ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Βελτιωμένη Αναζήτηση σε Άρθρα σχετικά με το Σύνδρομο Sjogren

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Στέφανου-Ευάγγελου Σαμωνά

Επιβλέπουσα: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2019

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Βελτιωμένη Αναζήτηση σε Άρθρα σχετικά με το Σύνδρομο Sjogren

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Στέφανου-Ευάγγελου Σαμωνά

Επιβλέπουσα: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 15η Φεβρουαρίου 2019.

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
.....	.....	.....
Θεοδώρα Βαρβαρίγου	Εμμανουήλ Βαρβαρίγος	Συμεών Παπαβασιλείου
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2019

.....
Στέφανος-Ευάγγελος Σαμωνάς
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η ραγδαία αύξηση της βιοϊατρικής βιβλιογραφίας τα τελευταία χρόνια έχει οδηγήσει σε δυσκολία αξιοποίησής της. Ο όγκος των επιστρεφόμενων δεδομένων σε μία αναζήτηση δυσκολεύει το έργο των ερευνητών, καθώς η κατανόηση της γνώσης και η ανάκτηση των κατάλληλων πληροφοριών απαιτεί περισσότερο κόπο και χρόνο. Συνεπώς η εύρεση και εμφάνιση όσο το δυνατό πιο σχετικών αποτελεσμάτων συμβάλλει στην αξιοποίηση της πληροφορίας.

Στην παρούσα διπλωματική εργασία, αρχικά δημιουργήθηκε μία βάση δεδομένων με τα άρθρα σχετικά με το σύνδρομο Sjogren. Τα άρθρα αυτά αντιστοιχήθηκαν στη συνέχεια με όρους του οργανωμένου λεξιλογίου MeSH. Για βελτιωμένη αναζήτηση χρησιμοποιήθηκε η μηχανή αναζήτησης Elasticsearch στην εύρεση κλινικών όρων στα κείμενα των άρθρων. Παρέχεται η δυνατότητα χρήσης του Elasticsearch στη βάση δεδομένων μέσω μίας διαδικτυακής εφαρμογής για διευκόλυνση της αναζήτησης, καθώς και η παρουσίαση κάποιων στοιχείων της βάσης. Για την αξιολόγηση του συστήματος συγκρίθηκαν τα σύνολα των άρθρων που επιστρέφει το Elasticsearch για κάποιους κλινικούς όρους σχετικούς με το σύνδρομο Sjogren και των άρθρων που περιέχουν τους όρους αυτούς στα MeSH Headings. Τελικά στο συγκεκριμένο πλαίσιο η ακρίβεια και η σχετικότητα των αποτελεσμάτων ήταν υψηλή.

Λέξεις Κλειδιά: βιοϊατρική βιβλιογραφία, σχεσιακή βάση δεδομένων, αναζήτηση ελεύθερου κειμένου, Elasticsearch, διαδικτυακή εφαρμογή.

Abstract

In the last years, the rapid growth of biomedical literature has led to difficult utilization of it. The volume of a search's returning data makes the work of researchers difficult as the understanding of knowledge and the information retrieval demands more effort and time. Therefore finding and diplayings the most possible relevant resutlts contributes to taking advantage of literature.

In this diploma thesis, first of all a database was created which contained articles about Sjogren's syndrom. Next, these articles were corresponded to terms of controlled vocabulary MeSH. The improved search used the search engine Elasticsearch for finding clinic words in articles's texts. A web application provides the capability of using Elasticsearch for facilitation of searching and diplaying some elements of the database. The system evaluated by comparing the set of articles which are returned by Elasticsearch for some clinic terms about Sjogren's Syndrom and the set of articles which contains these terms in MeSH Headings. Finally, in this context, both precision and recall of results were high.

Keywords: biomedical literature, relational database, free text searching, Elasticsearch, web application.

Ευχαριστίες

Καταρχάς θέλω να ευχαριστήσω την καθηγήτρια Θεοδώρα Βαρβαρίγου που μου έδωσε την δυνατότητα να εργαστώ στο τομέα των βάσεων δεδομένων στο εργαστήριό της και να ολοκληρώσω τη διπλωματική μου εργασία.

Ιδιαίτερες ευχαριστίες θέλω να εκφράσω στο Θύμιο Χονδρογιάννη που με καθοδήγησε και για όλο το χρόνο που αφιέρωσε καθ' όλη τη διάρκεια της διπλωματικής εργασίας.

Τέλος, ευχαριστώ θερμά την οικογένειά μου, που με τη στήριξή τους βοήθησαν να ολοκληρώσω τις σπουδές μου, καθώς και όλους τους φίλους που είχα κοντά μου στα φοιτητικά χρόνια.

Πίνακας Περιεχομένων

Πίνακας Σχημάτων

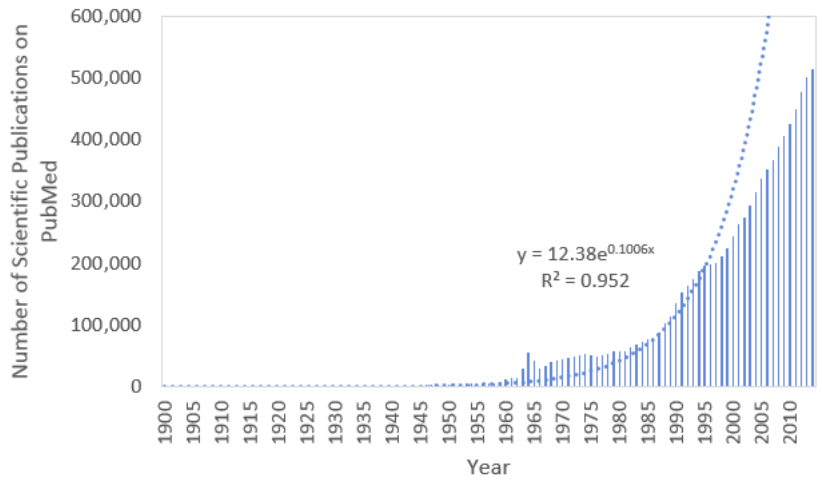
Πίνακας Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο της εργασίας

Τα τελευταία χρόνια παρατηρείται ραγδαία αύξηση της βιοϊατρικής βιβλιογραφίας. Ο όγκος της επιστημονικής έρευνας που παράγεται αυξάνεται κάθε χρόνο. Μία έρευνα στο Transactions of the American Clinical and Climatological Association [1] έδειξε ότι τα δεδομένα διπλασιάζονται με εκθετικό ρυθμό, σχεδόν κάθε λίγους μήνες. Συγκεκριμένα, το 1950 υπολογίζονταν ότι ο χρόνος για το διπλασιασμό τους θα ήταν 50 χρόνια, το 1980 πήγε στα 7 χρόνια, το 2010 στα 3,5, ενώ υπολογίζεται ότι μέχρι το 2020 ο διπλασιασμός θα χρειάζεται περίπου 73 μέρες. Στο PubMed για παράδειγμα, μία από τις μεγαλύτερες βάσεις δεδομένων, ο όγκος των δημοσιεύσεων το 2014 (514,395) έχει τριπλασιαστεί από το 1990 (136,545). Πιο αναλυτικά φαίνεται στο παρακάτω διάγραμμα:



Σχήμα 1.1: Αριθμός επιστημονικών άρθρων στο PubMed (1900-2014).

Για τη σωστή αξιοποίηση αυτών των δεδομένων στον ερευνητικό τομέα, χρειάζονται αποτελεσματικοί τρόποι αξιολόγησής τους. Να μπορεί να ανακτήσει ο ερευνητής-επιστήμονας τις πληροφορίες που απαιτούνται για το έργο του. Ο τεράστιος όγκος των δεδομένων, τόσο του αριθμού των άρθρων που έχουν δημοσιευθεί όσο και του μεγέθους τους, κάνει πιο δύσκολη την αναζήτηση και την κατανόηση της γνώσης, ενώ

υπάρχουν και κίνδυνοι όσον αφορά την αξιοποιστία των πληροφοριών που αυτά περιέχουν. Επιπλέον, το μεγαλύτερο μέρος της πληροφορίας που υπάρχει είναι με τη μορφή ελεύθερου κειμένου και κατά συνέπεια δεν είναι εύκολα επεξεργάσιμο. Στην προσπάθεια του χρήστη να αναζητήσει άρθρα σχετικά με κάποια ορολογία, εμφανίζονται πολλά αποτελέσματα, τα οποία μπορεί να μην είναι τόσο σχετικά με αυτό που αναζητά. Αυτό βέβαια δεν επηρεάζει μόνο τους επιστήμονες, αλλά και το γενικό κοινό που μπορεί να θέλει να ενημερώνεται για την πορεία των ερευνών αλλά να μην είναι ικανό να την ακολουθήσει. Επομένως απαιτούνται εργαλεία τα οποία κάνουν αποτελεσματική αναζήτηση και επιστρέφονται τα καλύτερα δυνατά αποτελέσματα.

1.2 Σκοπός

Ο σκοπός της διπλωματικής αυτής εργασίας είναι η βελτιωμένη αναζήτηση άρθρων σχετικά με το σύνδρομο Sjogren. Επιλέχθηκε ένα πιο ειδικό θέμα για να περιοριστεί ο όγκος των δεδομένων. Στην προσπάθεια να πετύχουμε καλύτερα αποτελέσματα αναζήτησης χρησιμοποιήθηκε η μηχανή αναζήτησης Elasticsearch στην εύρεση κλινικών όρων στα ελεύθερα κείμενα των άρθρων. Η αξιολόγηση του συστήματος έγινε με τη βοήθεια των MeSH Headings που έχουν αποδωθεί στα άρθρα. Επιπλέον, δημιουργήθηκε μία διαδικτυακή εφαρμογή που περιέχει στατιστικά σχετικά με τα άρθρα και στα οποία έχουν πρόσβαση οι χρήστες, όπως και επίσης τους δίνεται η δυνατότητα βελτιωμένης αναζήτησης στα άρθρα σχετικά με το σύνδρομο Sjogren, με τη χρήση του Elasticsearch.

1.3 Οργάνωση κειμένου

Η διπλωματική εργασία αποτελείται από 7 κεφάλαια, συμπεριλαμβανομένου και αυτού που παρουσιάζει το αντικείμενο, το σκοπό και την οργάνωση της εργασίας. Στο δεύτερο κεφάλαιο αναλύονται τα σχετικά θέματα και εργαλεία πάνω στα οποία θα στηριχτεί η εργασία, καθώς και εναλλακτικές επιλογές που υπάρχουν, όπως το GoPubMed και το UMLS. Στο κεφάλαιο 3 παρουσιάζεται η προσέγγιση που ακολουθήθηκε και η βάση που αναπτύχθηκε για την αποθήκευση τόσο των άρθρων όσο και των MeSH όρων. Στο τέταρτο κεφάλαιο βρίσκεται η υλοποίηση των επιμέρους συστημάτων καθώς και λεπτομέρειες ως προς τον τρόπο λειτουργίας τους. Στη συνέχεια στο επόμενο κεφάλαιο παρουσιάζονται τα αποτέλεσματα του Elasticsearch και η αξιολόγηση του συστήματος. Στο έκτο κεφάλαιο υπάρχουν πιθανές μελλοντικές έρευνες και βελτιώσεις πάνω στην διπλωματική εργασία, ενώ στο έβδομο και τελευταίο κεφάλαιο βρίσκεται η σύνοψη της εργασίας.

Κεφάλαιο 2

Σχετική Εργασία

2.1 PubMed

Το PubMed [3] είναι μία μηχανή αναζήτησης που παρέχει πρόσβαση στο MEDLINE. Το MEDLINE είναι βιβλιογραφική βάση δεδομένων επιστημών ζωής και βιοϊατρικής πληροφορίας. Μέχρι τις 6 Δεκεμβρίου 2018 είχε περισσότερες από 29.1 εκατομμύρια εγγραφές. Αναπτύχθηκε και διατηρείται από την National Library of Medicine (NLM) της Αμερικής ως τμήμα του συστήματος ανάκτησης πληροφοριών Entrez. Το PubMed είναι δωρεάν και βοηθά στην ερευνητική δραστηριότητα πολλών επιστημόνων. Πρόκειται ουσιαστικά για ένα Web-based σύστημα ανάκτησης πληροφοριών, Δεν περιέχει πλήρη άρθρα, όμως αρκετές φορές εκτός από περιλήψεις περιέχει και links για το πλήρες άρθρο. Καθημερινά προστίθεται νέο υλικό.

Ένα άρθρο στο PubMed αποτελείται από 8 βασικά πεδία. Πρώτα από όλα ο τίτλος του και το περιοδικό στο οποίο έγινε η δημοσίευση. Έπειτα ακολουθεί η γλώσσα συγγραφής, αν δεν είναι αγγλικά, μαζί με το abstract κείμενο (περίληψη). Να σημειωθεί ότι υποστηρίζονται όλες οι γλώσσες αλλά οι περισσότερες δημοσιεύσεις είναι στα αγγλικά, σε ποσοστό περίπου 87%. Στη συνέχεια υπάρχουν τα ονομάτα των συγγραφέων μαζί με τους οργανισμούς στους οποίους ανήκουν και κάποιες γενικές πληροφορίες που αφορούν το είδος δημοσίευσης. Τα περισσότερα άρθρα που εμφανίζονται είναι ήδη δημοσιευμένα. Στο τέλος υπάρχουν όροι του ελεγχόμενου λεξιλογίου MeSH που αντιστοιχούν στο άρθρο. Οι όροι αυτοί έχουν αποδωθεί στο άρθρο από ειδικούς indexers.

Στην αρχική σελίδα του PubMed μπορεί να γίνει η αναζήτηση με πλήθος επιλογών. Πέρα από την επιλογή βάσης και πληκτρολόγηση κειμένου-όρων, μπορούν να κατασκευαστούν πολύπλοκα ερωτήματα προσαρμοσμένα στις εκάστοτε ανάγκες με τη βοήθεια οδηγών και εργαλείων. Υπάρχουν φίλτρα, που περιορίζουν το εύρος των αποτελεσμάτων. Επίσης, το MEDLINE χρησιμοποιεί το MeSH για indexing και το PubMed κάνει αυτόματη αντιστοίχιση των όρων αναζήτησης σε MeSH όρους. Συνεπώς, έτσι μπορεί η αναζήτηση να γίνει πιο εξειδικευμένη δηλώνοντας με tag το MeSH. Επιπλέον, είναι δυνατόν να προσδιορίσουμε το πεδίο αναζήτησης. Για παράδειγμα να πούμε ότι αναζητούμε κάποιον συγγραφέα, οπότε η αναζήτηση να γίνει μόνο στους συγγραφείς. Με την επιστροφή των αποτελεσμάτων λοιπόν, υπάρχει πλήθος επιλογών για τη συνέχεια. Από την μορφοποίηση και ταξινόμησή τους μέχρι την πλοήγηση στα άρθρα και αξιοποίησή τους. Φυσικά πέρα από τη διαδικτυακή πλατφόρμα του PubMed, είναι δυνατή και η χρήση του με API, με τη βοήθεια του εργαλείου E-utilities.

2.2 MeSH Headings

Το MeSH [6] είναι το ελεγχόμενο λεξιλόγιο της U.S. National Library of Medicine (thesaurus) και είναι ακρωνύμιο για το Medical Subject Headings. Παρέχει ομοιομορφία και συνέπεια στα ευρετήρια και καταλόγους της βιοϊατρικής βιβλιογραφίας. Ανανεώνεται συχνά και έχει ιεραρχική δομή που ονομάζεται MeSH Tree Structures και υπάρχει η δυνατότητα πλοήγησης. Χρησιμοποιείται στο MEDLINE/PubMed για να περιγράφει το περιεχόμενο των άρθρων. Αυτό βοηθά στην αναζήτηση θεμάτων από τους ερευνητές. Οι περισσότεροι όροι που εισάγονται από ερευνητές αντιστοιχίζονται αυτόματα σε MeSH όρους. Πολλά συστήματα παρέχουν πρόσβαση στο MeSH ενώ το λεξιλόγιο είναι διαθέσιμο online και για κατέβασμα σε πολλές μορφές (μεταξύ των οποίων και XML) για την κάλυψη των προσωπικών αναγκών του εκάστοτε χρήστη.

Υπάρχουν τρεις βασικοί τύποι εγγραφών MeSH Headings:

1. Descriptors : Αποτελούν τη βασική μονάδα indexing και έχουν κεντρικό ρόλο. Οργανώνονται σε ιεραρχική δομή δέντρου που επιτρέπει τη πλοήγηση από το γενικό στο ειδικό. Χωρίζονται σε 4 κλάσεις : Main Headings, Publication Characteristics (Publication Types), Check Tags και Geographics.

2. Qualifiers: Χρησιμοποιούνται συμπληρωματικά με τους Descriptors και περιγράφουν μία πιο συγκεκριμένη οπτική ενός θέματος. Έχουν επίσης ιεραρχική δομή δέντρου.

3. Supplementary Concept Records (SCRs): Αφορούν το indexing χημικών και φαρμάκων. Δεν έχουν ιεραρχική δομή δέντρου και χωρίζονται σε 4 κλάσεις: Chemicals, Protocols, Diseases και Organisms.

Η κύρια μονάδα του MeSH, οι Descriptors, έχουν ιεραρχική δομή δέντρου και οργανώνονται σε 16 βασικές κατηγορίες. Αυτές είναι: Anatomy, Organisms, Diseases, Chemicals and Drugs, Analytical, Diagnostic and Therapeutic Techniques and Equipment, Psychiatry and Psychology, Phenomena and Processes, Disciplines and Occupations, Anthropology, Education, Sociology and Social Phenomena, Technology, Industry, Agriculture, Humanities, Information Science, Name Groups, Health Care, Publication Characteristics, Geographicals.

Κάθε κατηγορία χωρίζεται σε υποκατηγορίες. Κάθε μία υποκατηγορία οργανώνεται επίσης ιεραρχικά από το πιο γενικό στο πιο ειδικό μέχρι 13 επίπεδα. Η οργάνωση αυτή δεν είναι απόλυτη αλλά βοηθά στην πλοήγηση του χρήστη που αναζητά βιβλιογραφία. Επίσης, κάποιος descriptor μπορεί να εμφανίζεται πάνω από μία φορά στο δέντρο. Κάθε θέση στο δέντρο χαρακτηρίζεται και από ένα αριθμό. Επιπλέον με κάθε Descriptor υπάρχει λίστα με τα επιτρεπτά Qualifiers που μπορούν να συνδυαστούν μαζί του.

Το βασικό πλεονέκτημα του MeSH είναι ότι οργανώνει όλη την ιατρική γνώση και πληροφορία κάτω από το ίδιο πλαίσιο. Για να επιτευχθεί αυτό χρησιμοποείται η έννοια του concept, ότι κάποιοι όροι δηλαδή ανήκουν σε ομάδες. Κάθε όρος-εγγραφή έχει συγκεκριμένη σημασία. Παρόλα αυτά είναι πολλές οι περιπτώσεις που ένας όρος (term) είναι συνώνυμος με κάποιον άλλο. Δηλαδή δύο όροι μπορεί να γράφονται διαφορετικά αλλά να έχουν το ίδιο νόημα. Ένα τέτοιο παράδειγμα είναι οι όροι “Cardiac Arrest” και “Heart Arrest”. Με τον όρο concept λοιπόν αναφερόμαστε στο κοινό νόημα που εκφράζει κάποιες λέξεις-όρους. Συνεπώς αν δύο όροι ανήκουν στο ίδιο concept λέμε ότι είναι συνώνυμοι.

Η δομή ενός Descriptor στηρίζεται σε αυτή την ιδέα του concept. Κάθε Descriptor λοιπόν αποτελείται από ένα ή περισσότερα concepts αν είναι γενικότερος. Τα concepts αυτά ουσιαστικά περιγράφουν το Descriptor και ανάλογα κάθε concept αποτελείται από ένα ή περισσότερα terms. Φυσικά κάποιο concept ταιριάζει περισσότερο στο νόημα του Descriptor. Αυτό χαρακτηρίζεται ως Preferred ενώ τα υπόλοιπα ως Narrower. Κατ'επέκταση κάθε concept έχει Preferred term. Στο παρακάτω παράδειγμα ο Descriptor Cardiomegaly αποτελείται από 2 concepts και 5 terms. Τα δύο αυτά concepts περιγράφουν τον ίδιο Descriptor παρόλα αυτά βλέπουμε ότι προτιμότερο είναι το πρώτο. Αυτό σημαίνει ότι το συγκεκριμένο descriptor το χαρακτηρίζει καλύτερα αυτό το concept. Ανάλογα οι όροι κάθε concepts είναι συνώνυμοι και αναφέρονται στο ίδιο concept. Όπως και προηγουμένως όμως, ένας όρος ταιριάζει καλύτερα με το κάθε concept. Σχηματικά έχουμε το εξής:

Cardiomegaly [Descriptor]	
Cardiomegaly	[Concept, Preferred]
Cardiomegaly	[Term, Preferred]
Enlarged Heart	[Term]
Cardiac Hypertrophy	Concept, Narrower
Cardiac Hypertrophy	Term, Preferred
Heart Hypertrophy	Term

Παρότι οι όροι του ίδιου concept είναι συνώνυμοι, όροι διαφορετικών concepts ακόμα και αν ανήκουν στην ίδια εγγραφή δεν είναι απαραίτητα.

Όσον αφορά την αποδόση MeSH Headings στα άρθρα, η διαδικασία του indexing είναι ιδιαίτερα σημαντική εφόσον εκεί θα στηριχτεί η αναζήτηση στη συνέχεια. Αρχικά γίνεται έλεγχος του άρθρου και αποφασίζεται το θέμα του. Έπειτα περιγράφεται το περιεχόμενό του στηριζόμενο σε ελεγχόμενο λεξιλόγιο. Πιο συγκεκριμένα οι indexers του NLM's MEDLINE διαβάζουν αναλυτικά το τίτλο, το κείμενο, το συμπέρασμα, τις αναφορές και τις λέξεις κλειδιά του συγγραφέα. Στη συνέχεια με τη βοήθεια του online λεξιλογίου MeSH Browser βρίσκουν τους όρους που περιγράφουν με τον καλύτερο δυνατό τρόπο τη βασική ιδέα. Ένα θέμα μπορεί να είναι πολύπλοκο οπότε τότε θα χρησιμοποιηθούν πολλαπλοί Descriptors και Qualifiers.

2.3 Αποτίμηση Ερωτημάτων

Στο κομμάτι της χρησιμότητας του MeSH λεξιλογίου, αυτό αξιοποιείται σε πολύ μεγάλο βαθμό στην αναζήτηση που παρέχει το PubMed. Είναι δυνατή η αναζήτηση ακόμα και για πολύπλοκα θέματα με τη χρήση πολλαπλών όρων. Το πιο σημαντικό πλεονέκτημα όμως είναι η αυτόματη ανάπτυξη ερωτήματος. Η ανάπτυξη ενός query, που ορίζεται ως η ανασύνταξή του, στοχεύει στη βελτίωση των αποτελεσμάτων που επιστρέφονται. Αυτό επιτυγχάνεται με την προσθήκη επιπλέον ορολογίας. Παρότι στο PubMed υπάρχει η δυνατότητα του advanced search προσθέτωντας πεδία χειροκίνητα, πάνω από το 90% των ερωτημάτων γίνεται χωρίς πεδία.

Στο PubMed γίνεται αυτόματα η ανάπτυξη των ερωτημάτων (ATM) με τη χρήση του MeSH [7]. Όποιος από τους όρους δεν έχει tag, γίνεται αντιστοίχιση του όρου με κάποιον από τους παρακάτω πίνακες με την εξής σειρά : MeSH πίνακα (όρος σε MeSH έννοια), πίνακα περιοδικών (όρος σε όνομα περιοδικού) και ευρετήριο συγγραφέων

(όρος σε όνομα συγγραφέα). Αν βρεθεί λοιπόν αντιστοίχιση τότε ο νέος όρος προστίθεται στο ερώτημα. Έτσι δεν θα ανακτηθούν μόνο αρχεία που περιέχουν την αρχική λέξη στον τίτλο ή στο κείμενο. Συγκεκριμένα λόγω της ιεραρχημένης δομής του MeSH θα εμφανίζονται αρχεία που περιέχουν ακόμα πιο συγκεκριμένους όρους από τον αρχικό.

