουν παραπάνω από μία λέξεις, τότε η έκφραση μετατρέπεται σε πολλές λέξεις χωρισμένες απο τελεστές “και”[8].

Για παράδειγμα, έστω η έκφραση `"covid-19" & "lung"`. Το ΣΑΠ:
1. θα εντοπίσει το σύνολο των εγγράφων που περιέχουν (τουλάχιστον μία φορά) τον όρο “covid-19”,
2. θα εντοπίσει το σύνολο των εγγράφων που περιέχουν (τουλάχιστον μία φορά) τον όρο “lung”,
3. θα βρει την τομή των δύο συνόλων και
4. θα επιστρέψει το τελικό σύνολο εγγράφων.

Αν υπάρχει ο τελεστής “όχι” πριν από έναν όρο, τότε θα εντοπιστεί το σύνολο των εγγράφων στα οποία *δεν* υπάρχει ο όρος.

Ένα πιο σύνθετο παράδειγμα είναι το εξής: έστω η έκφραση `"covid-19" & !"lung" & !"heart"`. Το ΣΑΠ:
1. θα εντοπίσει το σύνολο των εγγράφων που περιέχουν (τουλάχιστον μία φορά) τον όρο “covid-19”,
2. θα εντοπίσει το σύνολο των εγγράφων που *δεν* περιέχουν (ούτε μία φορά) τον όρο “lung”,
3. θα εντοπίσει το σύνολο των εγγράφων που *δεν* περιέχουν (ούτε μία φορά) τον όρο “heart”,
4. θα βρει την τομή των τριών συνόλων και
5. θα επιστρέψει το τελικό σύνολο εγγράφων.

Αναλυτικότερα, η ανάκτηση Boole του ΣΑΠ λειτουργεί με τον εξής τρόπο:
1. η συνάρτηση `boolean.Lexer.tokenize` αναγνωρίζει τους τελεστές και τα σύμβολα του ερωτήματος,
2. η συνάρτηση `boolean.check_syntax` ελέγχει αν έχει γίνει σωστή σύνταξη των τελεστών,
3. η συνάρτηση `boolean.DeMorgan.convert` μετατρέπει εκφράσεις της μορφής $\neg(A \wedge B)$ σε $(\neg A \vee \neg B)$ και $\neg(A \vee B)$ σε $(\neg A \wedge \neg B)$ [12]

   Για παράδειγμα, η έκφραση `"covid-19" & !("lung" | "heart")` μετατρέπεται αυτόματα σε `"covid-19" & (!"lung" & !"heart")`. Έτσι οι τελεστές “όχι” βρίσκονται πάντοτε πριν από σύμβολα λέξεων και όχι από αριστερή παρένθεση. Ο λόγος που γίνεται αυτό θα γίνει αντιληπτός πιο κάτω.
4. Στην συνέχεια, η συνάρτηση `boolean.LinguisticProcessor.apply` εκτελεί την προ-επεξεργασία του ερωτήματος έτσι ώστε να είναι σε κανονική μορφή,
5. η συνάρτηση `boolean.convert_to_sets` μετατρέπει τα σύμβολα λέξεων σε σύμβολα. Αν ένα σύμβολο λέξης έχει τον τελεστή “όχι” πριν από αυτό, τότε γίνεται κλήση της μεθόδου `index.InvertedIndex.not_contains`, αλλιώς γίνεται κλήση της μεθόδου `index.InvertedIndex.contains`.
6. Τέλος, γίνεται κλήση της `boolean.Evaluate.evaluate_infix`, η οποία βρίσκει το τελικό σύνολο.