Η ανάπτυξη του ερωτήματος ξεκινά με την αφαίρεση των stop words. Στην συνέχεια απαριθμούνται όλοι οι πιθανοί συνδυασμοί λέξεων και καθένας αναπτύσσεται από το ATM. Τα πιο συχνά tag που εμφανίζονται είναι τα [MeSH Terms], [Text Words] και [All Fields]. Το πρώτο κάνει match τον όρο MeSH καθώς και αυτούς που είναι από κάτω του στην ιεραρχία, το δεύτερο όλες τις λέξεις στον τίτλο, το κείμενο και τον MeSH όρο χωρίς την ιεραρχία και το τελευταίο αν δεν βρεθεί κανένα άλλο match.

Για την αποτίμηση της ανάπτυξης των ερωτημάτων υπάρχουν διάφορα συστήματα μέτρησης. Το recall που δείχνει το ποσοστό των άρθρων απο τα συνολικά που είναι σχετικά με το ερώτημα, το precision που δείχνει το ποσοστό των άρθρων από τα ανεκτημένα που είναι σχετικά και το F-measure που προκύπτει από τον τύπο 2 x (precision x recall) / (precision + recall). Επίσης υπάρχει το mean ranking precision (MRP) που αναλύει μόνο τα καλύτερα αποτελέσματα (τα 10-20 πρώτα) και το TF-IDF.

Έρευνες [7] δείχνουν ότι η χρήση του MeSH στην ανάπτυξη των ερωτημάτων στο PubMed βελτιώνει την απόδοση της ανάκτησης αλλά δεν επηρεάζει τους χρήστες σε ρεαλιστικές συνθήκες. Αυτό συμβαίνει διότι επιστρέφονται περισσότερα αποτελέσματα και έτσι κάποιες φορές μπορεί να λειτουργήσει επιζήμια στο σύστημα. Συνεπώς μία πραγματική αλλαγή θα ήταν στη σειρά εμφάνισης των αρχείων κι όχι με αντίστροφη χρονολογική σειρά που είναι τώρα στο PubMed.

Ένα από τα πιο σημαντικά εργαλεία, που ακολουθεί μία διαφορετική λογική από αυτή της Boolean του PubMed, είναι το GoPubMed που επιτρέπει την αναζήτηση με χρήση του GO(Gene Ontology) [5]. Πρόκειται για ένα δομημένο, ελεγχόμενο και ιεραρχιμένο λεξιλόγιο. Στηρίζεται στην κατηγοριοποίηση των abstract. Οι όροι χωρίζονται σε τρεις υποκατηγορίες : cellular location, molecular function και biological process. Η κατηγοριοποίηση συμβάλει στην ευκολότερη πλοήγηση στα άρθρα. Ο server που μας επιτρέπει να χρησιμοποιούμε το PubMed συνδιαστικά με το GO είναι ο GoPubMed.

Η εύρεση ενός όρου επακριβώς δεν είναι τόσο εύκολο. Για αυτό πρέπει να υπάρχει μια συσχέτιση μεταξύ ενός όρου και άλλων που έχουν το ίδιο νόημα. Την δουλειά αυτή την αναλαμβάνει ένας αλγόριθμος που συγκρίνει λέξεις και από το abstract και GO terms. Αφού λοιπόν εξάγει τους GO όρους από τα abstract που έχουν επιστρέψει, παρουσιάζει τα άρθρα ταξινομημένα με βάση το ποσοστό ακριβείας του επιλεγμένου όρου.

Το GoPubMeb μπορεί να φανεί και σε άλλες περιπτώσεις χρήσιμο. Για παράδειγμα αναζητώντας το όνομα και το affiliation ενός συγγραφέα, οι GO όροι που επιστρέφοναι φανερώνουν τα κύρια θέματα με τα οποία ασχολείται αυτός. Επίσης, εμφανίζονται οι σχετικοί όροι με εκείνον της αναζήτησης, καθώς και ποιοι είναι πιο συχνά χρησιμοποιούμενοι.

Άλλα εργαλεία παρόμοια με το GoPubMed είναι τα Textpresso, XplorMed και Vivisimo. Το πρώτο από αυτά χρησιμοποιεί λιγότερες κατηγορίες, έχει περιορισμένες δυνατότητες πλοήγησης και δεν έχει μεγάλο εύρος συνώνυμων όρων. Το XplorMed αντιστοιχεί τα αποτελέσματα σε κάποιες κατηγορίες MeSH και είναι πολύ γενικό, ενώ το Vivisimo μοιάζει περισσότερο στο GoPubMed αλλά γυρνάει μόνο προκαθορισμένα

αποτελέσματα με τους ταιριασμένους GO όρους.

Συμπερασματικά, το GoPubMed παρέχει πολλά πλεονεκτήματα. Καταρχάς προσφέρει μια γενικότερη εικόνα των abstract μέσω της κατηγοριοποίησης. Έτσι γίνεται ευκολότερη και η πλοήγηση. Επιπλέον, η αναζήτηση γίνει πιο αποδοτική μιας και επιστρέφονται αποτελέσματα με βάση την ακρίβεια των όρων, καθώς και ο χρήστης βλέπει την δομή και ιεραρχία και κατανοεί καλύτερα τους όρους.

2.4 Elasticsearch

Προκειμένου να βρούμε κλινικούς όρους στα κείμενα των άρθρων χρειάζεται ένας αποδοτικός τρόπος αναζήτησης μιας και μιλάμε για αναζήτηση ελεύθερου κειμένου. Η αναζήτηση στο κείμενο με character matching είναι ένας απλός αλλά χρονοβόρος τρόπος. Χρειάζεται να λαμβάνονται υπόψιν και άλλοι παράγοντες για την εύρεση των καλύτερων δυνατών αποτελεσμάτων. Για παράδειγμα δύο διαφορετικά γραμμένες λέξεις, όπως το "indexer" και το "indexing", αναφέρονται στην ίδια λέξη και συνεπώς πρέπει να αναγνωρίζονται ως ίδιες. Το character matching για τη λέξη "index" δεν θα έκανε match με καμία από τις δύο. Μια πιθανή λύση, που λαμβάνει υπόψιν πολλούς παράγοντες και μπορεί να προσσαρμοστεί σε πλήθος καταστάσεων, είναι το Elastic-Search [8]. Πρόκειται για μία δημοφιλή μηχανή αναζήτησης ελεύθερου κειμένου σε πραγματικό χρόνο (real time) και ανάλυσης δεδομένων. Στηρίζεται στη βιβλιοθήκη Lucene [18], που επιτρέπει την αναζήτηση και ανάκτηση πληροφοριών από κείμενο, είναι ανοιχτού κώδικα και υποστηρίζει πολλές γλώσσες προγραμματισμού, μεταξύ των οποίων βρίσκεται και η Java, στην οποία μάλιστα είναι και γραμμένο. Επιπλέον υπάρχουν πολλά εργαλεία που μπορούν να αξιοποιηθούν παράλληλα, όπως το οπτικό εργαλείο Kibana που κάνει πιο εύκολη τη πλοήγηση και παρακολούθηση.

Το Elasticsearch είναι ικανό να απαντήσει σε ένα αίτημα πολύ γρήγορα (σχεδόν real-time) διότι αντί να ψάχνει απευθείας στο κείμενο, ψάχνει στο ευρετήριο (index). Ένα παράδειγμα ανάλογο στην καθημερινή ζωή είναι όταν ψάχνουμε κάτι σε ένα βιβλίο κοιτάμε στο πίσω μέρος το ευρετήριο για να δούμε που βρίσκεται κάθε λέξη.

Πρώτα όμως πρέπει να έχουν σταλεί τα δεδομένα στο Elasticsearch. Στη δικιά μας περίπτωση τα δεδομένα είναι τα abstract κείμενα. Αυτό γίνεται με τη χρήση API ή ingestion tools όπως το Logstash [19], που φροντίζει να στείλει τα κείμενα στο Elasticsearch. Τα δεδομένα στέλνονται με τη μορφή JSON αρχείων, αποθηκεύονται και προστίθεται αντίστοιχη αναφορά στο ευρετήριο, που γίνονται χωρίς τη παρέμβαση του χρήστη και με διαφανή τρόπο. Το ευρετήριο που χρησιμοποιείται είναι ανεστραμένο (inverted index) επειδή αντί το κέντρο της δομής δεδομένων να είναι η σελίδα (σελίδα σε λέξη), είναι η ίδια η λέξη (λέξη σε σελίδα). Συγκεκριμένα για κάθε λέξη υπάρχει αντιστοίχιση με τα κείμενα στα οποία βρίσκεται. Εφόσον οι όροι είναι ταξινομημένοι, εντοπίζονται εύκολα και κατά συνέπεια τα κείμενα στα οποία βρίσκονται. Για τη δημιουργία και τη διαχείριση αυτού του ευρετηρίου χρησιμοποιείται το Apache Lucene, όπως αναφέρθηκε και προηγουμένως. Η χρήση του Lucene βελτιστοποιεί την απόδοση και παρότι είναι πολύπλοκη, όλα γίνονται χωρίς να παρεμβάλεται ή να βλέπει κάτι ο χρήστης.

Το ευρετήριο αποτελείται από ένα ή περισσότερα αρχεία που το καθένα στη συνέχεια περιέχει ένα ή περισσότερα πεδία. Ένα Elasticsearch index αποτελείται ουσιαστικά από πολλά ξεχωριστά indexes που έχουν δημιουργηθεί από το Lucene. Στο inverted index οφείλεται η γρήγορη αναζήτηση λέξεων. Στο ευρετήριο οι λέξεις-όροι είναι ταξινομημένοι αλβαβητικά και είναι εύκολο να βρεθούν και κατά συνέπεια σε ποια

αρχεία υπάρχουν. Ανάλογα λειτουργεί και όταν γίνεται αναζήτηση πολλών λέξεων ταυτόχρονα. Επειδή όμως υπάρχουν πολλές περιπτώσεις που η καθεμία έχει διαφορετικές ανάγκες, υπάρχει η δυνατότητα να δημιουργηθούν ευρετήριο προσαρμοσμένα στις απαιτήσεις αυτές. Συνεπώς είναι ιδιαίτερα σημαντική η ανάλυση του κειμένου, προκειμένου να δημιουργήσουμε ένα ευρετήριο το οποίο θα βελτιστοποιεί την αναζήτηση.

Η αναζήτηση γίνεται επίσης με αυτοματοποιημένο τρόπο. Στέλνεται ένα query μέσα σε ένα HTTP GET με δομή JSON. Κατά τη διαδικασία αυτή συμβουλεύεται το ευρετήριο για ποια δεδομένα θα ανακτηθούν. Το Elasticsearch παρέχει πλήθος επιλογών και κατ' επέκταση ένα ερώτημα (query) μπορεί να γίνει περίπλοκο ανάλογα με τις ανάγκες του χρήστη. Κάθε εφαρμογή του πραγματικού κόσμου μπορεί να απαιτεί αναζήτηση σε συγκεκριμένα πεδία και συνθήκες, με διαφορετική βαρύτητα σε κάποιες λέξεις ή ακόμα και πιο πρόσφατα αρχεία χρονικά. Παρόλα αυτά υπάρχει συγκεκριμένος τρόπος γραφής ενός query σε JSON format. Συγκεκριμένα το Elasticsearch παρέχει ένα full Query DSL (Domain Specific Language) βασισμένο σε JSON και καθορίζει τη μορφή των queries. Επιπλέον, παρέχεται η δυνατότητα να γίνει search request χρησιμοποιώντας URI με κάποιους παραμέτρους. Το εύρος των παραμέτρων είναι περιορισμένο συγκριτικά με το JSON, όμως γίνεται ταχύτερα και πιο εύκολα. Κλείνοντας να τονιστεί ότι το searching είναι τόσο γρήγορο και αποδοτικό χάρη στην αποκανονικοποίηση των δεδομένων και του inverted index.

Η δημοφιλία του Elasticsearch οφείλεται στο πλήθος των προτερημάτων του καθώς και το εύρος αναγκών που καλύπτει. Οποιοσδήποτε χρήστης ή οργανισμός που έχει πολύ μεγάλο όγκο δεδομένων μπορεί να το αξιοποιήσει. Είτε σε ένα website για εύρεση προϊόντων είτε για ανάλυση και εξαγωγή στατιστικών. Επίσης, ο χρήστης δεν ασχολείται με το πως λειτουργεί το Elasticsearch (π.χ. πως καλεί το Lucene) παρά μόνο τα παρακολουθεί. Είναι κατανεμημένο, δηλαδή μπορεί να διαμοιράσει τον όγκο πληροφορίας σε ξεχωριστούς server και να τρέχει ταυτόχρονα σε αυτούς, και κλιμακώσιμο τόσο οριζόντια όσο και κάθετα, μιας και ρυθμίζεται αυτόματα από το σύστημα ποιος κόμβος αναλαμβάνει τι. Παράλληλα, προσφέρει υψηλή διαθεσιμότητα και απόδοση. Αν ένας κόμβος αποκοπεί από το cluster (το σύμπλεγμα των κόμβων), τα δεδομένα μετατίθενται αυτόματα στους υπόλοιπους. Συνεπώς προσφέρει ανθεκτικότητα και αξιοπιστία. Επίσης λειτουργεί βάση της αρχιτεκτονικής REST αξιοποιώντας όλα τα προτερήματά της. Τέλος τα αιτήματα (requests) και απαντήσεις (responses) για την ανάκτηση δεδομένων γίνονται πάνω σε HTTP web διεπαφή και χρησιμοποιούνται δομές JSON, γεγονός που οδηγεί στην ανεξαρτήσια από τους τύπους των δεδομένων.

2.5 Λοιπά εργαλέια/Τεχνολογίες

Η αύξηση αυτή της βιβλιογραφίας με εκθετικό ρυθμό δημιουργεί την ανάγκη για βελτιωμένα εργαλεία διαχείρισης και αναζήτησης. Όλο και περισσότερα αρχεία εμφανίζονται στους χρήστες και αυτό κάνει πιο δύσκολη την αναζήτηση και την κατανόηση τους. Με δεδομένη τη λειτουργία του PubMed, σε συνδυασμό με το γεγονός ότι τα δεδομένα του MEDLINE είναι δωρεάν και διαθέσιμα για εξωτερικούς παράγοντες, έχουν αναπτυχθεί πολλά και διάφορα εναλλακτικά Web εργαλεία, τα οποία λειτουργούν συμπληρωματικά με το PubMed, για την αποδοτική αναζήτηση και ανάκτηση αρχείων. Τα εργαλεία αυτά εκμεταλλεύονται την ραγδαία ανάπτυξη της τεχνολογίας και συμβάλλουν σημαντικά στις επιστημονικές ανακαλύψεις τόσο ερευνητικού σκοπού όσο και ιατρικής περίθαλψης κατά συνέπεια. Όσο περνάει ο καιρός τα εργαλεία αυτά όλο και αυξάνονται σε αριθμό αλλά και βελτιώνονται.

Θα μπορούσαν να κατηγοριοποιηθούν σε 4 βασικές κατηγορίες με βάση τα χαρακτηριστικά τους [9]. Η πρώτη αφορά την ταξινόμηση των αποτελεσμάτων αναζήτησης, η δεύτερη τον σχηματισμό ομάδων με βάση το θέμα, η επόμενη την εξαγωγή εννοιών και σχέσεων και τέλος την βελτίωση της διεπαφάνειας αναζήτησης. Όλα τα εργαλεία αναπτύσσονται διαρκώς και μάλιστα η ανάπτυξη αυτή συμπίπτει με την αντίστοιχη στους τομείς του text-mining και γενικότερα της Web τεχνολογίας. Κάποια δημιουργήθηκαν για ακαδημαϊκούς σκοπούς ενώ άλλα από ιδιώτες. Μερικά σημεία που διαφοροποιούνται είναι: το αν είναι ανοιχτού κώδικα ή όχι, η μορφή εμφάνισης των αποτελεσμάτων, αν υπάρχουν links για παραπομπές, η δυνατότητα εξαγωγής των αποτελεσμάτων.

Στην πρώτη κατηγορία (ranking search results) βλέπουμε εργαλεία με διάφορες στρατηγικές ταξινόμησης των αποτελεσμάτων. Από machine learning και user feedback (RefMed) μέχρι σύγκριση pre-indexed κειμένων (eTBLAST). Κάποια από αυτά είναι τα: Quertle, MedlineRanker, MiSearch, Hikia, Semantic MEDLINE, MScanner, PubFocus, Twease.

Η δεύτερη (clustering results into topics) είναι σχετικά με την κατηγοριοποίηση των αποτελεσμάτων με σκοπό την ευκολότερη πλοήγησή και διαχείρισή τους και την μείωση της υπερπληροφόρησης. Εδώ τα εργαλεία στηρίζονται σε προκαθορισμένες κατηγορίες, ιεραρχικές ή μη δομές και άλλους διαφορετικούς τρόπους ομαδοποίησης των δεδομένων. Κάποια εργαλεία είναι τα: Anne O'Tate, McSyBi, GOPubMed, ClusterMed, XplorMed.

Στην επόμενη κατηγορία (results with semantics and visualization) τα αποτελέσματα αναλύονται και παρουσιάζεται συγκεντρωμένη η γνώση της έννοιας με βάση τεχνικές εξαγωγής πληροφορίας. Μπορούν να υπάρχουν διαφορές τόσο στον τρόπο εξαγωγής της πληροφορίας όσο και στον τρόπο παρουσίασης αυτής. Τα MedEvi, EBIMED, CiteXplore, MEDIE και PubNet είναι κάποια από τα εργαλεία αυτής της κατηγορίας.

Στην τέταρτη και τελευταία κατηγορία (search interface and retrieval experience) υπάρχουν εργαλεία που στοχεύουν στη βελτίωση της αποτελεσματικότητας της αναζήτησης. Είτε με auto-complete είτε με διαφορετικό γραφικό περιβάλλον (π.χ. slide bars), ακόμα και διαφορετικές γλώσσες πέραν της αγγλικής. Κάποια είναι: iPubMed, PubGet, Babelmesh, HubMed, askMEDLINE, SLIM, PICO, PubCrawler.

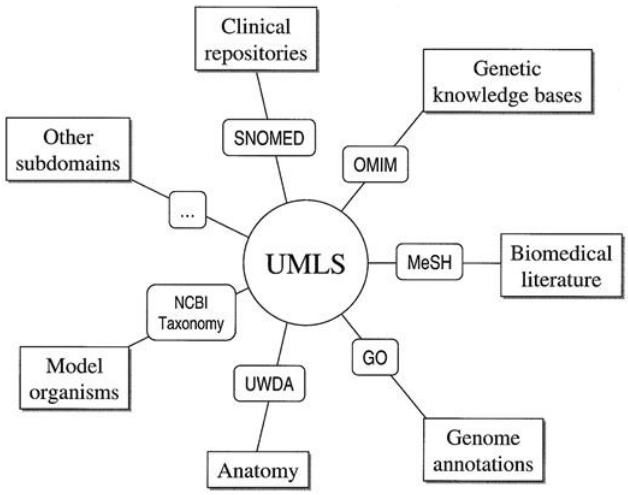
Κάποια εργαλεία που δεν εμπίπτουν σε καμία από τις προηγούμενες κατηγορίες είναι τα εξής: PubMed Assistant, AliBaba, PubMed-EX, iHop, Chilibot, PolySearch, Semedico, PubFinder, ReleMed, MedMiner, PubClust.

Όλα αυτά τα εργαλεία έχουν τον ίδιο τελικό στόχο με το PubMed. Κάνουν πλήρη ή μερική χρήση των δεδομένων του και λειτουργούν συμπληρωματικά. Το PubMed αλλάζει και βελτιώνεται παράλληλα με αυτά. Παρέχει ταυτόχρονη πρόσβαση σε όλες τις διαφορετικές βάσεις, links, sensors, έχει προσθέσει έννοιες στα κείμενα που κάνουν ευκολότερη την κατανόηση της γνώσης, εμφανίζονται πιο σχετικά άρθρα, ενώ υπάρχει και η δυνατότητα query suggestion και η προσαρμογή στις ειδικές ανάγκες του κάθε χρήστη. Τέλος να αναφερθεί ότι υπάρχει ιστοσελίδα [16] που περιέχει όλα τα υπάρχοντα συστήματα μαζί με τα χαρακτηριστικά τους και links για το πραγματική πλατφόρμα τους. Το site ενημερώνεται συνεχώς με τα νέα συστήματα.

Εκτός όμως από τα παραπάνω εργαλεία που σχετίζονται άμεσα με το PubMed και στα οποία η αναζήτηση γίνεται με διαφανή τρόπο για το χρήστη, υπάρχουν πολλά άλλα εργαλεία για το ξεχωριστό τμήμα που αφορά την αναζήτηση αποκλειστικά σε ελεύθερο κείμενο. Στην δική μας περίπτωση επιλέχθηκε το Elasticsearch. Τα εργαλεία αυτά μπορεί να είναι είτε online είτε API είτε άλλα διαθέσιμα για κατέβασμα, τα οποία εξάγουν MeSH από κείμενο. Γίνεται η εισαγωγή του κειμένου και επιστρέφονται οι

αντίστοιχοι βιοϊατρικοί όροι. Τα περισσότερα όμως χρειάζονται άδεια και για αυτό επιλέχτηκε μία διαφορετική προσέγγιση του θέματος με το Elasticsearch. Παρακάτω θα αναλυθούν τα βασικότερα από αυτά τα εργαλεία.

Καταρχάς όλα τα εργαλεία στηρίζονται στο Unified Medical Language System (UMLS) [10]. Πρόκειται για ένα σύνολο αρχείων και λογισμικού που συνδιάζει βιοϊατρικά λεξιλόγια. Ο σκοπός του είναι να ξεπεράσει δύο βασικά προβλήματα : την χρήση διαφορετικών ονομάτων για το ίδιο έννοια και την απουσία συγκεκριμένης δομής στην ορολογία, που έχουν ως συνέπεια οι πόροι να μην μπορούν να αξιοποιηθούν καθολικά. Το UMLS χρησιμοποιείται στις περισσότερες εφαρμογές και πλέον περιλαμβάνει ένα μεγάλο εύρος ορολογιών. Το NCBI taxonomy για ταυτοποίηση οργανισμών, το Gene Ontology για αντιστοίχιση γονιδίων σε οργανισμούς, το MeSH για βιβλιογραφία, το SNOMED για κλινικά θέματα, καθώς και άλλα για διάφορες υποκατηγορίες. Στο παρακάτω σχήμα φαίνονται οι διάφοροι υποτομείς που ενσωματώνονται στο UMLS:



Σχήμα 2.1: Οι διάφοροι υποτομείς που ενσωματώνονται στο UMLS [10].

Η οργάνωση του UMLS στηρίζεται στην ιδέα των εννοιών. Συγκεκριμένα διάφοροι όροι που είναι συνώνυμοι δημιουργούν μία ομάδα, που πρόκειται για την έννοια. Οι διάφορες έννοιες συνδέονται μεταξύ τους μέσω σχέσεων που μπορεί να είναι είτε εσωτερικές είτε εξωτερικές. Οι εσωτερικές σχέσεις μπορεί να παράγονται ή να είναι κληρονομημένες στα πλαίσια της ιεραρχικής δομής ή ακόμα και κάποια παραπομπή. Οι εξωτερικές μπορεί να είναι κάποια αναφορά σε άλλη βάση όπως π.χ. το Gene Bank. Η οργάνωση σε κατηγορίες λοιπόν βοηθά τους χρήστες να βρίσκουν συνώνυμα όρων της ίδιας έννοιας, τις σχέσεις μεταξύ των εννοιών, καθώς και τις έννοιες μιας κατηγορίας.

Το UMLS αποτελείται από τρία εργαλεία. Το Metathesaurus, που περιλαμβάνει τους όρους από τα διάφορα λεξιλόγια, το Semantic Network, που είναι οι γενικές κατηγορίες και οι σχέσεις μεταξύ τους, καθώς και το SPECIALIST Lexicon and Lexical Tools, που πρόκειται για μία συλλογή εργαλείων για την επεξεργασία φυσικής γλώσσας. Παράλληλα, τα εργαλεία αυτά μπορούν να χρησιμοποιηθούν και ξεχωριστά ανάλογα τις εκάστοτε ανάγκες. Η πρόσβαση στο UMLS γίνεται μέσω του UMLS Terminology Services (UTS) με τρεις τρόπους. Απαιτείται όμως λογαριασμός και UMLS Metathesaurus License.

Ο πρώτος τρόπος εξαγωγής MeSH από κείμενο είναι η αναζήτηση σε Web browser. Οι εφαρμογές αυτές είναι το Metathesaurus Browser και το Semantic Network Browser.

Έπειτα υπάρχει η δυνατότητα κατεβάσματος του UMLS και προσαρμογή στις προσωπικές ανάγκες. Τέλος μπορούμε και μέσω Web API. Τα προγράμματα αυτά είναι τα MetaMap, NLM Medical Text Indexer (MTI) και SemRep. Για το MetaMap υπάρχει και η Lite έκδοση που είναι πιο γρήγορη αλλά με λιγότερο ακριβή αποτελέσματα. Όλα αυτά εντοπίζουν και εξάγουν concepts και σχέσεις από κείμενα. Τέλος υπάρχει και το online εργαλείο MeSH on Demand. Είναι ιδιαίτερα αξιόπιστο και γρήγορο. Τοποθετείς το κείμενο και σου επιστρέφει όλους τους MeSH όρους.

Συμπερασματικά, υπάρχει πλήθος εργαλείων που θα μπορούσαν να χρησιμοποιηθούν. Αυξάνονται σε αριθμό και βελτιώνονται ραγδαία με το πέρασμα των χρόνων. Επσης, αναπτύσσονται συνεχώς νέες τεχνικές που στηρίζονται στο machine learning .Έτσι έχει αλλάξει η επιστημονική έρευνα προς το καλύτερο, μιας και έχουν έρθει πιο κοντά στους επιστήμονες ισχυρά εργαλεία και υποδομές που συμβάλλουν στην ανάλυση δεδομένων.

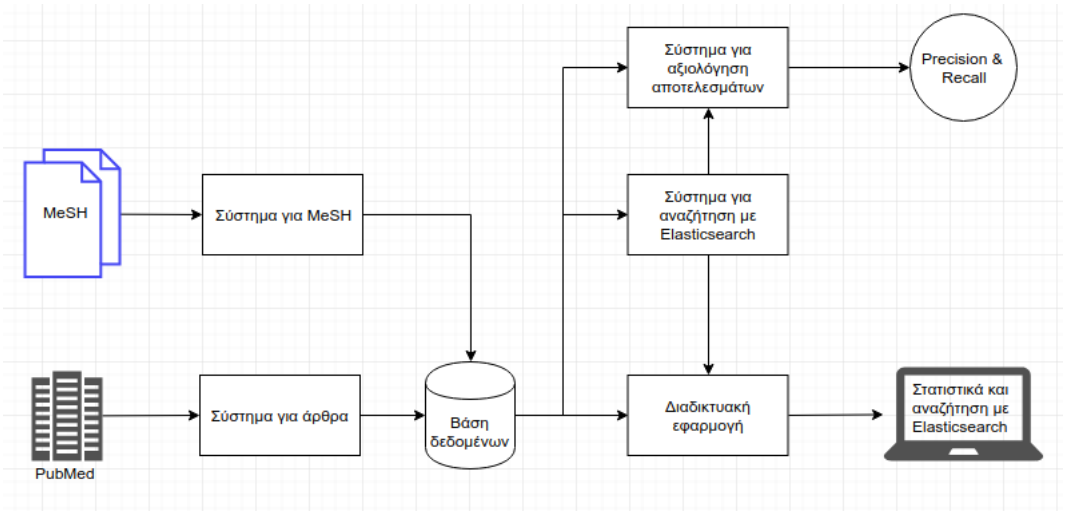
Κεφάλαιο 3

Προσέγγιση που Ακολουθήθηκε

3.1 Συνολική προσέγγιση

3.1.1 Αρχιτεκτονική συστήματος

Στα πλαίσια της εργασίας αυτής αναπτύχθηκαν επιμέρους συστήματα για: την εισαγωγή των άρθρων στη βάση, την εισαγωγή των MeSH όρων στη βάση, την αναζήτηση με χρήση του Elasticsearch, την αξιολόγηση των αποτελεσμάτων και τέλος την δημιουργία μιας διαδικτυακής εφαρμογής για την παρουσίαση στατιστικών και αναζήτηση άρθρων. Στην καρδιά του συστήματος βρίσκεται η βάση δεδομένων που δημιουργήθηκε για τη φιλοξενία τόσο των άρθρων που έχουν δημοσιευθεί στο PubMed όσο και των MeSH Headings που υπάρχουν, πάνω στην οποία στηρίχτηκαν όλες οι υπόλοιπες λειτουργίες. Στη πρώτη φάση του συστήματος γέμισε η βάση δεδομένων ενώ κατά τη δεύτερη χρησιμοποιήθηκαν τα δεδομένα αυτής της βάσης προκειμένου να παρέχεται η δυνατότητα βελτιωμένης αναζήτησης με αξιοποίηση του Elasticsearch. Η αξιολόγηση έγινε με την αξιοποίηση των MeSH Headings που έχουν αποδωθεί στα άρθρα. Τέλος, δημιουργήθηκε η διαδικτυακή εφαρμογή και βγήκαν τα στατιστικά στοιχεία, ενώ παρέχεται και η δυνατότητα αναζήτησης με το Elasticsearch. Η αρχιτεκτονική του συστήματος σε αφηρημένο επίπεδο φαίνεται στο παρακάτω σχήμα:



Σχήμα 3.1: Αρχιτεκτονική συστήματος.

Όπως βλέπουμε υπάρχουν 5 επιμέρους συστήματα:

1. **Σύστημα για άρθρα**: Δέχεται ως είσοδο τον όρο που ψάχνουμε (Sjogren) και αναλαμβάνει να κατεβάσει τα άρθρα από το PubMed και ακολούθως να τα αποθηκεύσει στη βάση δεδομένων.

2. **Σύστημα για MeSH**: Ως είσοδο παίρνει τα MeSH αρχεία και ως έξοδο αποθηκεύει στη βάση δεδομένων τα απαραίτητα στοιχεία.

3. **Σύστημα για αναζήτηση με Elasticsearch**: Με είσοδο τα abstract κείμενα των άρθρων και προσαρμογή του Elasticsearch γίνεται η αναζήτηση και προκύπτουν τα αποτελέσματά της.

4. **Σύστημα για αξιολόγηση αποτελεσμάτων**: Είσοδος είναι τα άρθρα που προκύπτουν από τα MeSH Headings και εκείνα από το Elasticsearch και γίνεται σύγκριση των αποτελεσμάτων.

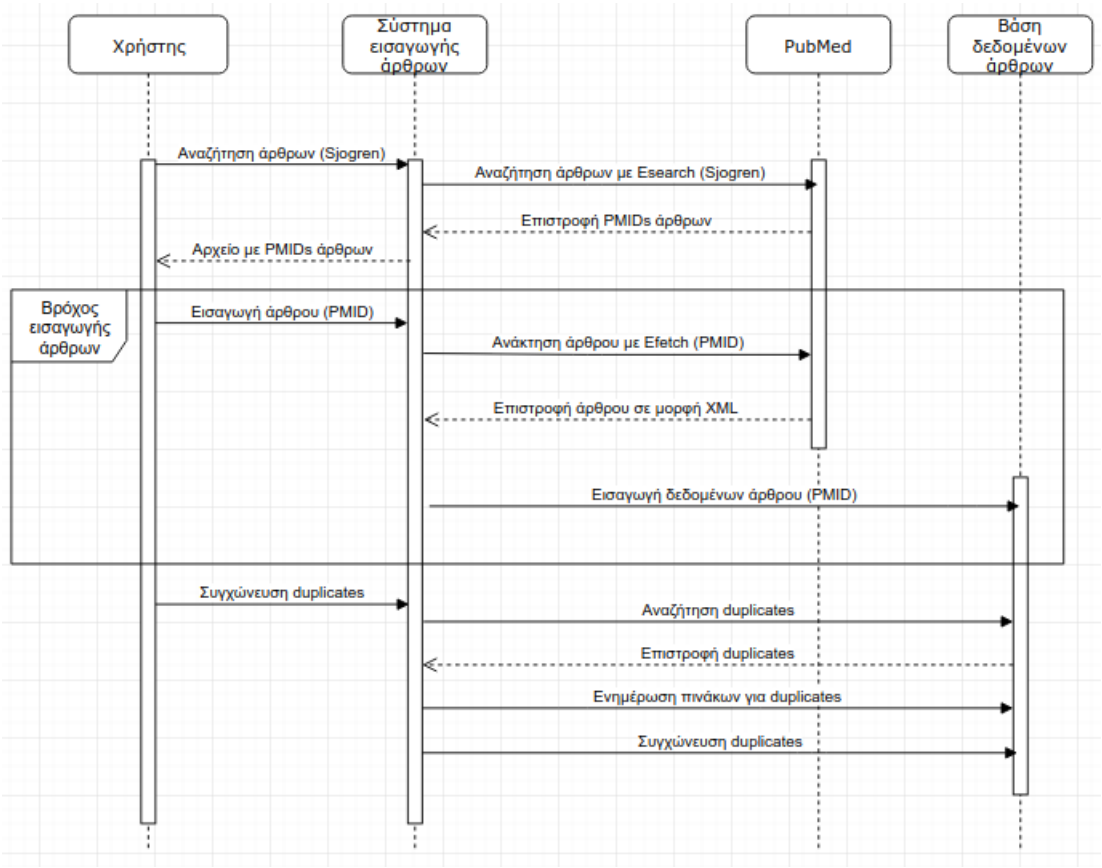
5. **Διαδικτυακή εφαρμογή**: Λαμβάνει τα δεδομένα από τη βάση και το Elasticsearch και παρουσιάζει τα στατιστικά αλλά δίνει και τη δυνατότητα αναζήτησης.

3.1.2 Αλληλεπίδραση μεταξύ των βασικών οντοτήτων (Sequence diagrams)

Για την περιγραφή της αλληλεπίδρασης μεταξύ των βασικών οντοτήτων χρησιμοποιούνται διαγράμματα ακολουθίας. Για κάθε ένα από τα συστήματα που παρουσιάστηκαν προηγουμένως, παραθέτουμε και το αντίστοιχο Sequence diagram.

Σύστημα για άρθρα

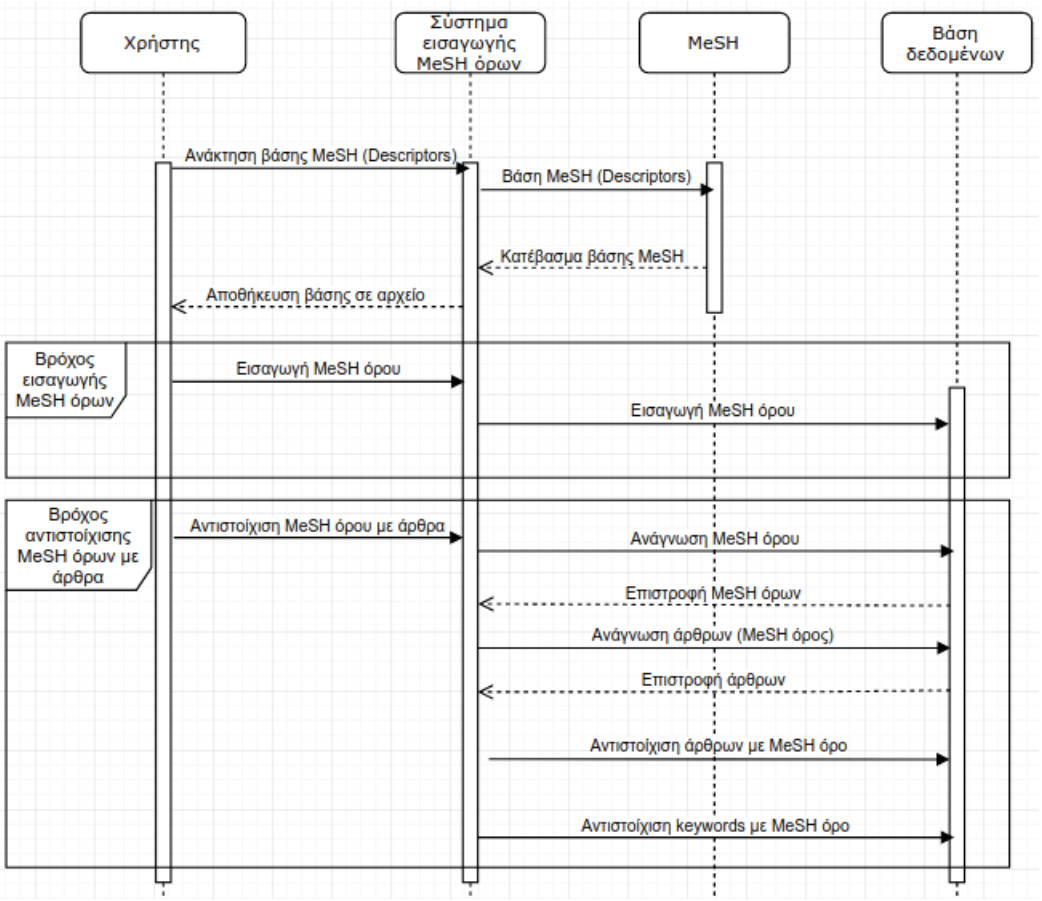
Το σύστημα αρχικά επικοινωνεί με το PubMed για να βρει τα IDs των άρθρων που περιέχουν τον όρο. Έπειρα για καθένα από αυτά τα PMIDs επικοινωνεί με το PubMed για να πάρει τη διαθέσιμη πληροφορία για κάθε άρθρο, την οποία επεξεργάζεται και ακολούθως σώζει στη βάση. Τέλος συγχωνεύει όλες τις duplicate εγγραφές για βελτιστοποίηση.



Σχήμα 3.2: Διάγραμμα ακολουθίας συστήματος για άρθρα.

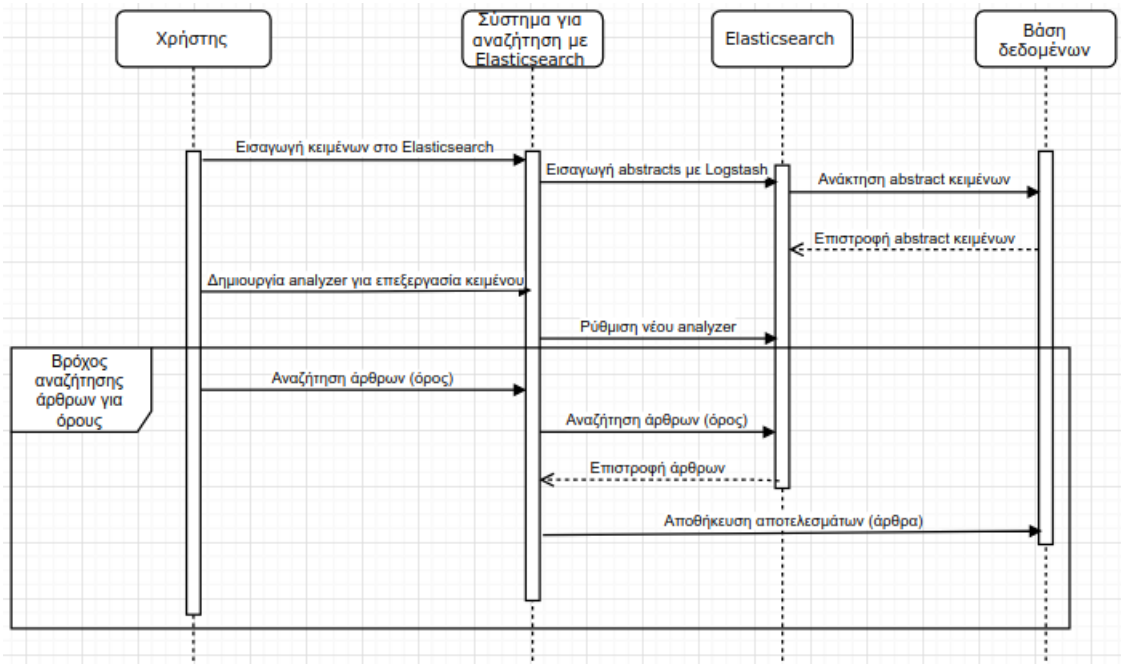
Σύστημα για MeSH

Το παρών σύστημα, πρώτα από όλα, ανακτά από τη βάση του MeSH τους descriptors και τους αποθηκεύει σε ένα αρχείο. Έπειτα κάθε όρος αντιστοιχείται στα αντίστοιχα άρθρα και keywords για την μετέπειτα αξιοποίηση του MeSH.



Σχήμα 3.3: Διάγραμμα ακολουθίας συστήματος για MeSH.

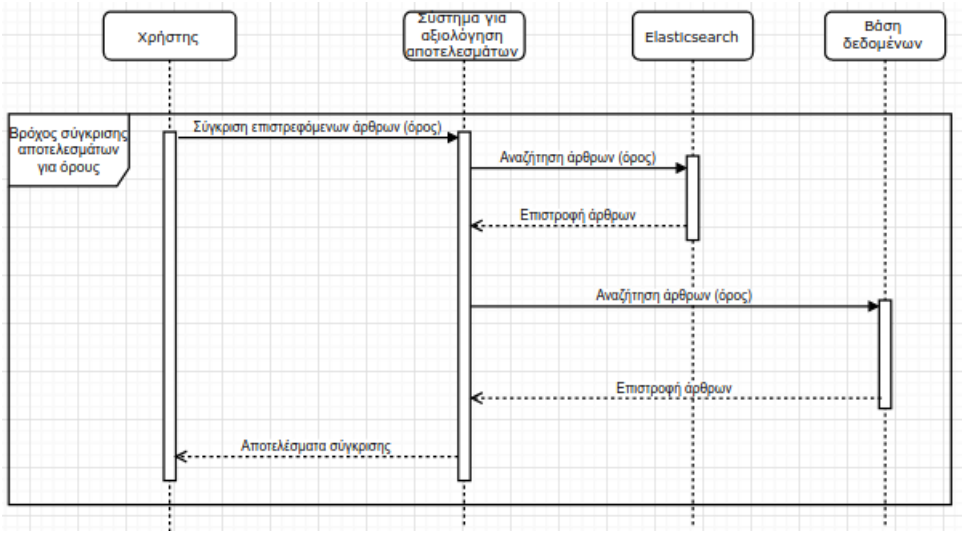
Στο παρακάτω διάγραμμα φαίνεται ότι το σύστημα χρειάζεται μία φάση αρχικοποίησης πριν χρησιμοποιηθεί από το χρήστη για την αναζήτηση των άρθρων. Αρχικά λοιπόν γίνεται η εισαγωγή των abstract κειμένων των άρθρων από τη βάση δεδομένων στο Elasticsearch και δημιουργείται κατάλληλος analyzer για την επεξεργασία ελεύθερου κειμένου ώστε η αναζήτηση να είναι προσαρμοσμένη στην περίπτωσή μας. Στη συνέχεια γίνεται η αναζήτηση άρθρων για όρους με τη χρήση του Elasticsearch και αποθηκεύονται τα αποτελέσματα στη βάση.



Σχήμα 3.4: Διάγραμμα ακολουθίας συστήματος για αναζήτηση με Elasticsearch.

Σύστημα για αξιολόγηση αποτελεσμάτων

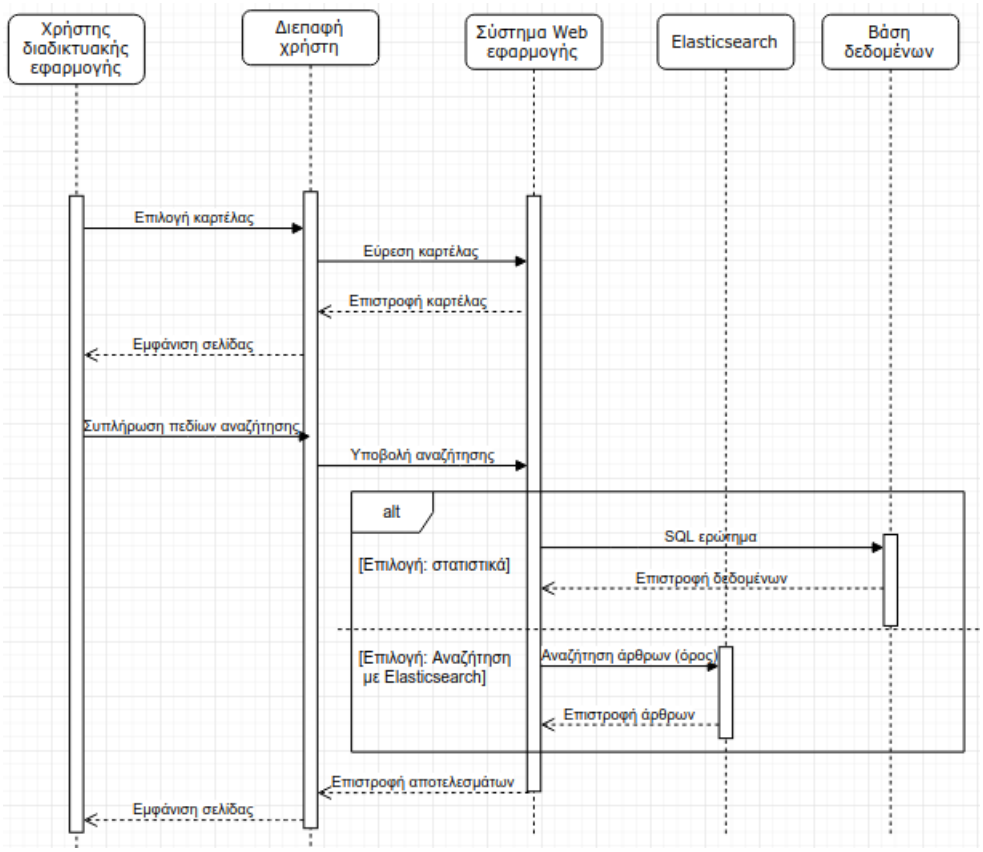
Για την αξιολόγηση των αποτελεσμάτων του Elasticsearch παράχθηκε το παρακάτω σύστημα. Αναζητά για κάποιος κλινικούς όρους τα άρθρα τόσο με το Elasticsearch όσο και βάση των MeSH Headings στη βάση δεδομένων. Επιστρέφει τα αποτελέσματα της σύγκρισης στον χρήστη.



Σχήμα 3.5: Διάγραμμα ακολουθίας συστήματος για αξιολόγηση αποτελεσμάτων.

Διαδικτυακή εφαρμογή

Η διαδικτυακή εφαρμογή σκοπεύει στην παρουσίαση στατιστικών στοιχείων και στη δυνατότητα να μπορεί ο χρήστης να αναζητά με τη χρήση του Elasticsearch. Ο χρήστης πλοηγείται στην ιστοσελίδα και επιλέγει ανάμεσα σε πλήθος εναλλακτικών. Αφού επιλέξει το είδος της πληροφορίας που θέλει, το σύστημα του επιστρέφει τη ζητούμενη καρτέλα. Στη συνέχεια ο χρήστης συμπληρώνει τα αντίστοιχα πεδία που χρειάζονται και γίνεται η αναζήτηση είτε στη βάση δεδομένων είτε στο Elasticsearch. Τελικά του επιστρέφεται η σελίδα με τα αποτελέσματα που ζήτησε.



Σχήμα 3.6: Διάγραμμα ακολουθίας διαδικτυακής εφαρμογής.

3.1.3 Τεχνολογίες που χρησιμοποιήθηκαν

Για την ανάπτυξη του λογισμικού και την ικανοποίηση των απαιτήσεων, επιλέχθηκαν οι κατάλληλες τεχνολογίες. Τα πλεονεκτήματά τους αναλύονται εκτενώς στο παράρτημα. Εδώ αναφέρονται συνοπτικά μαζί με τις εκδόσεις που χρησιμοποιήθηκαν.