Ανάκτηση διανυσματικού χώρου

Η ανάκτηση διανυσματικού χώρου εκφράζει τους όρους του ερωτήματος και των εγγράφων με χρήση διανυσμάτων. Αυτοί οι αλγόριθμοι ανήκουν σε μια ευρύτερη οικογένεια αλγορίθμων η οποία ονομάζεται “Σάκος λέξεων”, αφού ασχολούνται με το *πόσες φορές* εμφανίζονται οι όροι σε ένα

[8] Για παράδειγμα, το `"covid-19 lung"` μετατρέπεται σε `("covid-19" & "lung")`

έγγραφο παρά η σειρά τους. Σε αντίθεση με την ανάκτηση Boole, αυτοί οι αλγόριθμοι χρησιμοποιούν μεθόδους κατάταξης των αποτελεσμάτων, εμφανίζοντας τα πιο σχετικά πιο ψηλά σε σχέση με τα λιγότερο σχετικά [4].

Ορισμοί

Συχνότητα όρου (tf) μετρά πόσες φορές έχει εμφανιστεί ένας όρος σε ένα έγγραφο.

Συχνότητα εγγράφων (df) μετρά σε πόσα έγγραφα υπάρχει ένας όρος.

Αντίστροφη συχνότητα εγγράφων (idf) ο λογάριθμος του πλήθους των εγγράφων ως προς την συχνότητα εγγράφων ενός όρου.

Στάθμιση tf-idf το γινόμενο της συχνότητας όρου επί της αντίστροφης συχνότητας εγγράφων.

[4]

Αξίζει να σημειωθεί η σημασία της αντίστροφης συχνότητας εγγράφων, αφού δίνει μεγαλύτερη σημασία σε όρους οι οποίοι είναι σπανιότεροι στην συλλογή εγγράφων. Αν και εκ πρώτης όψης αυτό φαίνεται αντιπαραγωγικό, οι σπανιότεροι όροι ενδεχομένως να είναι πιο χρήσιμοι από τους συχνούς. Για παράδειγμα, έστω το ερώτημα "herpes virus". Η λέξη "virus" (που σημαίνει "ιός") είναι πάρα πολύ συχνή στην επιστήμη στης Βιολογίας[9]. Συνεπώς, πρέπει να δοθεί λιγότερη σημασία στην λέξη "virus" και περισσότερη στην λέξη "herpes" η οποία είναι πιο συγκεκριμένη.

Στο ΣΑΠ χρησιμοποιείται στάθμιση tf-idf για την βαθμολόγηση των εγγράφων. Συγκεκριμένα, η βιβλιοθήκη `vector` χρησιμοποιεί στην συνάρτηση `ntlk.text.tf_idf` και στην συνέχεια προσθέτει τους συντελεστές κάθε όρου έτσι ώστε να βρεθεί η τελική βαθμολογία. Τέλος, η λίστα ταξινομείται κατά *φθίνουσα* σειρά έτσι ώστε να εμφανιστούν τα πιο σχετικά αποτελέσματα πρώτα.

Ανάκτηση Okapi BM25

Η στάθμιση Okapi BM25 είναι μία συνάρτηση κατάταξης αποτελεσμάτων. Όπως και η tf-idf, ανήκει και αυτή στην οικογένεια αλγορίθμων "Σάκος λέξεων". Η εξίσωση η οποία χρησιμοποιήθηκε στο ΣΑΠ είναι η εξής:

$$\mathrm{BM25}_d = \sum_{t \in q} \left[\log \frac{N}{\mathrm{df}_t}\right] \cdot \frac{(k_1+1)\mathrm{tf}_{\mathrm{td}}}{k_1\left((1-b)+b\cdot\left(\frac{L_d}{L_{\mathrm{ave}}}\right)\right)+\mathrm{tf}_{\mathrm{td}}} \cdot \frac{(k_3+1)*\mathrm{tf}_{\mathrm{tq}}}{k_3+\mathrm{tf}_{\mathrm{tq}}}$$ [4]

Στο πρώτο μέρος της εξίσωσης υπάρχει η αντίστροφη συχνότητα εγγράφων. Το $\mathrm{tf}_{\mathrm{td}}$ εκφράζει πόσοι όροι t υπάρχουν στο έγγραφο d, ενώ το $\mathrm{tf}_{\mathrm{tq}}$ εκφράζει πόσοι όροι t υπάρχουν στο ερώτημα q. Το L_d είναι ο αριθμός όρων του εγγράφου d, ενώ το L_{ave} είναι ο μέσος όρος του αριθμού των όρων των εγγράφων στη συλλογή. Τέλος, οι σταθερές k_1, k_3 έχουν τιμή 1.2, ενώ η σταθερά b έχει τιμή 0.75, αφού έχει μελετηθεί ότι αυτές είναι αποδοτικές τιμές [4].