- **Java** (openjdk version "1.8.0_191") ως γλώσσα προγραμματισμού για τη κατασκευή μεθόδων και queries.
- **MySQL και phpMyAdmin**(4.8.2) για τις όλες λειτουργίες σχετικά με τη βάση δεδομένων. Η εγκατάστασή της έγινε μέσω του XAMP 7.2.7-0. Πρόκειται για ένα πακέτο προγραμμάτων ελεύθερου λογισμικού που συμβάλλει στην ανάπτυξη web εφαρμογών. Η έκδοση που χρησιμοποιείται είναι η mysql Ver 14.14 Distrib 5.7.24, for Linux (x86_64) using EditLine wrapper.
- **Apache Tomcat** (8.5.33) ως web server για την εφαρμογή.
- **Eclipse** ως το ενοποιημένο περιβάλλον ανάπτυξης (IDE). Συγκεκριμένα το Eclipse Java EE IDE for Web Developers, η έκδοση Photon Release (4.8.0).
- **Elasticsearch** (6.5.2) για τη βελτιωμένη αναζήτηση.
- **Logstash** (6.5.2) για να σταλούν τα δεδομένα στο Elasticsearch.

3.2 Σχεδιασμός Βάσης Δεδομένων

3.2.1 Σχεδιασμός και μοντελοποίηση βάσης δεδομένων άρθρων

Ο σχεδιασμό της βάσης δεδομένων με τα άρθρα στηρίχτηκε στην επίσημη λίστα περιγραφής των attributes and elements [17]. Περιέχει τη δομή των άρθρων σε XML μορφή έτσι όπως επιστρέφονται από την αναζήτηση στο PubMed. Από το πλήθος των πληροφοριών κρατήθηκαν εκείνες που θεωρήθηκαν απαραίτητες για τη σύγκριση με τη βελτιωμένη αναζήτηση αλλά και χρήσιμες για την εξαγωγή στατιστικών στοιχείων. Οι βασικές δομές για τις οποίες κρατήθηκαν πληροφορίες είναι τα άρθρα και τα κείμενά τους, οι συγγραφείς, οι οργανισμοί στους οποίους ανήκουν, καθώς και οι λέξεις που έχουν αποδώσει οι ίδιοι στα άρθρα (Keywords) ή το PubMed (MeSH Headings). Ο σχεδιασμός του μοντέλου έγινε στο phpMyAdmin.

Τα βασικά στοιχεία που κρατήθηκαν σε πίνακες μαζί με κάποια σημαντικά πεδία τους είναι :

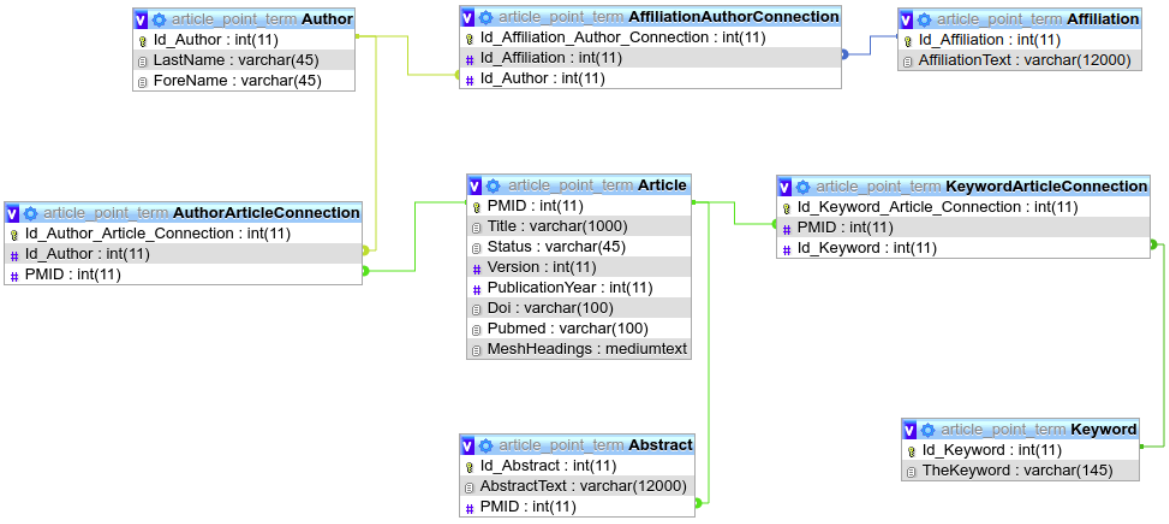
- **Article** : περιέχει το αναγνωριστικό id κάθε άρθρου (PMID), τον τίτλο (Title), την χρονιά έκδοσης (PublicationYear) και τους MeSH όρους του άρθρου (MeSH Headings). Τα τελευταία κρατήθηκαν σε ένα μόνο πεδία με τη μορφή JSON ώστε να γίνει ευκολότερη η αποθήκευση αρχικά και η ανάκτησή τους αργότερα.
- **Abstract** : με το κείμενο του άρθρου (AbstractText) και το PMID που το συνδέει με το αντίστοιχο άρθρο.
- **Author** : περιέχει το όνομα (ForeName) και του επώνυμο του συγγραφέα (LastName).

- **Affiliation** : ο οργανισμός στον οποίο ανήκει κάποιος συγγραφέας (Affiliation-Text).
- **Keyword** : η λέξη-κλειδί (TheKeyword) που έχει γράψει ο ίδιος ο συγγραφέας για το άρθρο.

Υπάρχουν και κάποια βοηθητικά στοιχεία που κρατήθηκαν αλλά δεν θα αναλυθούν παραπάνω αφού ο σκοπός τους είναι καθαρά βοηθητικός και δεν έχουν ουσιαστική σημασία. Επιπλέον δημιουργήθηκαν κάποιοι πίνακες για να συνδέσουν μεταξύ τους άλλους πίνακες. Αυτό γίνεται γιατί σε πολλές περιπτώσεις έχουμε N-N σχέσεις. Οι συνδέσεις είναι οι παρακάτω :

- **Article – Author** : ένα άρθρο μπορεί να έχει πολλούς συγγραφείς και ένας συγγραφέας να έχει γράψει πολλά άρθρα.
- **Article – Keyword** : ένα άρθρο μπορεί να έχει πολλές λέξεις κλειδιά, ενώ μία λέξη να αντιστοιχεί σε πολλά άρθρα.
- **Author – Affiliation** : ένας συγγραφέας μπορεί να ανήκει σε ένα οργανισμό αλλά και ένας οργανισμός να περιλαμβάνει πολλούς συγγραφείς.

Παρακάτω μπορούμε να δούμε την αναλυτική μορφή της βάσης δεδομένων (περιέχονται όλοι οι πίνακες με τα αντίστοιχα πεδία τους):



Σχήμα 3.7: Βάση δεδομένων άρθρων σε πλήρη μορφή.

3.2.2 Σχεδιασμός και μοντελοποίηση βάσης δεδομένων με MeSH όρους

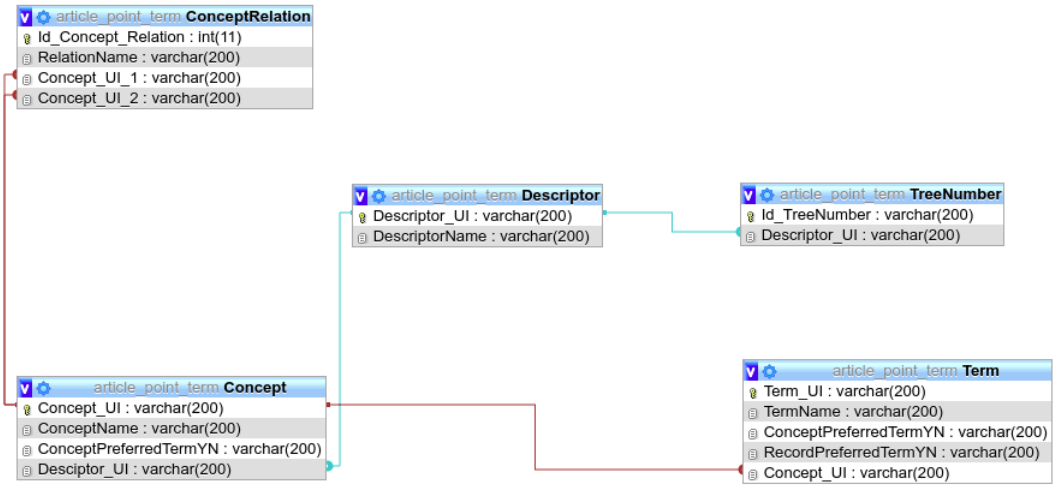
Όπως ήδη έχει αναφερθεί, το ελεγχόμενο λεξιλόγιο MeSH στηρίζεται στην έννοια του concept και των συνώνυμων όρων. Στο PubMed χρησιμοποιείται το MeSH τόσο στο indexing των άρθρων, όπου τους αποδίδονται συνδιασμοί Descriptors και

Qualifiers, όσο και στην αναζήτηση. Εμείς θέλουμε να αξιοποιήσουμε την πληροφορία από το indexing, μιας και τα άρθρα έχουν αντιστοιχηθεί σε MeSH όρους. Για να το πετύχουμε αυτό θα πρέπει να έχουμε αυτά τα δεδομένα σε μία βάση δεδομένων για να μπορούμε να τα επεξεργαζόμαστε.

Τα δεδομένα και η δομή τους είναι διαθέσιμα για κατέβασμα σε μορφή XML αρχείου. Από το official site του NLM [6] κατέβασαμε τα αρχεία για τους Descriptors σε μορφή XML, το desc2018.xml. Αρχικά για τους Descriptors, που αποτελούν το κύριο περιέχομενο των άρθρων και είναι το κέντρο της έρευνάς μας, επιλέχθηκαν τα βασικά στοιχεία. Συγκεκριμένα δημιουργήθηκε μία βάση δεδομένων με πίνακες για τους Descriptors, τα Concepts, τις σχέσεις μεταξύ αυτών, τους Terms και τέλος έναν πίνακα με τον αριθμό της θέσης του descriptor στο δέντρο. Στην παρούσα διπλωματική δεν ασχοληθήκαμε με τους qualifiers επειδή περιγράφουν τα θέματα από μία πιο συγκεκριμένη οπτική και παίζουν συμπληρωματικό ρόλο. Οι πληροφορίες που κρατήθηκαν, πιο αναλυτικά, είναι οι εξής :

- Descriptor : Περιέχει το όνομα του descriptor (DescriptorName) και το μοναδικό του UI.
- Concept : Διατηρεί το όνομα του concept (ConceptName), το αν είναι το preferred concept ή όχι, το μοναδικό του UI, καθώς και το UI του descriptor στο οποίο ανήκει.
- ConceptRelation : Κρατάει τα δύο UI μεταξύ των οποίων υπάρχει σχέση, όπως και το όνομα αυτής της σχέσης (RelationName).
- Term : Περιέχει το όνομα του όρου (TermName), το UI του concept στο οποίο ανήκει, καθώς και αν είναι preferred όρος και αν το concept στο οποίο ανήκει είναι το preferred.
- TreeNumber : Εδώ κρατείται μόνο ο αριθμός θέσης στο δέντρο και το UI του αντίστοιχου descriptor.

Δεν χρειάστηκαν παραπάνω πίνακες καθώς δεν υπάρχουν N-N σχέσεις. Για παράδειμα ένα concept μπορεί να έχει πολλά terms, όμως ένα term αντιστοιχεί σε ένα μόνο concept. Το ίδιο και με τα υπόλοιπα. Η βάση δεδομένων των Descriptors αναλυτικά έχει ως εξής:



Σχήμα 3.8: Βάση δεδομένων άρθρων σε πλήρη μορφή.

Κεφάλαιο 4

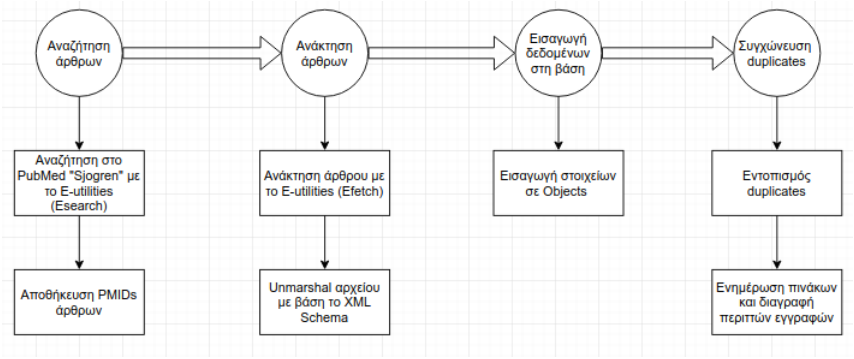
Υλοποίηση Επιμέρους Συστημάτων

4.1 Ανάκτηση και Αποθήκευση Άρθρων

Το πρώτο βήμα στην υλοποίηση του συστήματος ήταν η εισαγωγή των άρθρων στη βάση δεδομένων. Η αναζήτηση των άρθρων στο PubMed έγινε με το E-utilities (Esearch) και τα ζητούμενα PMIDs έγινε με HTTP request. Για κάθε PMID ανακτήσαμε το αντίστοιχο άρθρο με το E-utilities (Efetch) και HTTP request. Με βάση το XML Schema των άρθρων έγινε unmarshal το αρχείο και με κατάλληλες κλάσεις τα εισάγαμε στη βάση. Τέλος έγινε συγχώνευση των duplicate εγγραφών.

Είσοδος συστήματος: Ο όρος "Sjogren" στην αναζήτηση.
Έξοδος συστήματος: Τα στοιχεία των άρθρων έχουν εισαχθεί στη βάση δεδομένων.

Η διαδικασία που ακολουθήθηκε επιγραμματικά φαίνεται στο παρακάτω σχήμα, ενώ στη συνέχεια αναλύονται τα επιμέρους βήματα.



Σχήμα 4.1: Βάση δεδομένων άρθρων σε πλήρη μορφή.

4.1.1 Αναζήτηση άρθρων σχετικά με το σύνδρομο Sjogren

Αρχικά λοιπόν ψάξαμε όλα τα άρθρα που σχετίζονται με το Sjogren. Αυτό έγινε με τη χρήση χρήση του E-utilities [11]. Πρόκειται για ένα API που επιτρέπει την αναζήτηση στις NCBI βάσεις (στην περίπτωση μας χρησιμοποιήθηκε μόνο η βάση του PubMed) μέσα από το πρόγραμμά μας. Για την αναζήτηση και ανάκτηση δεδομένων το E-utilities χρησιμοποιεί δομημένα URLs. Ζητά από το NCBI server τις πληροφορίες και αυτές επιστρέφονται σε συγκεκριμένο format. . Η βάση του URL είναι η διεύθυνση του E-utilities server και είναι https://eutils.ncbi.nlm.nih.gov/entrez/eutils/ . Στην συνέχεια επιλέγεται το ανάλογο εργαλείο. Στη συγκεκριμένη περίπτωση χρειάστηκε το Esearch, το οποίο ψάχνει ένα ερώτημα κειμένου σε μία βάση δεδομένων και επιστρέφει μία λίστα μοναδικών αναγνωριστικών (UIDs) που ταιριάζουν με το ερώτημα. Ειδικά επιστρέφει μια λίστα PMIDs (id των άρθρων). Στο τέλος προστίθενται η κατάληξη "fcgi?".Μπορούν να προστεθούν και άλλες παράμετροι στο query. Οι δύο υποχρεωτικοί είναι η βάση δεδομένων (db) στην οποία προσδιορίζεται που θα ψάξει και ο όρος (term) που είναι το κείμενο του ερωτήματος. Για βάση θέλουμε "pubmed" και όρο το "Sjogren". Επιπλέον, χρησιμοποιήσαμε δύο ακόμα προαιρετικές παραμέτρους: το retmax που δηλώνει το μέγιστο αριθμό UIDs που επιστρέφονται και το retmode που αφορά το format της ανακτούμενης πληροφορίας. Το retmax είναι "10.000" και το retmode "xml". Συνολικά το τελικό URL ήταν το εξής:

https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?
db=pubmed&retmax=10000&term=Sjogren&retmode=xml

Για να ζητήσουμε από το server την πληροφορία που θέλουμε, χρησιμοποιήσαμε το πρωτόκολλο HTTP. Στέλνουμε ένα HTTP request με μέθοδο GET και αν δεν παρουσιαστεί κάποιο πρόβλημα, μας επιστρέφεται ένα HTTP response. Η απάντηση αυτή περιείχε όλα τα UIDs (και συγκεκριμένα PMIDs) που σχετίζονται με το Sjogren όπως ακριβώς ζητήσαμε. Την απάντηση την αποθηκεύσαμε σε αρχείο με τη μορφή xml ώστε να μπορέσουμε στη συνέχεια να κάνουμε την επεξεργασία.

4.1.2 Ανάκτηση άρθρων

Εφόσον πλέον γνωρίζουμε ποια άρθρα θέλουμε μέσω των PMIDs τους, τώρα πρέπει να τα ανακτήσουμε. Πριν γίνει όμως αυτό πρέπει να γίνει η κατάλληλη προετοιμασία. Χρειάζεται να είμαστε σε θέση να αντλήσουμε συγκεκριμένα στοιχεία που επιθυμούμε από τα άρθρα, καθώς και να τα βάζουμε στη βάση δεδομένων.

Βρήκαμε λοιπόν τη δομή των άρθρων που επιστρέφονται ώστε να μπορούμε να πάρουμε τα εκάστοτε στοιχεία που μας ενδιαφέρουν. Για το σκοπό αυτό κατέβασα το pubmed_180101.dtd από το site [3], το οποίο καθορίζει τα περιεχόμενα των άρθρων. Μιας και το πρόγραμμα είναι σε Java, ο στόχος μας είναι η μετατροπή του xml αρχείου που ανακτήσαμε σε μορφή κατάλληλη για επεξεργασία. Αυτό επιτεύχθηκε με τη βοήθεια του JAXB (Java Architecture for XML Binding). Πρόκειται για μία βιβλιοθήκη που βοηθά τη μετατροπή XML σχημάτων σε Java αναπαραστάσεις και το αντίστροφο. Συγκεκριμένα παρέχει έναν μηχανισμό που κάνει τα Java objects σε XML, το λεγόμενο marshal. Η αντίστροφη διαδικασία (μετατροπή XML σε Java objects) ονομάζεται unmarshal.

Προτού όμως γίνει η μετατροπή του XML σε Java object, πρέπει να γνωρίζουμε τη μορφή του Object αυτού και συγκεκριμένα τη κλάση του. Δεδομένου ότι γνωρίζαμε τη μορφή του XML σχήματος από το pubmed_180101.dtd που αναφέραμε προηγουμένως, παράγαμε τις Java κλάσεις που αντιστοιχούν την JAXB binding compiler 'xjc' εντολή. Η μορφή της εντολής ήταν η εξής:

xjc -d /home/user/Downloads/check_xjc/ -dtd
"https://dtd.nlm.nih.gov/ncbi/pubmed/out/pubmed_180101.dtd"

Με την εντολή αυτή παράχθηκε ένας φάκελος generated που περιέχει όλες τις ζητούμενες Java κλάσεις, με βάση τις οποίες γίνεται αργότερα το unmarshal.

Πριν το unmarshal όμως έπρεπε να έχουμε προετοιμαστεί να βάλουμε τα δεδομένα στη βάση. Για να διευκολύνουμε λοιπόν την εισαγωγή αυτή, κατασκευάσαμε κλάσεις που αντιστοιχούν σε όλους τους πίνακες της βάσης δεδομένων. Για κάθε πίνακα λοιπόν δημιουργήθηκε μία κλάση που περιέχει τα πεδία του, έναν constructor για την αρχικοποίησή του, καθώς και μία μέθοδο που κατασκευάζει και γυρνάει σε μορφή String το sql query που αναλαμβάνει το insertion στη βάση του εκάστοτε στοιχείου. Με αυτόν τον τρόπο διαχωρίσαμε την αποθήκευση των δεδομένων από το υπόλοιπο πρόγραμμα. Έτσι γλυτώσαμε περιττό κώδικα από το κύριο πρόγραμμα, γεγονός που βοηθά στη συντήρησή του και το debugging. Επιπλέον, αν γίνει κάποια αλλαγή στη βάση γίνεται ευκολότερη η τροποποίηση στο πρόγραμμα.

Αφού έχουμε κατασκευάσει όλες τις απαραίτητες κλάσεις και έχουμε μεταφέρει τον φάκελο generated, που παράχθηκε προηγουμένως, μέσα στο project μας στο Eclipse με τη μορφή package, πλέον είμαστε κατάλληλα προετοιμασμένοι για να δεχτούμε τα άρθρα-δεδομένα από το PubMed. Αρχικά διαβάσαμε το xml αρχείο που δημιουργήσαμε προηγουμένως και περιέχει όλα τα PMIDs των άρθρων που θέλουμε. Ενδεικτικά αναφέρουμε ότι συνολικά επιστράφηκαν 6417 άρθρα. Έχουμε λοιπόν ένα βρόχο (loop) και σε κάθε κάθε κύκλο διαβάσαμε ένα PMID από τα 6417 συνολικά. Αυτό είναι και το κύριο πρόγραμμά μας. Δηλαδή αυτό "τρέξαμε" για να γίνει η εισαγωγή όλων των άρθρων. Σε κάθε κύκλο λοιπόν μέσω διαφόρων κλάσεων έγινε η εισαγωγή στη βάση δεδομένων όλων των στοιχείων που σχετίζονται με το άρθρο με το τρέχον PMID.

Πιο αναλυτικά, με το που παίρναμε το PMID γνωρίζαμε πλέον ποιο άρθρο θέλουμε να ανακτήσουμε. Όπως έχει αναφερθεί και προηγουμένως, χρησιμοποιήθηκε το E-utilities. Στην αναζήτηση είχαμε χρησιμοποιήσει το ESearch. Τώρα όμως χρησιμοποιήσαμε την κατάλληλη λειτουργία για την ανάκτηση δεδομένων. Αυτή είναι το EFetch, το οποίο επιστρέφει ολόκληρη την καταχώρηση για ένα συγκεκριμένο UID (ή και περισσότερα από ένα αν το επιθυμούμε). Όπως και και προηγουμένως, έπρεπε να δηλώσουμε τη βάση δεδομένων (db) από την οποία ανακτούμε τα δεδομένα, καθώς και τον τύπο (retmode) που επιστρέφεται η πληροφορία αυτή. Το db είναι "pubmed" και το retmode "xml". Η επιπλέον παράμετρος σε σχέση με το ESearch είναι το id στο οποίο βάζουμε το PMID του άρθρου που θέλουμε. Το τελικό URL είναι το εξής :

https://eutils.ncbi.nlm.nih.gov/entrez/eutils
/efetch.fcgi?db=pubmed&retmode=xml&id=<PMID>,
όπου <PMID>το εκάστοτε id άρθρου.

Η πρώτη ενέργεια ήταν να ζητήσουμε την πληροφορία που θέλουμε από το server με τον ίδιο τρόπο που το κάναμε και πριν με τη χρήση του πρωτοκόλλου HTTP. Στείλαμε

ένα HTTP request με τη μέθοδο GET και λάβαμε response την πλήρη καταχώρηση (full record) για το συγκεκριμένο άρθρο. Πλέον έχοντας το άρθρο σε μορφή XML, μπορούμε να το μετατρέψουμε σε JAVA object με unmarshal. Έτσι καταλήξαμε να έχουμε ένα αντικείμενο άρθρο (PubMedArticle) το οποίο περιέχει όλα τα στοιχεία του record. Τα στοιχεία αυτά μπορούμε να τα προσπελάσουμε με την κλήση μεθόδων από το αρχικό άρθρο object.

Από το κυρίως πρόγραμμα, καλούνται επιμέρους μέθοδοι που αναλαμβάνουν την εισαγωγή συγκεκριμένων στοιχείων. Για παράδειγμα άλλη μέθοδος ανέλαβε τα άρθρα και άλλη τους συγγραφείς. Σε κάθε τέτοια μέθοδο ανακτούνται οι αντίστοιχες πληροφορίες και με τη βοήθεια των κλάσεων που έχουμε δημιουργήσει για το διαχωρισμό του κύριου προγράμματος και της εισαγωγής δεδομένων στη βάση. Συνεπώς σε κάθε μέθοδο έγινε και η εισαγωγή της αντίστοιχης πληροφορίας. Επειδή όμως κάποιοι πίνακες σχετίζονται μεταξύ τους, η εισαγωγή έπρεπε να γίνει με συγκεκριμένη σειρά προκειμένου να αποφευχθούν προβλήματα σχετικά με ξένα κλειδιά. Για παράδειγμα το άρθρο έπρεπε να εισαχθεί πριν το abstract κείμενο διότι το abstract έπρεπε να "δείχνει" στο πρωτεύων κλειδί του άρθρου (PMD). Ειδικά στις περιπτώσεις που είχαμε N-N σχέσεις κρινόταν επιτακτική η ανάγκη συγκεκριμένης σειράς εισαγωγής. Πιο αναλυτικά η εισαγωγή έγινε ως εξής :

1. Article και Keywords : Αρχικά έγινε η εισαγωγή του άρθρου με όλα τα στοιχεία του και έπειτα όλα εκείνα τα keywords που του αντιστοιχούν.

2. Abstract : Το κείμενο που αντιστοιχεί στο άρθρο αυτό.

3. Author : Εισήχθει ο συγγραφέας με τα κατάλληλα στοιχεία.

4. Affiliation : Στη συνέχεια έγινε και η εισαγωγή των οργανισμών στους οποίους ανήκει ο κάθε συγγραφέας. Μετά από αυτό εισήχθει και ο επόμενος συγγραφέας (βήμα 3) κ.ο.κ.

Παραλείπεται πότε εισάγονται δεδομένα στους συνδετικούς πίνακες για λόγους απλότητας και ευκολίας.

Μόλις τελείωσε το πρόγραμμα, πλέον ολοκληρώθηκε η εισαγωγή των άρθρων και των ανάλογων πληροφοριών στη βάση δεδομένων μας. Για τα insertions στη βάση δεδομένων χρησιμοποιήθηκαν parameterized queries για να μην υπάρχουν προβλήματα με τους ειδικούς χαρακτήρες, ενώ όπου υπήρχε η δυνατότητα χρησιμοποιήθηκε το batch για βελτίωση της απόδοσης του συστήματος. Επίσης, επειδή αρκετά άρθρα στο PubMed δεν περιέχουν όλα τα απαραίτητα στοιχεία (συγγραφείς, abstract κείμενα, κ.α.) έγινε χειρισμός εξαιρέσων. Τέλος, επειδή το πρόγραμμα συνολικά ήταν πολύ κοστοβόρο τόσο σε χρόνο όσο και σε πόρους, αυξήσαμε το Heap size.

4.1.3 Συγχώνευση όμοιων εγγραφών (duplicates)

Κατά την εισαγωγή των άρθρων δεν έγινε πουθενά έλεγχος αν κάποια εγγραφή (π.χ. συγγραφέας) υπήρχε ήδη στη βάση δεδομένων. Αυτό είχε ως αποτέλεσμα στο τέλος να υπάρχουν κάποιες ταυτόσημες εγγραφές. Δεν επηρέασε λειτουργικά το σύστημα, το επιβάρυνε όμως σε κάποιο βαθμό. Για το λόγο αυτό βρήκαμε πρώτα ποιες εγγραφές είναι duplicates και από αυτές κρατήσαμε μόνο τη μία. Παράλληλα ενημερώθηκαν και όποιοι πίνακες σχετίζονταν με τις εγγραφές αυτές. Οι πίνακες στη βάση δεδομένων με τα άρθρα, που χρειάστηκαν έλεγχο, είναι οι Author, Affiliation

και Keyword. Στη βάση δεδομένων του MeSH δεν υπήρχε κανένα duplicate οπότε δεν χρειάστηκε να γίνει κάποια ενέργεια.

Keywords : Αφορά τις εγγραφές που είχαν την ίδια λέξη-κλειδί. Κάθε λέξη όμως αντιστοιχεί σε ένα άρθρο. Αυτή η αντιστοίχιση μεταφράζεται μέσω ενός πίνακα (KeywordArticleConnection). Συνεπώς πρώτα έπρεπε να ενημερωθεί αυτός ο πίνακας ώστε να "δείχνει" στην μία και μοναδική λέξη-κλειδί που απομένει από τα duplicates και στη συνέχεια να διαγραφούν οι περιττές εγγραφές. Αυθαίρετα επιλέξαμε να κρατάμε την λέξη-κλειδί με το μικρότερο id. Ενημερώσαμε λοιπόν τον πίνακα με τις σχέσεις να "κοιτάει" στο μικρότερο id και αφού ολοκληρωθεί αυτή η διαδικασία, πήγαμε στον πίνακα Keyword και διαγράψαμε τα duplicates που δεν έχουν το μικρότερο id. Τα βήματα με τα οποία εκτελέστηκαν αυτές οι ενέργειες είναι τα εξής:

(1) Ενημέρωση του πίνακα της σχέσης του άρθρου με τη λέξη-κλειδί
(2) Διαγραφή των περιττών εγγραφών Keyword

Authors: Αφορά τους συγγραφείς που είχαν ίδιο επώνυμο και είτε ίδιο όνομα είτε το όνομα του ενός να ήταν το αρχικό του άλλου (π.χ. James Paul με James P). Πρώτα ασχοληθήκαμε με την πρώτη περίπτωση που είχαμε ίδιο επώνυμο και ίδιο όνομα. Ο συγγραφέας έχει σχέσεις με δύο άλλους πίνακες, τα άρθρα (AuthorArticleConnection) και τους οργανισμούς στους οπι (AffiliationAuthorConnection). Συνεπώς έπρεπε να ενημερωθούν και οι δύο πριν γίνει οποιαδήποτε διαγραφή. Αρχικά λοιπόν ενημερώσαμε τον πίνακα με τις σχέσεις μεταξύ συγγραφέα και άρθρου να "δείχνει" στο author των duplicates με το μικρότερο id. Το αντίστοιχο κάναμε και για τον πίνακα σχέσης συγγραφέα-affiliation. Στη συνέχεια αφαιρέσαμε τα duplicates από τον πίνακα Author κρατώντας εκείνο με το ελάχιστο id. Τα βήματα ήταν τα εξής:

(1) Ενημέρωση του πίνακα της σχέσης του άρθρου με το συγγραφέα
(2) Ενημέρωση του πίνακα της σχέσης του οργανισμού με το συγγραφέα
(3) Διαγραφή των περιττών εγγραφών Author

Έπειτα ασχοληθήκαμε με την περίπτωση που το επώνυμο ήταν ίδιο και το όνομα του ενός ήταν το αρχικό γράμμα του άλλου. Με παρόμοια διαδικασία με πριν ενημερώσαμε τους πίνακες σχέσεων συγγραφέα-άρθρου και συγγραφέα-οργανισμού. Ανάλογα διαγράψαμε τις περιττές εγγραφές από τον Author, όχι όμως με βάση το μικρότερο id όπως προηγουμένως, αλλά με βάση αυτό που είχε το μεγαλύτερο μήκος ForeName(δεν ήταν δηλαδή το γράμμα μόνο του). Έτσι βεβαιωνόμαστε ότι κρατήσαμε το πλήρες όνομα κι όχι μόνο το αρχικό γράμμα. Τα βήματα με τα οποία εκτελέστηκαν αυτές τις ενέργειες είναι τα εξής :

(1) Ενημέρωση του πίνακα της σχέσης του άρθρου με το συγγραφέα
(2) Ενημέρωση του πίνακα της σχέσης του οργανισμού με το συγγραφέα
(3) Διαγραφή των περιττών εγγραφών Author

Affiliations: Αφορά τους ίδιους οργανισμούς . Με παρόμοια διαδικασία με προηγουμένως, αρχικά ενημερώθηκε ο πίνακας σχέσης μεταξύ συγγραφέα και οργανισμού ώστε να "δείχνει" στο affiliation των duplicates με το μικρότερο id. Στη συνέχεια αφαιρέθηκαν τα duplicates από τον πίνακα Affiliation και κρατήθηκε αυτό με το ελάχιστο id. Τα βήματα ήταν τα εξής:

(1) Ενημέρωση του πίνακα της σχέσης του συγγραφέα με τον οργανισμό
(2) Διαγραφή των περιττών εγγραφών Affiliation

Με το πέρας των διαγραφών των περιττών εγγραφών και των αντίστοιχων ενημερώσεων, πετύχαμε βελτίωση του συστήματος. Συγκεκριμένα καταγράφηκαν οι μεταβολές στο πλήθος των εγγραφών για τους 3 αυτούς πίνακες. Οι πίνακες σχέσεων, όπως ήταν λογικό και αναμενόμενο, έμειναν αμετάβλητοι ως προς το πλήθος εγγραφών.

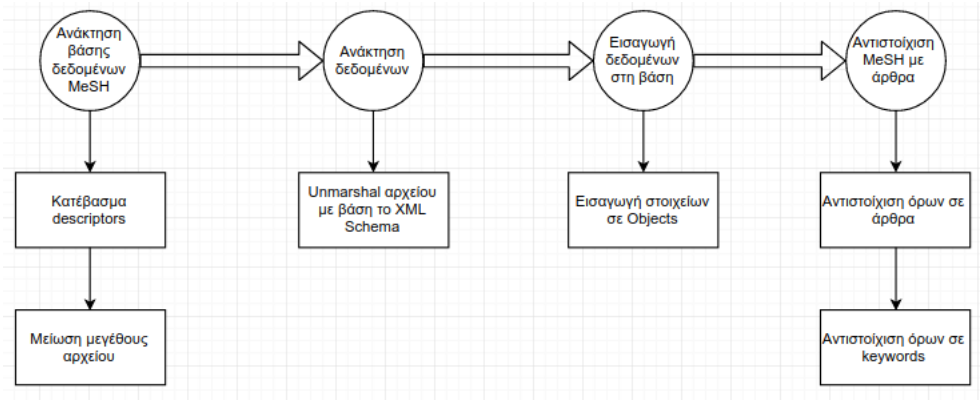
Keywords: από 8.448 εγγραφές έμειναν 4.558 → **μείωση 46%**
Authors: από 36.667 εγγραφές αρχικά πήγαν 24.269 και τελικά 22.294
→ **μείωση 39%**
Affiliations: από 16.038 πήγαν 9507 εγγραφές → **μείωση 40%**

4.2 Επεξεργασία και Αποθήκευση MeSH όρων

Μετά την εισαγωγή των άρθρων στη βάση έγινε η εισαγωγή των MeSH όρων. Κατεβάσαμε το XML αρχείο που περιέχει τη βάση δεδομένων των Descriptors διότι αυτούς θα χρησιμοποιήσουμε στην περίπτωσή μας. Μειώσαμε το μέγεθος αυτού του αρχείου για να γίνει διαχειρίσιμο και με βάση των XML Schema του ανακτήσαμε τα επιθυμητά δεδομένα. Κάθε στοιχείο το εισάγαμε στη βάση με τη βοήθεια κατάλληλων κλάσεων. Στο τέλος έγινε η αντιστοίχιση των MeSH όρων με τα άρθρα και τα keywords.

Είσοδος συστήματος: Η βάση δεδομένων του MeSH για τους Descriptors.
Έξοδος συστήματος: Οι MeSH όροι έχουν εισαχθεί στη βάση και έχει γίνει η αντιστοίχισή τους με τα άρθρα.

Στο παρακάτω σχήμα φαίνεται η διαδικασία που ακολουθήθηκε επιγραμματικά. Τα επιμέρους βήματα αναλύονται στη συνέχεια.



Σχήμα 4.2: Διαδικασία εισαγωγής MeSH όρων στη βάση δεδομένων.

4.2.1 Ανάκτηση και Αποθήκευση MeSH όρων

Πρώτα από όλα κατεβάσαμε τη βάση των descriptor. Το αρχείο ήταν σε XML μορφή και μεγάλου μεγέθους. Για μπορέσουμε να το επεξεργαστούμε όμως έπρεπε να μειωθεί το μέγεθος του. Για το λόγο αυτό δημιουργήθηκε ένα πρόγραμμα το οποίο αναλαμβάνει το έργο αυτό. Συγκεκριμένα, επειδή για την εισαγωγή των descriptors δεν χρειάζονται οι επιτρεπόμενοι qualifiers, αφαιρέθηκε το πεδίο αυτό με tag <AllowableQualifiersList>. Έτσι μειώθηκε το μέγεθος κατά 60% και έγινε πλέον διαχειρίσιμο το αρχείο.

Πριν αρχίσουμε το κυρίως πρόγραμμα, χρησιμοποιήσαμε το XML Schema που περιγράφει τη δομή των αρχείων των descriptors σε Java Objects. Με αυτόν τον τρόπο ήμασταν σε θέση να τα διαχειριστούμε και να αντλήσουμε τις πληροφορίες που θέλουμε πιο εύκολα. Για την παραγωγή των κλάσεων χρησιμοποιήσαμε την JAXB binding compiler 'xjc' εντολή. Συγκεκριμένα με την παρακάτω εντολή παράχθηκε ένας φάκελος με όλες τις ζητούμενες κλάσεις.

xjc -d /home/user/Downloads/check_xjc_2/ -dtd "https://www.nlm.nih.gov /databases/dtd/nlmdescriptorrecordset_20180101.dtd"

Στη συνέχεια για να είμαστε σε θέση να βάλουμε τα δεδομένα στη βάση, όπως κάναμε και προηγουμένως στην περίπτωση των άρθρων, κατασκευάσαμε κατάλληλες κλάσεις. Κάθε κλάση περιέχει τα πεδία του πίνακα, έναν constructor για την αρχικοποίησή του, καθώς και μία μέθοδο που κατασκευάζει και γυρνάει σε μορφή String το sql query που αναλαμβάνει το insertion στη βάση του εκάστοτε στοιχείου. Εργαστήκαμε δηλαδή τελείως ανάλογα με τα άρθρα. Έτσι διαχωρίσαμε πάλι την αποθήκευση των δεδομένων από το υπόλοιπο πρόγραμμα και γλυτώσαμε περιττό κώδικα από το κύριο πρόγραμμα, γεγονός που βοηθά στη συντήρησή του και το debugging. Επίσης, αν γίνει κάποια αλλαγή στη βάση γίνεται ευκολότερη η τροποποίηση στο πρόγραμμα.

Εφόσον γνωρίζουμε τη δομή των αρχείων και έχουμε κατασκευάσει όλες τις απαιτούμενες κλάσεις, μπορούμε πλέον να εισάγουμε τα δεδομένα στη βάση. Αρχικά λοιπόν διαβάσαμε το τροποποιημένο αρχείο των descriptors. Για τη μετατροπή του XML σε Java Object κάναμε unmarshal χρησιμοποιώντας τις κλάσεις που προέκυψαν από το XML Schema. Έτσι προέκυψε ένα αντικείμενο που περιέχει όλους τους descriptors και τις αντίστοιχες πληροφορίες του, οι οποίες είναι προσπελάσιμες μέσα από την κλήση μεθόδων από το αρχικό αντικείμενο. Φυσικά δεν έγινε άμεσα η εισαγωγή όλων των στοιχείων στο κύριο πρόγραμμα αλλά καλούνται επιμέρους. Κάθε επιμέρους ανέλαβε να αντλήσει τις αντίστοιχες πληροφορίες και να τις εισάγει. Επιπλέον, οι εισαγωγές έπρεπε να γίνονται με συγκεκριμένη σειρά ώστε να μην υπάρχουν θέματα με τα ξένα κλειδιά. Για παράδειγμα ένα term δεν μπορούσε να μπει πριν το αντίστοιχο του concept διότι έχει ξένο κλειδί το πρωτεύον κλειδί του concept. Η σειρά με την οποία εισάχθηκαν τα δεδομένα είναι η παρακάτω :

1. **Descriptor** : Αρχικά εισήχθει ο descriptor με τα κατάλληλα στοιχεία του.
2. **TreeNumber** : Για κάθε descriptor έγιναν εισαγωγή όλοι οι αριθμοί θέσεων στο δέντρο που του αντιστοιχούν.
3. **Concept** : Έπειτα έγινε εισαγωγή του κάθε concept με τα αντίστοιχα στοιχεία του.
4. **Term** : Για κάθε concept εισήχθησαν οι όροι εκείνοι που ανήκουν στο concept αυτό.

Επειδή τα **ConceptRelation** προϋποθέτουν την ύπαρξη των concept, μιας και έχουν ξένα κλειδιά τα πρωτεύοντά τους, η εισαγωγή τους στη βάση δεδομένων έγινε αφού έχουν εισαχθεί όλα τα υπόλοιπα προηγομένως. Για αυτό το λόγο κατασκευάσαμε ένα επιπλέον πρόγραμμα. Διαβάσαμε όπως και πριν το τροποποιημένο αρχείο και κάναμε unmarshal. Η σειρά διαβάσματος έγινε με τον ίδιο τρόπο, απλώς τώρα δεν εισάγαμε όλα τα δεδομένα. Μας αποσχόλησαν μόνο τα ConceptRelation. Αυτά υπάρχουν στο αρχείο μέσα σε κάθε concept. Για κάθε concept δηλαδή βρήκαμε με ποια άλλα concept σχετίζεται αυτό. Την πληροφορία αυτή κρατήσαμε και εισάγαμε στη βάση μας.

Πλέον έχει ολοκληρωθεί η εισαγωγή των πληροφοριών για τους Descriptors. Για τα insertions στη βάση δεδομένων χρησιμοποιήθηκαν parameterized queries για να μην υπάρχουν προβλήματα με τους ειδικούς χαρακτήρες, ενώ όπου υπήρχε η δυνατότητα χρησιμοποιήθηκε το batch για βελτίωση της απόδοσης του συστήματος.

4.2.2 Αντιστοίχιση MeSH όρων με άρθρα

Σε αυτό το σημείο μπορούμε πλέον να "συνδέσουμε" τη βάση με τα άρθρα με αυτή του MeSH. Λέγοντας σύνδεση εννοούμε τη δημιουργία πινάκων που θα δηλώνουν μία σχέση. Οι δύο σχέσεις που χρειάστηκαν ήταν : μία μεταξύ των άρθρων και των MeSH όρων και μία μεταξύ των λέξεων-κλειδί και των MeSH όρων. Η πρώτη έχει να κάνει με το πεδίο του άρθρου MeSH Headings, που περιέχει όλους τους descriptors και qualifiers και έχουν αποδωθεί στο άρθρο από ειδικούς indexers. Η δεύτερη από την άλλη έχει να κάνει με τα keywords και το κατά πόσο ταυτίζονται με κάποιο MeSH όρο. Τα keywords έχουν οριστεί από τους συγγραφείς, οι οποίοι όταν τα γράφουν δεν σκέφτονται το MeSH, με αποτέλεσμα να αρκετές φορές να μην αντιστοιχούν σε κάποιον όρο. Συνολικά τις σχέσεις αυτές τις θέλαμε για τη βελτίωση του συστήματός μας, ώστε η αναζήτηση και η επεξεργασία δεδομένων να γίνεται ευκολότερα και πιο γρήγορα, αλλά και την αξιοποίηση των συνωνύμων του MeSH.

Αρχικά λοιπόν αντιστοιχίσαμε κάθε άρθρο με όρους MeSH. Για αυτό το σκοπό χρειάστηκαν δύο πίνακες σχέσεων. Σε αφηρημένο επίπεδο είναι κάπως έτσι:

Σχήμα 4.3: Σύνδεση άρθρου και descriptor στη βάση δεδομένων.

Στο πρόγραμμά μας πήραμε από τη βάση δεδομένων τα άρθρα και για καθένα από αυτά ψάξαμε τους MeSH όρους για να βρούμε σε ποιους όρους αντιστοιχεί τελικά. Η πληροφορία όμως για τους MeSH όρους, που περιγράφουν το θέμα κάθε άρθρου, είναι αποθηκευμένη στο πεδίο MeshHeadings κάθε άρθρου σε μορφή JSON. Συνεπώς για να πάρουμε τα δεδομένα που θέλουμε, έπρεπε πρώτα να αναλύσουμε τα JSON σε Java objects ώστε να μπορέσουμε να τα χειριστούμε. Αυτό το πετύχαμε με τη χρήση της βιβλιοθήκης org.json. Στη συνέχεια ήταν δυνατόν πλέον να βρούμε τις αντιστοιχίσεις και να εισάγουμε εγγραφές στη βάση μας που δηλώνουν τις σχέσεις αυτές.

Στη συνέχεια ασχοληθήκαμε με την περίπτωση των keywords. Σε κάθε άρθρο, οι συγγραφείς ορίζουν κάποιες λέξεις-κλειδιά που θεωρούν οι ίδιοι ότι αποδίδουν το κύριο θέμα του άρθρου. Κάποιες από αυτές μπορεί να ταυτίζονται με MeSH

όρους ενώ κάποιες άλλες πάλι όχι. Η αντιστοίχιση όσων keywords ήταν δυνατό πραγματοποιήθηκε για τη διευκόλυνση μετέπειτας μελέτης. Η βάση έχει την εξής μορφή:

Σχήμα 4.4: Σύνδεση keyword και descriptor στη βάση δεδομένων.

Το πρόγραμμά μας εδώ ήταν ανάλογο με το προηγούμενο, μόνο που εδώ ήταν πιο απλά τα πράγματα μιας και τα δεδομένα δεν ήταν σε JSON μορφή. Συνεπώς συγκρίναμε όλα τα keywords με τους MeSH όρους και όπου υπήρχε αντιστοίχιση, δημιουργήσαμε μία εγγραφή που δηλώνει αυτή τη σχέση και την εισάγαμε στη βάση μας.

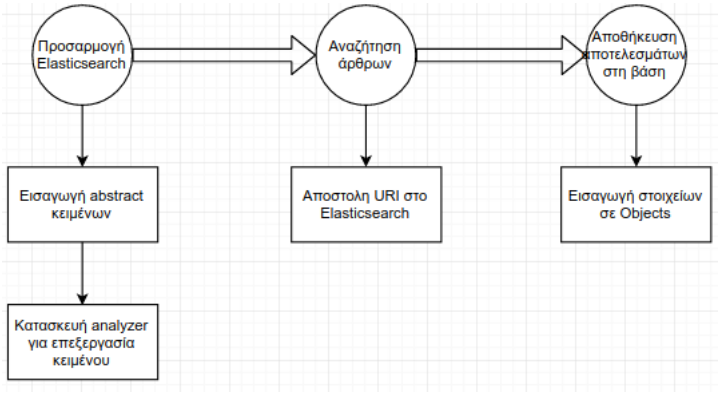
Ολοκληρώνοντας αυτήν την αντιστοίχιση είναι ευκολότερο πλέον να κάνουμε αναζητήσεις στη βάση μας. Για παράδειγμα μπορουμέ να βρούμε όλες τις λέξεις που σχετίζονται με ένα άρθρο, συμπεριλαμβανομένων και των συνώνυμων που φαίνονται κάπου άμεσα. Παράλληλα γλυτώνουμε και χρόνο, αφού πλέον δεν χρειάζεται για τέτοιες ενέργειες να ψάχνουμε ολόκληρη τη βάση μιας και υπάρχει ήδη η αντιστοίχιση.

4.3 Βελτιωμένη Αναζήτηση χρησιμοποιώντας το Elasticsearch

Έχουμε πλέον ολοκληρώσει το τμήμα του λογισμικού που αφορά τη βάση δεδομένων με άρθρα σχετικά με το σύνδρομο Sjogren. Για βελτιωμένη αναζήτηση χρησιμοποιήσαμε το Elasticsearch. Αρχικά εισάγαμε σε αυτό τα abstract κείμενα των άρθρων και κατασκευάσαμε analyzer για την επεξεργασία κειμένου, ώστε να προσαρμόσουμε το Elasticsearch στις απαιτήσεις. Στην συνέχεια μπορούμε να το χρησιμοποιήσουμε για να αναζητήσουμε άρθρα και να αποθηκεύσουμε κάποια αποτελέσματα στη βάση δεδομένων.

Είσοδος συστήματος: Τα abstract κείμενα των άρθρων, ο analyzer και οι όροι προς αναζήτηση.
Έξοδος συστήματος: Έτοιμο το Elasticsearch για αναζήτηση άρθρων και τα αποτελέσματα αποθηκευμένα στη βάση.

Στη συνέχεια μπορούμε να δούμε την διαδικασία που ακολουθήθηκε, ενώ παρακάτω υπάρχει η ανάλυση των επιμέρους βημάτων.



Σχήμα 4.5: Διαδικασία αναζήτησης με το Elasticsearch.