Η συνάρτηση βρίσκεται στην βιβλιοθήκη `okapi` και επιστρέφει ταξινομημένη λίστα με τα πιο σχετικά έγγραφα πρώτα.

Αξιολόγηση συστήματος

Για την ορθή αξιολόγηση ενός ΣΑΠ είναι αναγκαία τα εξής:

1. μία συλλογή εγγράφων,
2. μία συλλογή ερωτημάτων,
3. την συνάφεια ερωτήματος-εγγράφου. Ένα έγγραφο μπορεί να είναι μόνο *συναφές* ή *μη-συναφές*.

Η συλλογή εγγράφων είναι μία λίστα με δομές οι οποίες φορτώνονται από αρχείο μορφής JSON. Αυτό έχει ήδη υλοποιηθεί.

[9] Εκτιμάται ότι υπάρχουν τουλάχιστον περίπου 320.000 (!) είδη ιών που προσβάλουν τα *θηλαστικά* [13].

Η συλλογή ερωτημάτων θα μπορούσε να είναι απλώς μια λίστα συμβολοσειρών (τύπου `str`) η οποία να φορτώνεται από plain-text αρχείο.

Η συνάφεια ερωτήματος-εγγράφου μπορεί να εκφραστεί ως ένας δισδιάστατος πίνακας. Θα μπορούσε να φορτώνεται απο αρχείο απλού κειμένου ένας πίνακας δυαδικών τιμών (flags) όπου "1" θα σημαίνει *συναφές* ενώ "0" θα σημαίνει *μη-συναφές*.

Η αξιολόγηση ξεκινάει παρατηρώντας τα έγγραφα τα οποία ανακτήθηκαν καταγράφοντας τα αποτελέσματα με βάση τον παρακάτω πίνακα:

	συναφή	*μη-συναφή*
ανακτημένα	Αληθώς θετικά	Ψευδώς θετικά
μη-ανακτημένα	Ψευδώς αρνητικά	Αληθώς αρνητικά

- Τα *συναφή* αποτελέσματα τα οποία *ανακτήθηκαν* καταγράφονται ως αληθώς θετικά.
- Τα *μη-συναφή* αποτελέσματα τα οποία *ανακτήθηκαν* καταγράφονται ως ψευδώς θετικά.
- Τα *συναφή* αποτελέσματα τα οποία *δεν ανακτήθηκαν* καταγράφονται ως ψευδώς αρνητικά.
- Τα *μη-συναφή* αποτελέσματα τα οποία *δεν ανακτήθηκαν* καταγράφονται ως αληθώς αρνητικά.

Με βάση αυτόν τον πίνακα μπορούν να οριστούν τα εξής:
- $\text{Ακρίβεια (P)} = \frac{\text{Αληθώς θετικά}}{\text{ανακτημένα}}$
- $\text{Ανάκληση (R)} = \frac{\text{Αληθώς θετικά}}{\text{συναφή}}$

[4]

Ένα ΣΑΠ μπορεί να αξιολογηθεί με βάση τις προτεραιότητες του χρήστη. Αν ένας χρήστης χρειάζεται να ανακτά μεγάλη ποσότητα εγγράφων, τότε ενδεχομένως να χρειάζεται ένα ΣΑΠ με μεγάλη ανάκληση έναντι ακρίβειας. Αντιθέτως, αν ένας χρήστης χρειάζεται να ανακτά πολύ συγκεκριμένα αποτελέσματα, τότε ίσως να χρειάζεται ένα ΣΑΠ υψηλής ακρίβειας παρόλο που μπορεί να μην ανακτά πολλά αποτελέσματα.