Έχουμε πλέον ολοκληρώσει το τμήμα του λογισμικού που αφορά τη βάση δεδομένων με άρθρα σχετικά με το σύνδρομο Sjogren. Ο τελικός στόχος, όπως ήδη έχουμε πει, είναι η βελτιωμένη αναζήτηση άρθρων. Για το λόγο αυτό δοκιμάστηκε το Elasticsearch, το οποίο ψάχνει στα abstract κείμενα των άρθρων. Το Elasticsearch όμως, προκειμένου να απαντά γρήγορα στα ερωτήματα, πρέπει να έχουν εισαχθεί τα δεδομένα σε αυτό και να έχουν δημιουργηθεί οι κατάλληλοι indexes. Στη συνέχεια πρέπει να ρυθμιστεί έτσι ώστε να προσαρμοστεί στις ανάγκες της αναζήτησης μας. Αυτό επιτεύχθηκε με τη χρήση analyzer, κατασκευασμένου βάση των απαιτήσεων μας.

4.3.1 Προσαρμογή Elastisearch στις απαιτήσεις

Το πρώτο βήμα ήταν να φορτώσουμε στο Elasticsearch τα abstract, στα οποία γίνεται μετέπειτα η αναζήτηση. Αυτό έγινε με τη χρήση του εργαλείου Logstash. Επειδή τα δεδομένα είναι σε sql βάση, η δομή τους δεν επιτρέπει άλλη άμεση επιλογή. Το Logstash παρέχει ένα ευέλικτο και δυνατό τρόπο επεξεργασίας τύπων δεδομένων με τη μορφή κειμένου. Έπρεπε λοιπόν να παραμετροποιηθεί το Logstash ώστε να διαβάζει από το input τη βάση μας και σαν output να τα προσλάβει το Elasticsearch. Η παραμετροποίηση αυτή έγινε με ένα config αρχείο, στο οποίο ορίζονται οι πρόσθετες λειτουργίες και ρυθμίσεις. Στη δική μας περίπτωση, δηλώσαμε στο input τη σύνδεση με τη βάση δεδομένων μας μαζί με όλα τα απαραίτητα στοιχεία. Αυτά ήταν τα στοιχεία σύνδεσης και οι βιβλιοθήκες και κλάσεις του JDBC driver. Από όλη τη βάση επιλέξαμε να εισάγουμε μόνο τα κείμενα των άρθρων (Abstract) εφόσον μόνο πάνω σε αυτά εκτελείται η αναζήτηση. Στο output δηλώσαμε το όνομα του index του Elasticsearch για τα δεδομένα μας, “abstracts”. Υπήρχαν και πρόσθετες πληροφορίες σχετικά με το Elasticsearch, όπως το το είδος αρχείων, το όνομα του αναγνωριστικού των αρχείων, καθώς και την τοποθεσία του (εδώ localhost), που βοηθάνε αργότερα και στην ανάκτηση των αποτελεσμάτων. Τρέξαμε λοιπόν το Logstash δηλώνοντας για αρχείο παραμετροποίησης αυτό που δημιουργήσαμε. Με την ολοκλήρωσή του έχουν εισαχθεί τα δεδομένα και έχει δημιουργηθεί το index.

Παρότι μπορούμε πλέον να χρησιμοποιήσουμε το Elasticsearch για την αναζήτηση στα abstracts, τα αποτελέσματα που θα επιστρέφονταν δεν θα ήταν τα επιθυμητά. Αυτό οφείλεται σε διάφορους παράγοντας. Πρώτα από όλα δεν θέλουμε να το χρησιμοποιήσουμε σαν μία απλή μηχανής αναζήτησης. Ο επιθυμητός στόχος είναι να το προσαρμόσουμε στις ανάγκες μας και συγκεκριμένα στα βιοϊατρικά δεδομένα του MeSH. Επίσης εφόσον μιλάμε για ελεύθερο κείμενο συνεπάγεται ότι σπάνια θα βρίσκαμε αυτούσιους

MeSH όρους μέσα στο κείμενο. Ακόμα και μία κατάληξη μπορεί να αλλοιώσει το αποτέλεσμα. Καταλήξαμε στο γεγονός ότι έπρεπε να καθορίσουμε τον τρόπο επεξεργασίας κειμένου στην αναζήτηση. Αυτό επιτεύχθηκε με analyzer.

Ο analyzer είναι ένα ενσωματομένο συστατικό του Elasticsearch που αναλαμβάνει την επεξεργασία κειμένου. Υπάρχουν κάποιοι προεπιλεγμένοι που καλύπτουν ένα εύρος αναγκών. Οι Standard, Simple και Language Analyzers είναι μερικοί από τους πιο διαδεδομένους. Καθένας έχει διαφορετικό τρόπο ανάλυσης. Για τη δική μας περίπτωση κατασκευάσαμε έναν προσαρμοσμένο analyzer. Πριν γίνει αυτό έπρεπε να αναλυθούν οι απαιτήσεις μας. Αξιοποιήσαμε το γεγονός ότι τα άρθρα είναι στα αγγλικά (ακόμα και αυτά που έχουν γραφτεί σε άλλη γλώσσα έχουν μεταφραστεί). Τα προβλήματα που έπρεπε να αντιμετωπιστούν ήταν τα ακόλουθα:

Καταλήξεις: Το ελεύθερο κείμενο έχει το αρνητικό (για την αναζήτηση) ότι μία λέξη χρησιμοποιείται με πολλές διαφορετικές καταλήξεις. Διαφορές ανάμεσα σε ενικό και πληθυντικό αριθμό ή ανάμεσα σε ρήμα και ουσιαστικό κάνουν τις λέξεις φαινομενικά διαφορετικές. Παρόλα αυτά έχουν το ίδιο θέμα, το βασικό κομμάτι της λέξης δηλαδή. Θέλουμε οι λέξεις με ίδιο θέμα να αντιμετωπίζονται ως ίδιες στην αναζήτηση. Ένα παράδειγμα είναι οι λέξεις "woman" και "women". Προφανώς είναι ίδιες αλλά με τη συμβατική μέθοδο σύγκρισης Strings θα θεωρούνταν διαφορετικές.

Stop words: Με τον όρο αυτό εννοούμε όλες αυτές τις συνδετικές λέξεις που ενώνουν τις προτάσεις και βοηθάνε στην ομαλή πορεία του κειμένου. Σύνδεσμοι, αντωνυμίες και βοηθητικά ρήματα ανήκουν σε αυτή την κατηγορία. Φυσικά αναφερόμαστε σε αγγλικές stop words. Παραδείγματα τέτοιων λέξεων είναι οι "you", "have", "and", "after", "will" και πολλές άλλες. Μπορεί να παίζουν πολύ σημαντικό ρόλο στη δομή του κειμένου, δεν προσφέρουν όμως τίποτα στο νοήμα. Θέλουμε λοιπόν στην αναζήτηση όλες αυτές οι λέξεις να αγνοούνται. Διαφορετικά επηρεάζονται τα αποτελέσματα της αναζήτησης.

Συνώνυμα: Όπως ήδη έχουμε πει, θέλουμε να προσαρμόσουμε το Elasticsearch στη δική μας περίπτωση με τα βιοϊατρικά δεδομένα. Αυτό μπορεί να επιτευχθεί με την αξιοποίηση του MeSH και πιο συγκεκριμένα της δομικής του μονάδας, του concept. Κάθε concept περιέχει διάφορες συνώνυμες λέξεις. Γράφονται διαφορετικά αλλά έχουν το ίδιο νόημα. Για παράδειγμα οι λέξεις "Cardiac Arrest" και "Heart Arrest" αναφέρονται στο ίδιο πράγμα. Συνεπώς, ο στόχος μας είναι το Elasticsearch να αντιμετωπίζει τις συνώνυμες λέξεις σαν ίδιες στην αναζήτηση.

Για την ικανοποίηση των παραπάνω απαιτήσεων κατασκευάσαμε λοιπόν τον analyzer "hmmy_analyzer". Κάνει stemming στις λέξεις για τις καταλήξεις, αφαιρεί τις stop words και κάνει χρήση των συνώνυμων του MeSH. Για τα δύο πρώτα υπάρχουν έτοιμες υλοποιήσεις στο Elasticsearch τις οποίες και προσαρμόσαμε στην περίπτωση μας. Για τα συνώνυμα χρησιμοποιήσαμε μία λίστα ειδικών κλινικών όρων για το σύνδρομο Sjogren, για τους οποίους βρήκαμε τους αντίστοιχους MeSH όρους και τα αντίστοιχα συνώνυμα. Για να συμβεί αυτό φτιάξαμε αρχικά έναν απλό πίνακα που περιείχε απλώς τα ονόματα των όρων. Στη συνέχεια γράφτηκε ένα πρόγραμμα το οποίο ανέλαβε να διαβάσει όλους τους όρους αυτούς και να βρει τα συνώνυμα κάθε όρου. Έχοντας πλέον και το τελευταίο κομμάτι του analyzer, το προσθέσαμε και στο τέλος δηλώσαμε στο Elasticsearch ότι πρόκειται για συνώνυμα. Με τον analyzer μας έτοιμο, το επομένο βήμα ήταν η ενημέρωση του Elasticsearch. Με ένα HTTP request με μέθοδο PUT αλλάξαμε τις ρυθμίσεις του και πλέον ήμασταν σε θέση να

χρησιμοποιήσουμε τον custom analyzer στην αναζήτησή μας.

4.3.2 Αναζήτηση με το Elasticsearch

Σε αυτό το σημείο έχουμε εισάγει τα abstract κείμενα των άρθρων στο Elasticsearch και έχουμε κατασκευάσει κατάλληλο analyzer για την επεξεργασία κειμένου στην αναζήτηση. Είμαστε πλέον σε θέση να αρχίσουμε την αναζήτηση. Προκειμένου να μπορούμε να χρησιμοποιήσουμε τα αποτελέσματα πολλές φορές για την εξαγωγή στατιστικών στοιχείων, τα αποθηκεύσαμε σε έναν πίνακα. Ο πίνακας αυτός ενώνει τα άρθρα με τους όρους που κάνουμε την αναζήτηση. Να σημειωθεί ότι το Elasticsearch βλέπει έναν όρο και τα συνώνυμά του σαν τον όρο αυτό. Συνεπώς αρκεί να "δείχνουμε" τον έναν αυτόν όρο κι να μην ασχοληθούμε περαιτέρω με τα συνώνυμα. Ο πίνακας λοιπόν, εκτός από τα στοιχεία για την αναγνώριση του άρθρου και του όρου, κρατάει και το score της αναζήτησης. Το score ενός αρχείου δείχνει το πόσο σχετικό είναι το αρχείο με το query. Όσο υψηλότερο το score τόσο πιο σχετικό είναι και το αρχείο. Το Elasticsearch έχει συγκεκριμένο αλγόριθμο που υπολογίζει αυτό το νούμερο και ονομάζεται "term frequency/inverse document frequency". Βασίζεται στη βιβλιοθήκη Lucene και ο τύπος υπολογισμού περιλαμβάνει πολλούς παράγοντες. Οι βασικότεροι από αυτούς είναι η συχνότητα εμφάνισης κάθε όρου στο αρχείο, η συχνότητα εμφάνισης του όρου στον index και ο παράγοντας κανονικοποίησης του ερωτήματος. Παρά τους πολλούς παράγοντες, ακόμα και υψηλό score δεν μπορεί να μας εγγυηθεί ότι το αποτέλεσμα είναι το επιθυμητό. Η σχετικότητα αυτή στηρίζεται στο κατά πόσο μοιάζουν τα κείμενα. Ο κάθε χρήστης μπορεί να θέλει κάτι διαφορετικό εν τέλει.

Όσον αφορά την αναζήτηση των άρθρων τώρα, έγινε με κατάλληλα URI. Συγκεκριμένα στείλαμε search request με κατάλληλες παραμέτρους στο URI. Μπορεί να μην υποστηρίζονται όλες οι επιλογές με αυτή τη μέθοδο όμως οι βασικές που υπάρχουν αρκούν στην περίπτωσή μας. Το URI είχε την ακόλουθη δομή :

http://localhost:9200/abstracts/_search?q=abstracttext:
<query_string>&analyzer=hmmy_analyzer&size=5000

Αρχικά επιλέξαμε σε ποιο index να ψάξει το Elasticsearch. Αυτό ήταν το "abstracts", όπως το είχαμε δηλώσει με το Logstash. Έπειτα δηλώσαμε τις παραμέτρους αναζήτησης. Πρώτα το κείμενο προς αναζήτηση με την παράμετρο q. Συγκεκριμένα να ψάξει στο πεδίο abstractext που βρίσκεται το abstract κείμενο του άρθρου. Μετά θέσαμε τον "hmmy_analyzer" που έχουμε κατασκευάσει για να γίνεται η αναζήτηση προσαρμοσμένη στο δικό μας πλαίσιο. Τέλος, επειδή το Elasticsearch επιστρέφει default μόνο 10 αποτελέσματα, βάλαμε την παράμετρο size ίση με 5000 ώστε να παίρνουμε όλα τα άρθρα που εντοπίζονται.

Στο πρόγραμμά μας λοιπόν διαβάσαμε αρχικά όλους τους όρους που έχουμε επιλέξει για να κάνουμε τη σύγκριση αργότερα. Για κάθε όρο λοιπόν, στείλαμε ένα search request στο Elasticsearch για να ψάξει σε όλα τα άρθρα τα abstract κείμενα. Με το URI Search επιστρέφονται τα αποτελέσματα σε πραγματικό χρόνο. Είναι σε JSON μορφή οπότε χρειάζεται η βιβλιοθήκη org.json για να ανακτήσουμε τα δεδομένα που χρειαζόμαστε. Αυτά που θέλαμε να αποθηκεύσουμε ήταν το PMID του άρθρου και το score που έχει το συγκεκριμένο αρχείο για τα στατιστικά αργότερα. Για κάθε άρθρο βάλαμε αυτές τις πληροφορίες, μαζί με το id του όρου που αναζητάγαμε, στη βάση μας

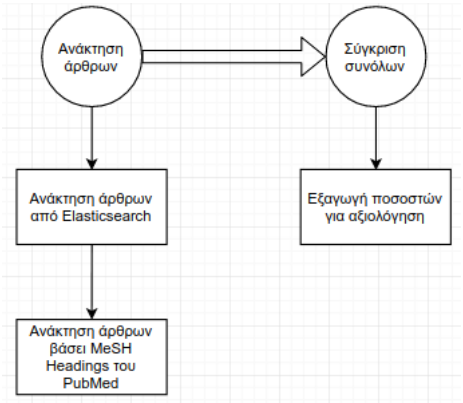
σύμφωνα με τον πίνακα που έχουμε φτιάξει. Η διαδικασία επαναλήφθηκε για όλους τους όρους. Στο τέλος έχουμε στη βάση δεδομένων μας όλα τα στοιχεία εκείνα που αντιστοιχούν τα άρθρα στην λίστα με τους όρους για σύγκριση.

4.4 Αξιολόγηση Αποτελεσμάτων

Στο σημείο αυτό έχουμε διαθέσιμα τα δεδομένα των άρθρων από το PubMed αλλά και το Elasticsearch για αναζήτηση. Για την αξιολόγηση των αποτελεσμάτων του Elasticsearch κατασκευάστηκε ένα σύστημα που για κάποιους κλινικούς όρους επιστρέφει τα άρθρα που προκύπτουν από την αναζήτηση με το Elasticsearch και εκεινά που περιέχουν τους όρους αυτούς στα MeSH Headings. Στην συνέχεια συγκρίθηκαν τα δύο αυτά σύνολα και προέκυψαν κάποια στατιστικά.

Είσοδος συστήματος: Τα άρθρα από το Elasticsearch και από το PubMed.
Έξοδος συστήματος: Ποσοστά από την σύγκριση των συνόλων των άρθρων.

Η διαδικασία αξιολόγησης των αποτελεσμάτων φαίνεται στο παρακάτω σχήμα. Πιο αναλυτικά εξηγείται στη συνέχεια.



Σχήμα 4.6: Διαδικασία αξιολόγησης αποτελεσμάτων.

Για την αξιολόγηση των αποτελεσμάτων του Elasticsearch αποφασίστηκε να συγκριθεί κατά κύριο λόγο με τα MeSH Headings που έχουν αποδωθεί στα άρθρα και συγκεκριμένα με τους descriptors. Οι descriptors περιγράφουν το κύριο θέμα ενός άρθρου. Συνεπώς είναι αρκετά πιθανό να υπάρχει ο όρος που περιγράφει το άρθρο μέσα στο abstract κείμενό του. Αυτή η σύγκριση δεν είναι απόλυτη αλλά μπορεί να μας δείξει κατά πόσο αποτελεσματική ήταν η αναζήτηση με το Elasticsearch και αν προέκυψε κάποια επιπλέον πληροφορία.

Για κάθε κλινικό όρο λοιπόν που έχουμε επιλέξει, συγκρίναμε τα άρθρα που επιστρέφονται από το Elasticsearch με αυτά που αντιστοιχούν στα MeSH Headings των άρθρων και έχουν ήδη αποδωθεί από το PubMed. Βέβαια, όπως ήδη έχουμε πει, κάθε αποτέλεσμα που επιστρέφεται έχει συγκεκριμένο score. Οπότε η σύγκριση έγινε για συγκεκριμένα διαστήματα εύρους του score. Το πρόγραμμα λοιπόν που αναλαμβάνει αυτή την εργασία στηρίχτηκε στα HashSet. Πρόκειται για σύνολα αντικειμένων, στα οποία υποστηρίζονται λειτουργίες που διευκολύνουν τη σύγκριση μεταξύ συνόλων, όπως εύρυση κοινών ή διαφορετικών αντικειμένων. Συγκεκριμένα, διαβάστηκαν όλοι οι όροι και για κάθε όρο κρατήθηκαν σε HashSet όλα τα PMID των

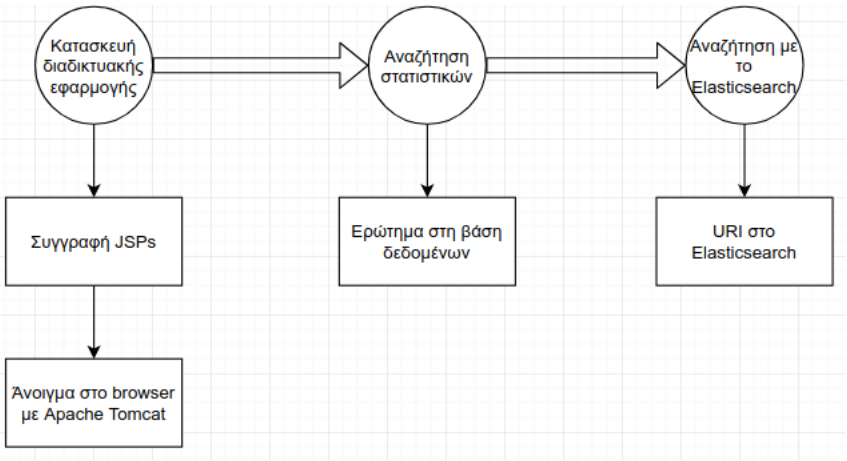
άρθρων που προέκυψαν από την αναζήτηση του Elasticsearch. Στη συνέχεια για κάθε όρο βρήκαμε τον αντίστοιχο MeSH όρο και εντοπίσαμε στη βάση τα PMID των άρθρων που περιείχαν τον όρο αυτό ή κάποιο συνώνυμό του (να άνηκαν στο ίδιο concept δηλαδή). Έχοντας τα δύο αυτά σύνολα, τα συγκρίναμε βλέποντας ποια ήταν κοινά και τι παραπάνω PMID είχε το κάθε σύνολο. Στο τέλος εμφανίστηκαν αθροιστικά τα στοιχεία αυτά για το εκάστοτε εύρος τιμών score. Συγκεκριμένα προκύπτει ο συνολικός αριθμός των άρθρων που επιστράφηκαν από το Elasticsearch και από το MeSH, καθώς και τα αντίστοιχα ποσοστά ταιριάσματος για κάθε μέθοδο, δηλαδή πόσα άρθρα από το Elasticsearch επιστρέφονται και από το MeSH και το αντίστροφο. Τα αποτελέσματα της αξιολόγησης παρουσιάζονται στο 5ο κεφάλαιο.

4.5 Διαδικτυακή Εφαρμογή

Για την παρουσίαση στατιστικών των άρθρων αλλά και τη δυνατότητα αναζήτησης με το Elasticsearch κατασκευάσαμε μία διαδικτυακή εφαρμογή. Αυτή αποτελείται από μία συλλογή JSPs αρχείων, που επικοινωνούν τόσο μεταξύ τους όσο και με τη βάση δεδομένων και το Elasticsearch. Μπορεί να χρησιμοποιηθεί από κάποιον browser με τη βοήθεια του Apache Tomcat.

Είσοδος συστήματος: JSP αρχεία και τα πεδία αναζήτησης.
Έξοδος συστήματος: Σελίδα με τα ζητούμενα αποτελέσματα.

Η διαδικασία για την κατασκευή και χρήση της εφαρμογής φαίνεται στο παρακάτω σχήμα. Η δομή και οι λειτουργίες της φαίνεται στη συνέχεια.



Σχήμα 4.7: Διαδικασία κατασκευής και χρήσης διαδικτυακής εφαρμογής.

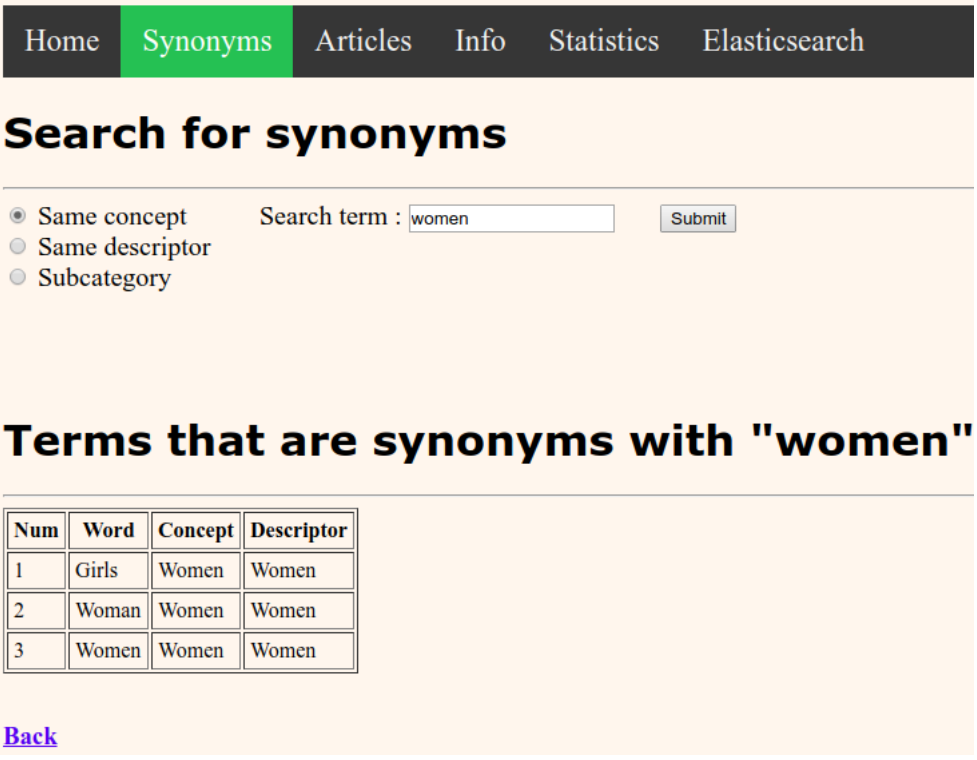
Όπως ήδη αναφέραμε, η διαδικτυακή εφαρμογή αποτελείται από JSP σελίδες. Αυτές περιλαμβάνουν τον κώδικα και αφορούν το λειτουργικό κομμάτι. Το τελικό αποτέλεσμα διαμορφώνεται με τη βοήθεια CSS, για το οπτικό μέρος. "Τρέχει" στο browser localhost με τη βοήθεια του Apache Tomcat. Να τονιστεί ότι τα στατιστικά προκύπτουν από τη MeSH βάση συνδιαστικά με τα άρθρα του PubMed που σχετίζονται με το Sjogren, ενώ για την αναζήτηση με το Elasticsearch υπάρχει άμεση επικοινωνία μαζί του.

Η πλατφόρμα λοιπόν αποτελείται από 6 βασικά τμήματα-καρτέλες (tabs). Ονομαστικά αυτά είναι : Home, Synonyms, Articles, Info, Statistics, Elasticsearch. Το πρώτο

απλώς παρουσιάζει κάποιες πληροφορίες ενώ στα υπόλοιπα συμμετέχει και ο χρήστης με τη δυνατότητα αναζήτησης. Κάθε tab κατασκευάστηκε και είναι ανεξάρτητο από τα υπόλοιπα. Μέσω του μενού στο πάνω μέρος γίνεται η πλοήγηση μεταξύ των τμημάτων. Για όλες τις σελίδες υπάρχει συγκεκριμένο στιλ, το οποίο καθορίζεται από το CSS που έχει οριστεί και δηλώνεται παντού. Επίσης, κάποια tabs μπορεί να περιέχουν περισσότερες από μία σελίδες αλλά συνολικά όλες ανήκουν στο ίδιο. Τα αποτελέσματα που εμφανίζονται ανακτώνται δυναμικά από τη βάση δεδομένων με SQL queries. Η σύνδεση στη βάση γίνεται εύκολα με τη χρήση του JDBC μιας και τα JSP στηρίζονται στη Java. Παρακάτω αναλύεται κάθε tab ξεχωριστά.

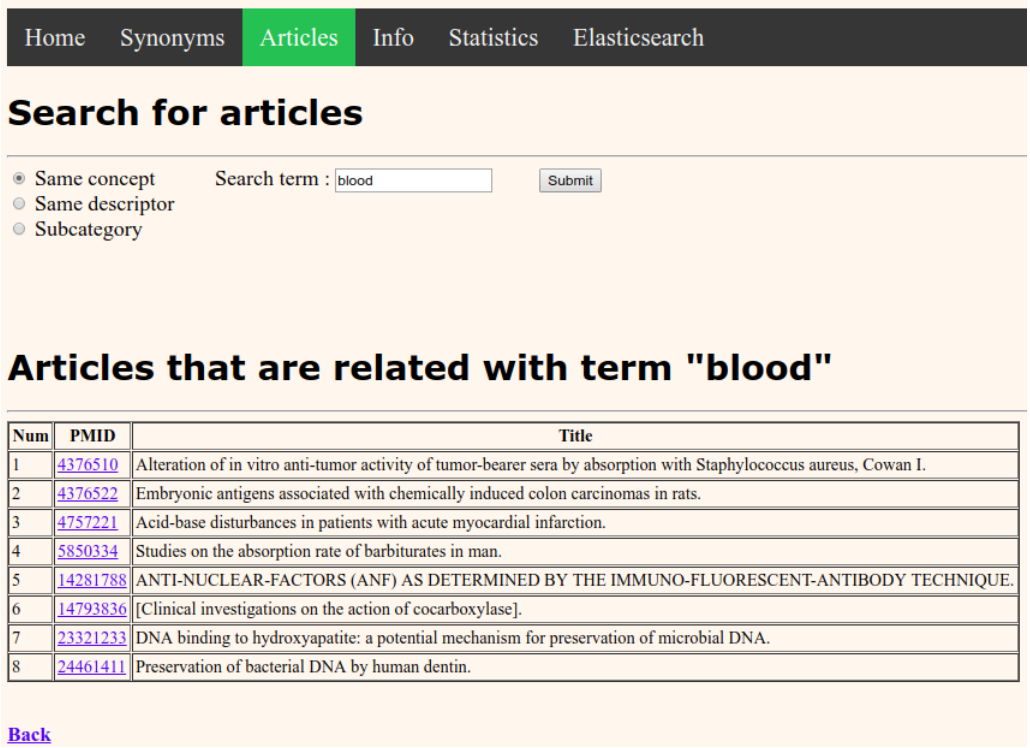
Home: Πρόκειται για την αρχική σελίδα, η οποία είναι και η πιο απλή. Στη σελίδα αυτή εμφανίζονται κάποια συγκεντρωτικά στατιστικά που αφορούν το πλήθος κάποιων στοιχείων, όπως π.χ. των συγγραφέων. Είναι η μοναδική η οποία δεν αλληλεπιδρά με το χρήστη.

Synonyms: Η δεύτερη σελίδα έχει να κάνει με την εύρεση συνώνυμων όρων στο MeSH κυρίως. Πέραν όμως αυτού, δίνεται η δυνατότητα εύρεσης όρων που ανήκουν στον ίδιο descriptor. Δεν είναι ακριβώς συνώνυμα αλλά βρίσκονται στην ίδια ευρύτερη κατηγορία. Η τρίτη επιλογή μάς δίνει τη δυνατότητα να αξιοποιήσουμε την ιεραρχική δομή του MeSH και την κληρονομικότητα. Συγκεκριμένα επιστρέφονται όλοι οι όροι, οι οποίοι ανήκουν σε descriptor "κάτω" από αυτόν του αναζητούμεου όρου (δηλαδή εκείνοι που είναι πιο ειδικοί). Αφού ο χρήστης επιλέξει μία από τις τρεις κατηγορίες αναζήτησης και πληκτρολογήσει τον όρο που θέλει, επιστρέφονται τα αποτελέσματα, οι συνώνυμοι όροι καθώς και σε ποιο concept και descriptor ανήκει ο καθένας. Στο παρακάτω παράδειγμα έχει επιλεχθεί η πρώτη κατηγορία και ο όρος αναζήτησης είναι "women":



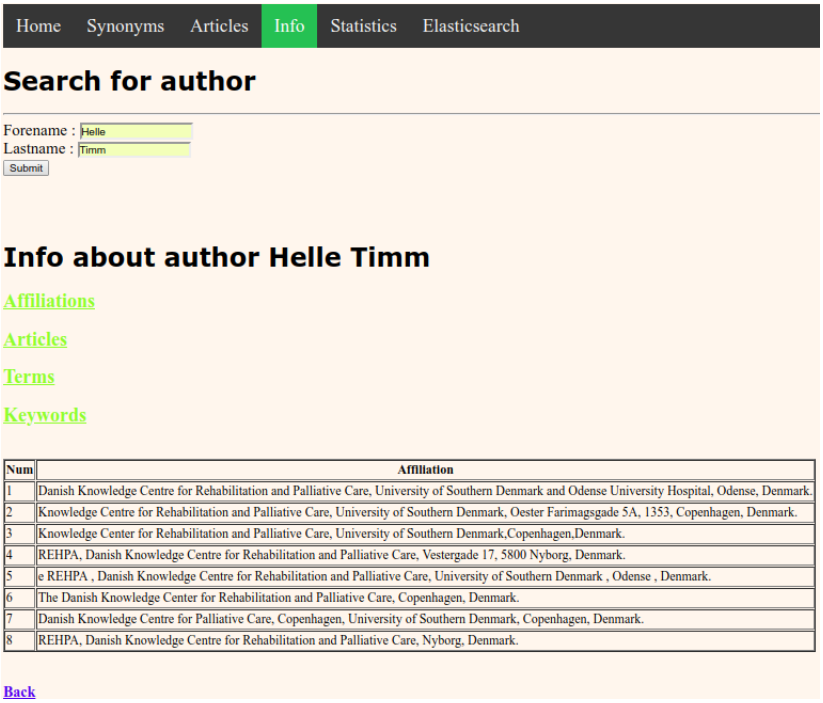
Σχήμα 4.8: Παράδειγμα αναζήτησης συνωνύμων του όρου "women".

Articles: Στο tab αυτό οι επιλογές είναι ίδιες με προηγουμένως και οι λειτουργίες ανάλογες. Πιο συγκεκριμένα, όπως και πριν υπάρχουν τρεις επιλογές : μία για συνώνυμα (ίδιου concept), μία για σχεδόν συνώνυμα (ίδιου descriptor) και μία για όρους πιο ειδικούς (subcategory). Εδώ όμως δεν θα επιστρέφονται λέξεις-όροι. Θα επιστρέφονται όλα εκείνα τα άρθρα που σχετίζονται με τους αντίστοιχους αυτούς όρους. Μετά την επιλογή κατηγορίας και πληκτρολόγηση όρου, επιστρέφονται τα αποτελέσματα. Η παρακάτω εικόνα παρουσιάζει τα αποτελέσματα αναζήτησης για τον όρο "blood" και κατηγορία "same descriptor". Όπως βλέπουμε εμφανίζεται τόσο ο τίτλος όσο και το PMID, για το οποίο υπάρχει και η δυνατότητα ανακατεύθυνσης στο xml αρχείο του συγκεκριμένου άρθρου.



Σχήμα 4.9: Παράδειγμα αναζήτησης άρθρων σχετικών με τον όρο "blood" και τα συνώνυμά του.

Info: Η καρτέλα αυτή δεν περιέχει στατιστικά στοιχεία, όπως όλες οι υπόλοιπες, αλλά πληροφορίες. Ειδικότερα, παρέχει τη δυνατήτοτα στους χρήστες να αναζητήσουν πληροφορίες για κάτι πιο συγκεκριμένο. Οι επιλογές που δίνονται είναι τρεις: συγγραφείς (Author), άρθρα (Article) και οργανισμοί (Affiliation). Από το αρχικό μενού αρχικά επιλέγεται μία από τις τρεις ομάδες και γίνεται ανακατεύθυνση σε επόμενη σελίδα για εισαγωγή των στοιχείων προς αναζήτησης. Τυχαία επιλέγουμε να αναζητήσουμε πληροφορίες για το συγγραφέα "Helle Timm". Στη συνέχεια μπορούμε να επιλέξουμε τι πληροφορία θέλουμε να μάθουμε για το συγκρεκριμένο συγγραφέα. Οι εναλλακτικές είναι οι εξής: οι οργανισμοί στους οποίους ανήκει ο συγγραφέας (Affiliations), τα άρθρα τα οποία έχει γράψει ο συγγραφέας (Articles), οι MeSH όροι που περιέχουν τα άρθρα του (Terms), καθώς και οι λέξεις-κλειδιά που έχει αποδώσει ο ίδιος ο συγγραφέας (Keywords). Ανάλογα με την επιλογή μας τα αποτελέσματα φορτώνονται στην ίδια σελίδα. Ενδεικτικά επιλέξαμε τα Affiliations:

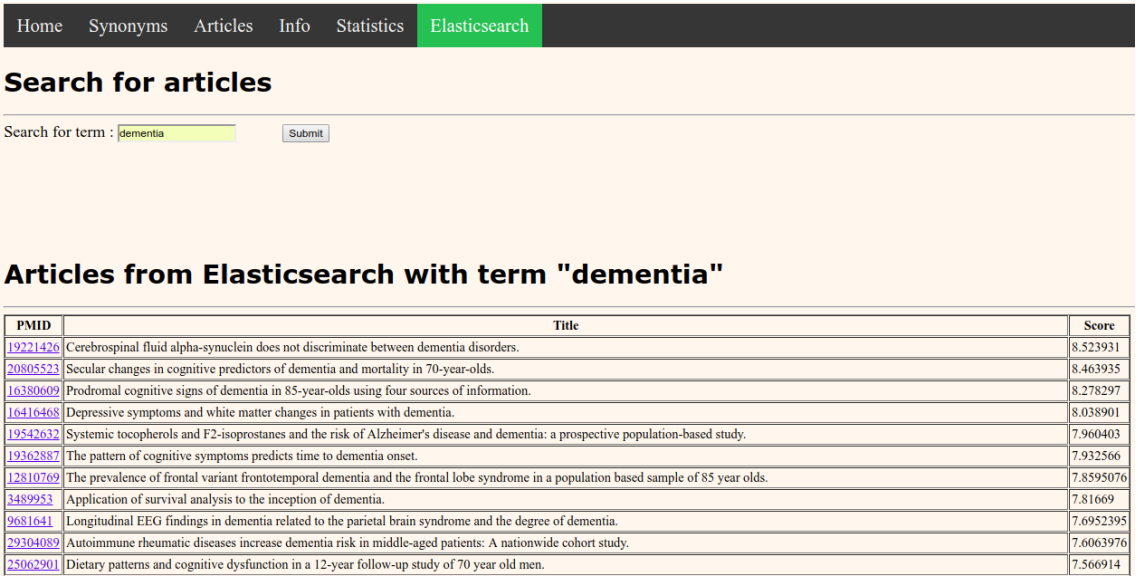


Σχήμα 4.10: Παράδειγμα αναζήτησης συγγραφέα και των οργανισμών που ανήκει.

Στις παραπάνω εικόνες επιλέξαμε να δείξουμε ένα τυχαίο παράδειγμα για έναν συγγραφέα και τους οργανισμούς που ανήκει. Όπως προαναφέραμε υπάρχουν και επιλογές για αναζήτηση οργανισμών και άρθρων. Αν επιλέξουμε άρθρα (Article), οι πληροφορίες που μπορούμε να βρούμε αφορούν τους συγγραφείς που έχουν γράψει αυτό το άρθρο, τους οργανισμούς τους, τους MeSH όρους που έχουν αποδωθεί και τις λέξεις-κλειδιά που έχουν αποδώσει οι συγγραφείς σε αυτό. Εάν επιλεγούν οι οργανισμοί (Affiliation), τότε μπορούμε να βρούμε τους συγγραφείς που ανήκουν στον οργανισμό αυτό, τα άρθρα που του αντιστοιχούν, καθώς και τους MeSH όρους και λέξεις-κλειδιά που περιέχουν.

Statistics: Στην καρτέλα των στατιστικών βρίσκονται συγκεντρωμένα κάποια στατιστικά. Είναι σχετικά τόσο με τα άρθρα σχετικά με το Sjogren όσο και συνδιαστικά με το MeSH. Κάποια από αυτά Θα παρουσιαστούν στο επόμενο κεφάλαιο.

Elasticsearch: Στην τελευταία καρτέλα δίνεται η δυνατότητα αναζήτησης στο Elasticsearch. Ο χρήστης πληκτρολογεί τον όρο που θέλει να αναζητήσει χρησιμοποιώντας το Elasticsearch. Τα αποτελέσματα που του επιστρέφονται είναι οι τίτλοι των ζητούμενων άρθρων, τα PMIDs τους, για τα οποία υπάρχει και η δυνατότητα ανακατεύθυνσης στα xml αρχεία και τέλος τα score του κάθε αρχείου. Στο παράδειγμα αναζητήσαμε τον όρο “dementia” και παρουσιάζονται τα κορυφαία αποτελέσματα:



Σχήμα 4.11: Παράδειγμα αναζήτησης όρου ”dementia” στο Elasticsearch.

Κεφάλαιο 5

Αποτελέσματα και Σχολιασμός

5.1 Περιεχόμενα Βάσης

Για την καλύτερη κατανόηση του περιεχομένου της βάσης και της πληροφορίας που υπάρχει γύρω από το σύνδρομο Sjogren βγάλαμε κάποια στατιστικά στοιχεία σχετικά με το πλήθος των εγγραφών κάθε πίνακα, τους όρους που χρησιμοποιούνται, τις αναλογίες πλήθους στοιχείων ανά άρθρο, καθώς και τα πιο χρησιμοποιούμενα στοιχεία. Τα περισσότερα από αυτά τα στατιστικά στοιχεία είναι επίσης διαθέσιμα και μέσω της διαδικτυακής εφαρμογής που αναπτύχθηκε. Στον πρώτο πίνακα φαίνεται το πλήθος των εγγραφών των πινάκων.

Στοιχείο βάσης	**Πλήθος**
Articles	6417
Terms	116706
Authors	22954
Affiliations	9507
Keywords	4558
Concepts	55533
Descriptors	28939

Πίνακας 5.1: Πλήθος στοιχείων της βάσης δεδομένων.

Στην παρακάτω εικόνα βλέπουμε τα ποσοστά χρησιμοποίησης των MeSH στοιχείων στα άρθρα.

Είδος	**Ποσοστό**
MeSH όροι που χρησιμοποιούνται	7520/116706 (6%)
Concepts που χρησιμοποιούνται	7516/55533 (13%)
Descriptors που χρησιμοποιούνται	7516/28939 (25%)
Keywords που αντιστοιχούν σε MeSH όρους	1470/4558 (32%)

Πίνακας 5.2: Στατιστικά στοιχεία της βάσης δεδομένων.

Στην συνέχεια παρουσιάζονται οι 10 πιο συχνά χρησιμοποιούμενοι, στα άρθρα, MeSH όροι και λέξεις-κλειδιά.

MeSH όρος	**Πλήθος εμφανίσεων**
Humans	5121
Female	3218
Male	2500
Sjogren's Syndrome	2250
Middle Aged	2062
Adult	1861
Aged	1448
Animals	879
Adolescent	514
Sjogren-Larsson Syndrome	450

Πίνακας 5.3: Οι 10 κορυφαίοι MeSH όροι.

Λέξη-κλειδί	**Πλήθος εμφανίσεων**
SJOGREN'S SYNDROME	534
Sjögren syndrome	190
Systemic lupus erythematosus	80
Autoimmunity	64
Rheumatoid arthritis	63
Primary Sjögren's syndrome	60
Autoimmune diseases	51
Autoantibodies	46
Salivary glands	42
Sjøgren's Syndrome	41

Πίνακας 5.4: Οι 10 κορυφαίες λέξεις-κλειδιά.

Κάποια πρόσθετα στατιστικά στοιχεία είναι οι MeSH όροι, οι λέξεις-κλειδιά και οι συγγραφείς που υπάρχουν ανά άρθρο και οι οργανισμοί που αντιστοιχούν σε κάθε συγγραφέα.

Είδος	**Αναλογία**
MeSH όροι ανά άρθρο	$66834/6417 = 10.4$
Keywords ανά άρθρο	$8447/6417 = 1.3$
Συγγραφείς ανά άρθρο	$36667/6417 = 5.7$
Οργανισμοί ανά συγγραφέα	$16038/22954 = 0.7$

Πίνακας 5.5: Επιπλέον στατιστικά στοιχεία των άρθρων.

Στο σημείο αυτό προσπαθήσαμε να εντοπίσουμε κάποιο μοτίβο των MeSH όρων στα άρθρα. Ασχοληθούμε με τους πιο συχνά χρησιμοποιούμενους όρους. Στα άρθρα που έχουμε υπάρχουν 7516 διαφορετικοί όροι. Αυτοί εμφανίζονται συνολικά 66834 φορές. Ο αριθμός των άρθρων είναι 6417. Όπως ήδη έχουμε αναφέρει, οι MeSH αυτοί όροι δείχνουν το κύριο θέμα ενός άρθρου. Παρατηρώντας τα νούμερα του πίνακα με τους 10 κορυφαίους MeSH όρους βλέπουμε με μία πρόχειρη ματιά ότι τα περισσότερα

άρθρα αφορούν άτομα μεγάλης ηλικίας και κατα κύριο λόγο γυναίκες. Συνεχίζοντας την ανάλυση προκύπτει ότι όποιο άρθρο έχει θέμα που σχετίζεται με τον άνθρωπο, τότε θα μιλάμε για άτομο μέσης ηλικίας. Συγκεκριμένα για τους όρους "Humans", "Female" και "Male" οι όροι που εμφανίζονται συχνότερα με αυτούς είναι οι παρακάτω, μαζί με τα αντίστοιχα ποσοστά εμφάνισής τους:

MeSH όρος	**Humans**	**Female**	**Male**
Middle Aged	40%	58%	61%
Adult	36%	50%	55%

Πίνακας 5.6: Εύρεση pattern MeSH όρων (1).

Μέχρι στιγμής λοιπόν, από τα στατιστικά καταλαβαίνουμε ότι το σύνδρομο Sjogren εμφανίζεται κατά κύριο λόγο σε γυναίκες μέσης ηλικίας.

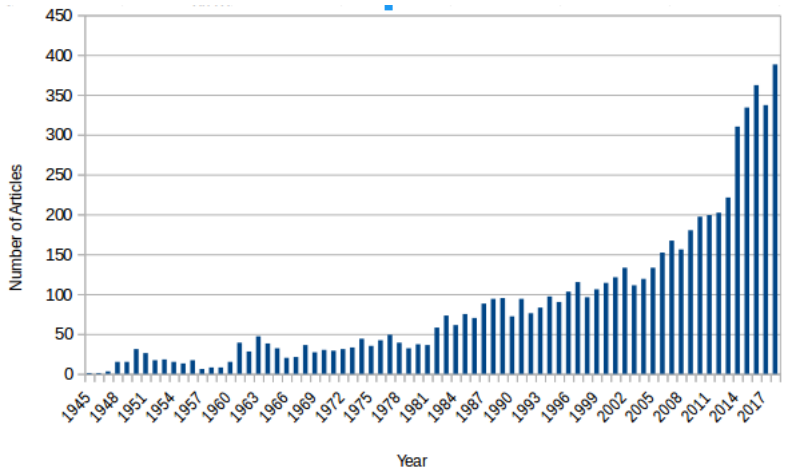
Όσον αφορά το ποσοστό του όρου "Animals" (14%) προφανώς αναφέρεται σε άρθρα που σχετίζονται με πειράματα σε ζώα. Περαιτέρω ανάλυση δείχνει ότι το μεγαλύτερο νούμερο αφορά ποντίκια και αρουραίους μιας και αυτοί οι όροι εμφανίζονται με μεγαλύτερο ποσοστό μαζί με τον όρο "Animals".

MeSH όρος	**Animals**
Mice	35%
Rats	29%

Πίνακας 5.7: Εύρεση pattern MeSH όρων (2).

Για να αποκτήσουμε μία σφαιρική άποψη γύρω από τα άρθρα του PubMed σχετικά με το Sjogren κατασκευάστηκαν και κάποιες γραφικές παράστασεις. Τα στοιχεία στα οποία στηριχτήκαμε είναι τα άρθρα, οι συγγραφείς, οι λέξεις-κλειδιά και οι MeSH όροι.

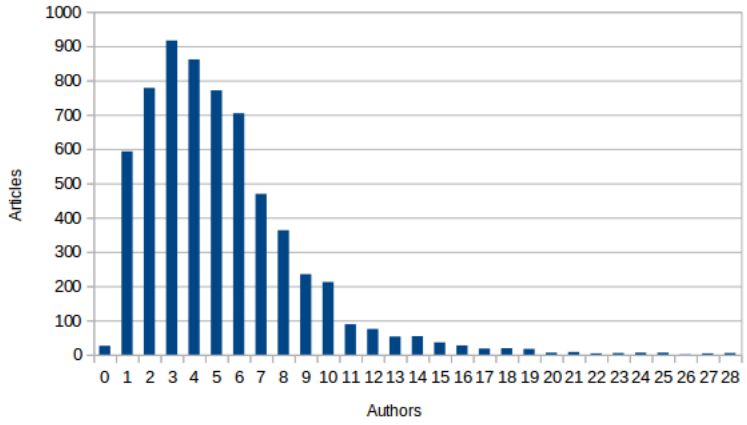
- Αριθμός άρθρων που δημοσιεύθηκαν ανά έτος.



Σχήμα 5.1: Γράφημα αριθμού άρθρων ανά έτος.

Παρατηρούμε μία σταδιακή αύξηση των άρθρων που δημοσιεύονται για το σύνδρομο Sjogren κάθε χρόνο.

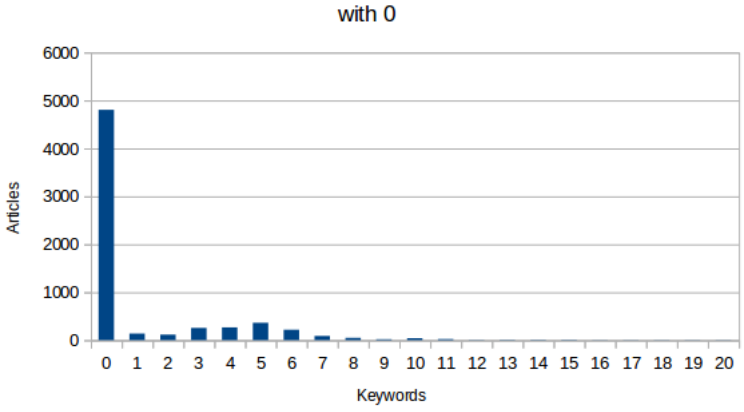
- Αριθμός άρθρων ανά πλήθος συγγραφέων που έχουν συμμετάσχει στη δημιουργία του άρθρου.