Βέβαια, υπάρχουν χρήστες οι οποίοι χρειάζονται ένα ΣΑΠ το οποίο να συνδυάζει και τα δύο μεγέθη. Αυτό μετριέται με χρήση του *μέτρου F*, και συγκεκριμένα του *ισορροπημένου μέτρου F* το οποίο ορίζεται ως εξής:

$F_{\beta=1} = \frac{2PR}{P+R}$ [4]

Με χρήση του τελικού μέτρου μπορεί να γίνει αξιολόγηση ενός ΣΑΠ σε σχέση με ένα άλλο.

Βιβλιογραφία

[1] Cambridge Dictionary, "INFORMATION AGE | English meaning". [Online]. Available: https://dictionary.cambridge.org/dictionary/english/information-age

[2] Oxford English Dictionary, "information age, n. meanings, etymology and more". [Online]. Available: https://www.oed.com/dictionary/information-age_n

[3] Google, "Zeitgeist 2012". [Online]. Available: https://archive.google/zeitgeist/2012/

[4] C. D. Manning, P. Raghavan, and H. Schütze, *Εισαγωγή στην ανάκτηση πληροφοριών. (Ελληνικά) [Introduction to information retrieval.]*. Αθήνα: Εκδόσεις Κλειδάριθμος, 2012, pp. 24–25, 45, 49–53, 56, 65, 145–147, 183, 186–188, 268–270.

[5] Μ. Τριανταφυλλίδης, *ΝΕΟΕΛΛΗΝΙΚΗ ΓΡΑΜΜΑΤΙΚΗ*. Αθήνα: Οργανισμός Εκδόσεως Διδακτικών Βιβλίων (ΟΕΔΒ), p. 69. [Online]. Available: https://www.greek-language.gr/greekLang/files/document/modern_greek/grammatiki.triantafyllidi.pdf

[6] Cambridge Dictionary, "LEMMA | English meaning". [Online]. Available: https://dictionary.cambridge.org/dictionary/english/lemma

[7] R. Pramana, Debora, J. J. Subroto, A. A. S. Gunawan, and Anderies, "Systematic Literature Review of Stemming and Lemmatization Performance for Sentence Similarity". IEEE 7th International Conference on Information Technology and Digital Applications (ICITDA), Yogyakarta, Indonesia, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9971451

[8] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola, "Stemming and lemmatization in the clustering of finnish text documents". Association for Computing Machinery, New York, NY, USA, pp. 625–633. [Online]. Available: https://doi.org/10.1145/1031171.1031285

[9] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and lemmatization: A comparison of retrieval performances". Proceedings of SCEI Seoul Conferences, Seoul, Korea. [Online]. Available: https://eprints.um.edu.my/13423/

[10] D. Khyani, B. Siddhartha, N. Niveditha, and B. Divya, "An interpretation of lemmatization and stemming in natural language processing". Journal of University of Shanghai for Science and Technology, pp. 350–357. [Online]. Available: https://www.researchgate.net/profile/Siddhartha-B-S/publication/348306833_An_Interpretation_of_Lemmatization_and_Stemming_in_Natural_Language_Processing/links/6048467f299bf1e078696a3a/An-Interpretation-of-Lemmatization-and-Stemming-in-Natural-Language-Processing.pdf

[11] M. F. Porter, "An algorithm for suffix stripping". MCB UP Ltd, 1980.

[12] K. H. Rosen, *ΔΙΑΚΡΙΤΑ ΜΑΘΗΜΑΤΙΚΑ και Εφαρμογές τους (Ελληνικά) [Discrete Mathematics and its Applications, 8th Edition]*. Αθήνα: ΕΚΔΟΣΕΙΣ ΤΣΙΟΛΑ, p. 26.

[13] S. Anthony *et al.*, "A Strategy To Estimate Unknown Viral Diversity in Mammals". American Society for Microbiology. [Online]. Available: https://journals.asm.org/doi/abs/10.1128/mbio.00598-13