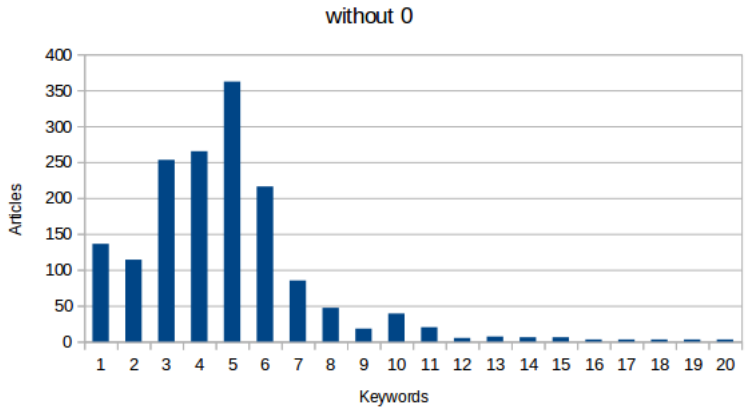
Σχήμα 5.2: Γράφημα αριθμού άρθρων ανά πλήθος συγγραφέων που έχουν συμμετοχή στη δημιουργία του άρθρου.

Βλέπουμε ότι τα περισσότερα άρθρα έχουν μέχρι 10 συγγραφείς, με το πιο σύνηθες να υπάρχουν 3 συγγραφείς.

- Αριθμός άρθρων ανά πλήθος λέξεων-κλειδί (keyword) που έχουν αποδώσει οι συγγραφείς στο εκάστοτε άρθρο. Υπάρχουν δύο γραφικές με στήλες, μία που αγνοεί την περίπτωση κανενός keyword στο άρθρο και μία όχι, για να διακρίνουμε καλύτερα τις διαφορές.



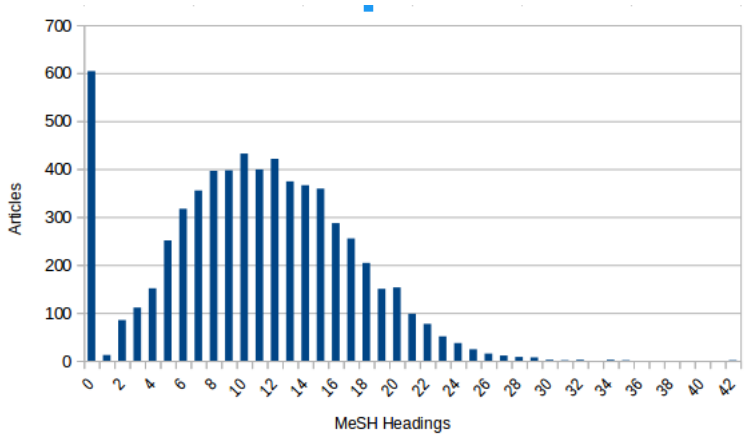
Σχήμα 5.3: Γράφημα αριθμού άρθρων ανά πλήθος keywords (υπολογίζεται το 0).



Σχήμα 5.4: Γράφημα αριθμού άρθρων ανά πλήθος keywords (χωρίς να υπολογίζεται το 0).

Συνολικά παρατηρούμε ότι τα περισσότερα άρθρα δεν έχουν καμία λέξη-κλειδί. Πέρα από αυτό τα περισσότερα έχουν έως 7 keywords, με το 5 το πιο συχνό.

- Αριθμός άρθρων ανά πλήθος MeSH όρων που έχουν αποδωθεί στο εκάστοτε άρθρο.



Σχήμα 5.5: Γράφημα αριθμού άρθρων ανά πλήθος MeSH όρων.

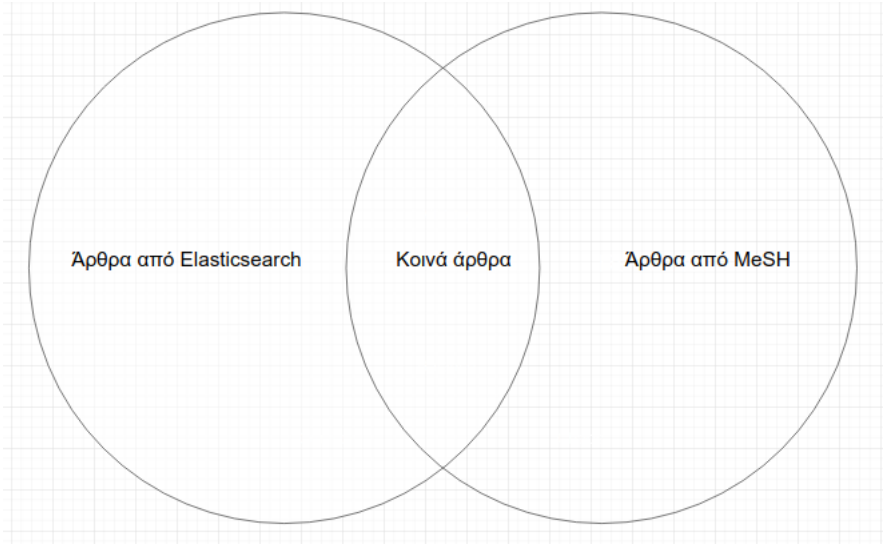
Βλέπουμε ότι πολλά άρθρα δεν έχουν MeSH Headings, αν και όχι στο βαθμό με τα Keywords προηγουμένως. Παρατηρούμε ότι το μεγαλύτερο ποσοστό των άρθρων έχουν μεταξύ 2 και 24 MeSH Headings, με τις πιο συχνές τιμές ανάμεσα στο 8 και το 12.

5.2 Αποτελέσματα Elasticsearch

Όπως έχουμε ήδη αναφέρει στο κεφάλαιο 4, το Elasticsearch έχει αποδώσει σε κάθε άρθρο ένα συγκεκριμένο score για καθένα όρο, που δείχνει πόσο σχετικό είναι το άρθρο με τον όρο αυτόν. Όσο υψηλότερο το score τόσο πιο σχετικό είναι και το άρθρο. Για να εντοπίσουμε ποιες τιμές του score επιστρέφουν τα καλύτερα αποτελέσματα δοκιμάσαμε πολλά εύρη τιμών με διαφορετικό μήκος και διαφορετικές περιοχές τιμών.

Έτσι μπορέσαμε να δούμε για ποιες τιμές του score μπορούμε να θεωρήσουμε ότι ο όρος που θέλουμε υπάρχει στο κείμενο.

Για να αξιολογήσουμε τα αποτελέσματα συγκρίναμε τα άρθρα που επιστρέφει το Elasticsearch με αυτά που παίρνουμε με βάση το MeSH. Συγκεκριμένα βρήκαμε πόσα άρθρα επιστρέφει το Elasticsearch, σε πόσα άρθρα βρίσκεται ο όρος στα MeSH Headings και πόσα συνολικά από αυτά τα άρθρα είναι κοινά. Σχηματικά η σχέση των δύο συνόλων που θέλουμε να συγκρίνουμε φαίνεται από το διάγραμμα Venn.



Σχήμα 5.6: Venn διάγραμμα συνόλων σύγκρισης.

Συνολικά υπάρχουν 4 πίνακες: στον πρώτο τα αποτελέσματα είναι οργανωμένα σε ομάδες με εύρος score 2, στον δεύτερο με εύρος 5, στον τρίτο με 10 και στον τέταρτο οι ομάδες είναι επικαλυπτόμενες. Στην πρώτη στήλη φαίνεται το εύρος τιμών του score. Στην δεύτερη είναι ο αριθμός των άρθρων που προέκυψαν από το Elasticsearch για το συγκεκριμένο εύρος score, ενώ ο αντίστοιχος αριθμός για το MeSH βρίσκεται στην τέταρτη στήλη. Στις στήλες 3 και 5 φαίνονται τα ποσοστά πόσων άρθρων του Elasticsearch έχουν τον όρο και στα MeSH Headings και, αντίστοιχα, τα ποσοστά των άρθρων που περιέχουν τον όρο στα MeSH Headings αλλά έχουν επιστραφεί και από το Elasticsearch.

Score	**Articles of Elasticsearch**	**Matched (%)**	**Articles of MeSH**	**Matched (%)**
[0, 2)	0	0%	0	0%
[2, 4)	114	13%	185	8%
[4, 6)	553	13%	320	24%
[6, 8)	178	40%	391	18%
[8, 10)	65	41%	322	8%
[10, 12)	11	63%	97	7%
[12, 14)	7	28%	47	4%
[14, 16)	0	0%	0	0%
[16, 18)	1	100%	11	9%
[18, 20)	0	0%	0	0%
[20, $+\infty$)	0	0%	0	0%

Πίνακας 5.8: Αποτελέσματα αναζήτησης για τιμές του score εύρους 2.

Score	Articles of Elasticsearch	Matched (%)	Articles of MeSH	Matched (%)
[0, 5)	400	12%	276	17%
[5, 10)	556	25%	438	32%
[10, 15)	26	34%	133	6%
[15, 20)	1	100%	11	9%
[20, +∞)	0	0%	0	0%

Πίνακας 5.9: Αποτελέσματα αναζήτησης για τιμές του score εύρους 5.

Score	Articles of Elasticsearch	Matched (%)	Articles of MeSH	Matched (%)
[0, 10)	1582	12%	438	43%
[5, 15)	588	25%	438	34%
[10, 20)	27	37%	133	7%
[15, 25)	1	100%	11	9%
[20, +∞)	0	0%	0	0%

Πίνακας 5.10: Αποτελέσματα αναζήτησης για τιμές του score εύρους 10.

Score	Articles of Elasticsearch	Matched (%)	Articles of MeSH	Matched (%)
[0, 20)	1615	12%	438	45%
[5, 40)	1615	12%	438	45%
[5, 20)	589	25%	438	34%

Πίνακας 5.11: Αποτελέσματα αναζήτησης για διάφορες τιμές του score.

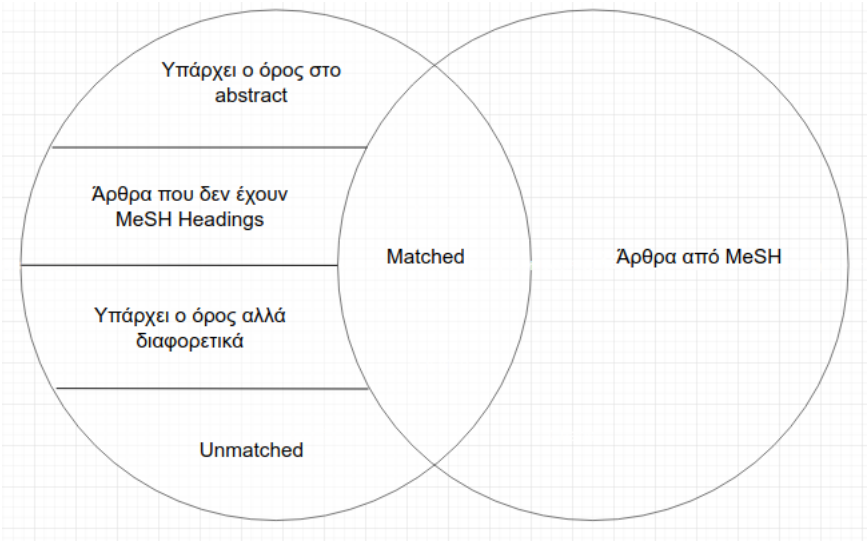
Όπως βλέπουμε στους παραπάνω πίνακες, τα αποτελέσματα ποικίλουν ανάλογα με το εύρος του score. Αρχικά στις πολύ μικρές τιμές (<2) και στις μεγάλες (>20) τα αποτελέσματα αποκλίνουν σημαντικά (Πίνακας 8). Αυτό οφείλεται σε δύο λόγους. Πρώτον, για πολύ μικρές τιμές τα ανακτούμενα αποτελέσματα είναι λιγότερο σχετικά. Αντίστοιχα, οι μεγάλες τιμές του score σημαίνουν μεγάλη σχετικότητα. Αυτό είναι δύσκολο να επιτευχθεί δεδομένου ότι μιλάμε για κείμενα μικρού μήκους και σύντομες λέξεις αναζήτησης και αυτό καθιστά δύσκολο το βέλτιστο ταίριασμα. Το καλύτερο δυνατό ταίριασμα βλέπουμε ότι επιτυγχάνεται στο εύρος [5, 15) (Πίνακας 8 και Πίνακας 10). Παρόλα αυτά, ακόμα και σε αυτές τις τιμές τα ποσοστά ταιριάσματος είναι μικρά. Από τα άρθρα που επιστρέφει το Elasticsearch, τα περισσότερα δεν έχουν τον όρο στα MeSH Headings του και το αντίστροφο. Χρειάζεται λοιπόν περαιτέρω ανάλυση των αποτελεσμάτων για να βρούμε την σχετικότητα και ακρίβειά τους, καθώς και τα πιθανά προβλήματα στα οποία οφείλονται οι αποκλίσεις.

5.3 Ακρίβεια και Ανάκληση (Precision and Recall)

Για την περαιτέρω ανάλυση των αποτελεσμάτων του Elasticsearch επιλέξαμε 10 ειδικούς όρους σχετικά με το σύνδρομο Sjogren, οι οποίοι αντιστοιχούν σε κάποιον MeSH όρο, και για αυτούς τους όρους εντοπίσαμε τα άρθρα που επιστρέφει το Elasticsearch και όλα εκείνα τα άρθρα τα οποία έχουν τον όρο αυτό στα MeSH Headings. Αυτό που θέλουμε να βρούμε ουσιαστικά είναι κατά πόσο είναι σωστά ή λάθος τα αποτελέσματα και στην περίπτωση του λάθους, εάν αυτό είναι αναμενόμενο και λογικό. Για καθένα από τα επιπλέον άρθρα ανατρέξαμε στο άρθρο αυτό στο PubMed και παρατηρήσαμε τι συμβαίνει και για ποιο λόγο ανήκει σε κάποιο σύνολο ενώ δεν ανήκει στο άλλο. Συγκεκριμένα κοιτάξαμε το abstract κείμενο αν υπάρχει ο όρος

αυτούσιος ή με διαφορετικά γραμμένος, καθώς και στα MeSH Headings αν ο όρος εμφανίζεται με άλλη ορολογία. Τα αποτελέσματα της ανάλυσης αυτής φαίνονται στους δύο παρακάτω πίνακες, πρώτα όσον αφορά τα επιπλέον άρθρα από το Elasticsearch και έπειρα τα επιπλέον από το MeSH.

Ο πρώτος πίνακας περιέχει για κάθε όρο το σύνολο των άρθρων που επιστράφηκαν από το Elasticsearch, το ποσοστό ταιριάσμος με τα άρθρα του MeSH και το πλήθος των επιπλέον άρθρων. Επίσης επεξηγείται τι συμβαίνει με τα επιπλέον άρθρα. Οι περιπτώσεις είναι τρεις: (I) όντως υπάρχει ο όρος στο abstract κείμενο αλλά όχι στα MeSH Headings, (II) το άρθρο δεν έχει πεδίο MeSH Headings γενικά και (III) υπάρχει ο όρος στα MeSH Headings αλλά με διαφορετική ορολογία (συνήθως πιο ειδική). Με το διάγραμμα Venn φαίνεται καλύτερα τι γίνεται με τα επιπλέον άρθρα του Elasticsearch.

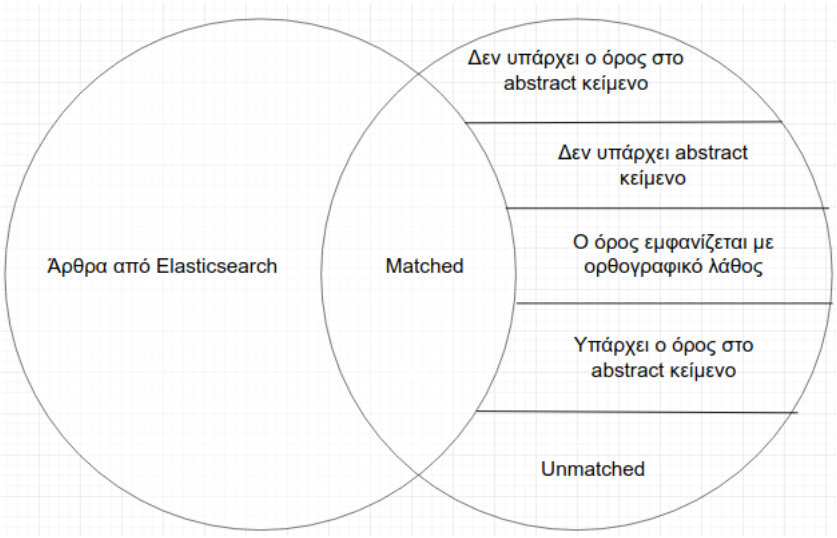


Σχήμα 5.7: Venn διάγραμμα συνόλων σύγκρισης: ανάλυση Elasticsearch.

Term	Articles of Elasticsearch	Matched (%)	Unmatched	Υπάρχει ο όρος στο abstract	Δεν έχουν MeSH Headings	Υπάρχει ο όρος διαφορετικά
Cyclosporine	18	8 (45%)	10	10	1	1
Twins	9	0 (0%)	9	9	0	5
Thrombocytopenia	36	7 (20%)	29	29	1	7
Proteinuria	26	5 (19%)	21	21	6	1
Lymphopenia	9	3 (34%)	6	6	0	2
Leukopenia	15	0 (0%)	15	15	1	0
Dementia	75	42(56%)	33	33	4	7
Cryoglobulins	12	3 (25%)	9	9	0	4
Anemia	50	6 (12%)	44	44	6	12
Rituximab	86	37 (43%)	49	47	8	4
Total	336	111 (33%)	225	223	27	43

Πίνακας 5.12: Ανάλυση δείγματος 10 ειδικών όρων (1/2: Elasticsearch).

Στον δεύτερο πίνακα, το σχετικό με τα MeSH Headings, για κάθε όρο πάλι υπάρχουν ο αριθμός των άρθρων που περιέχουν τον όρο στα MeSH Headings, το ποσοστό ταιριάσματος με τα άρθρα του Elasticsearch και το πλήθος των επιπλέον άρθρων. Όπως και προηγουμένως, επεξηγείται τι συμβαίνει με τα επιπλέον άρθρα. Εδώ υπάρχουν 4 περιπτώσεις: (I) ο όρος να μην υπάρχει καθόλου στο abstract, (II) το άρθρο να μην έχει abstract κείμενο, (III) ο όρος εμφανίζεται με ορθογραφικό λάθος στο κείμενο και (IV) να υπάρχει όντως ο όρος στο abstract κείμενο. Όπως και προηγουμένως, με το διάγραμμα Venn φαίνεται τι συμβαίνει με τα επιπλέον άρθρα του MeSH.



Σχήμα 5.8: Venn διάγραμμα συνόλων σύγκρισης: ανάλυση MeSH.

Term	Articles of MeSH	Matched (%)	Unmatched	Δεν υπάρχει ο όρος στο abstract	Δεν υπάρχει abstract κείμενο	Ο όρος εμφανίζεται με ορθογραφικό λάθος	Υπάρχει ο όρος στο abstract κείμενο
Cyclosporine	23	8 (35%)	15	8	1	0	6
Twins	3	0 (0%)	3	3	2	0	1
Thrombocytopenia	12	7 (58%)	5	0	4	1	0
Proteinuria	8	5 (62%)	3	2	1	0	0
Lymphopenia	3	3 (100%)	0	0	0	0	0
Leukopenia	3	0 (0%)	3	2	1	0	0
Dementia	56	42(75%)	14	2	11	1	0
Cryoglobulins	11	3 (27%)	8	0	6	2	0
Anemia	15	6 (40%)	9	1	8	0	0
Rituximab	51	37 (72%)	14	4	9	0	1
Total	185	111 (60%)	74	19	43	4	8

Πίνακας 5.13: Ανάλυση δείγματος 10 ειδικών όρων (2/2: MeSH Headings).

Συγκεντρωτικά λοιπόν βλέπουμε ότι το Elasticsearch επιστρέφει 336 άρθρα εκ των οποίων τα 334 όντως περιέχουν τον όρο στο abstract κείμενο. Τα 111 είναι αυτά που ταίριαξαν με το MeSH και συνεπώς υπάρχει επιπλέον πληροφορία 180 άρθρων (225-2-43). Όσον αφορά το MeSH, αν αφαιρέσουμε τα 43 άρθρα χωρίς κείμενο που δεν μπορούμε να τα λάβουμε υπόψιν μας, αναμέναμε εν τέλει 142 άρθρα, από τα οποία επιστράφηκαν από το Elasticsearch τα 111. Άρα λείπουν 31 άρθρα και το ποσοστό ταιριάσματος του MeSH γίνεται 78%.

Παρατηρώντας τα αποτελέσματα της αξιολόγησης αυτής βγάλαμε κάποια συμπεράσματα:

1. Όντως σε κάποια άρθρα υπάρχει ο όρος στα MeSH Headings αλλά όχι στο κείμενο (σε ποστοστό 13%), όπως και επίσης κάποιοι όροι εντοπίζονται στο abstract κείμενο αλλά δεν βρίσκονται στα MeSH Headings (σε ποσοστό 54%). Αυτό μπορεί να οφείλεται στο γεγονός ότι κάποια λέξη μπορεί να εκφράζει το νόημα του κειμένου και να είναι MeSH Heading, αλλά να μην εμφανίζεται αυτούσια (ή κάποια συνώνυμή της) μέσα στο κείμενο. Αντίστοιχα, ένας όρος να υπάρχει στο abstract, όμως να μην έχει τόσο μεγάλη σημασία.

2. Σε ένα μικρό αριθμό άρθρων υπήρχαν ορθογραφικά λάθη στα κείμενά τους (σε ποσοστό 2%). Πρόκειται για λάθος των συγγραφέων και δεν μπορούμε να το αντιμετωπίσουμε εύκολα στην περίπτωσή μας γιατί μιλάμε για βιοϊατρικούς όρους. Πρόκειται δηλαδή για πολύ ειδικούς όρους που ακόμα και ένα γράμμα θα έκανε τη διαφορά. Για παράδειγμα οι όροι “osteoblast” και “osteoclast” διαφέρουν μόνο κατά ένα γράμμα αλλά σημαίνουν κάτι τελείως το διαφόρετικό. Από την άλλη οι λέξεις “heart” και “haert” φαίνεται ότι πρόκειται για την ίδια λέξη, απλώς έχει γίνει διαφορετικό. Εν τέλει παρότι υπάρχουν τρόποι σύγκρισης δύο λέξεων και το κατά πόσο μοιάζουν, όμως από σημασιολογική πλευρά η διάκριση είναι δύσκολη. Ενδεικτικά ένας τέτοιος τρόπος είναι ο αλγόριθμος “Edit Distance” που παίρνει δύο strings και επιστρέφει ένα αριθμό που δείχνει πόσες αλλαγές χρειάζονται για να γίνουν ίδιες οι λέξεις.

3. Ένας σημαντικός αριθμός άρθρων περιέχουν τους ζητούμενους όρους στα keywords, τα οποία δεν εξετάζονται, ή ακόμα και στα MeSH Headings αλλά με διαφορετική ορολογία. Στην υλοποίησή μας λαμβάνουμε υπόψιν τα συνώνυμα αλλά όχι όλους τους όρους που σχετίζονται. Για παράδειγμα μπορεί να ψάχνουμε τον όρο “Twins” αλλά στα MeSH Headings να εμφανίζεται ο όρος “Twins, Monozygotic”. Συνειδητά έχουμε επιλέξει να μην επιλέγονται τέτοιοι όροι προκειμένου το αποτέλεσμα να είναι πιο ακριβές.

4. Σε κάποιες περιπτώσεις παρατηρήθηκε το φαινόμενο να μην υπάρχει abstract κείμενο για κάποιο άρθρο αλλά να έχει MeSH Headings. Το ποσοστό αυτό ήταν αρκετά σημαντικό (23%). Αυτό πρόκειται είτε για τεχνικό λάθος του PubMed είτε μπορεί να έχει αφαιρεθεί το κείμενο για κάποιο λόγο. Παρατηρώντας όμως τα άρθρα που συνέβαινε αυτό, τα περισσότερα ήταν γραμμένα σε ξένη γλώσσα και η ημερομηνία δημοσίευσής τους αρκετά παλιά. Προφανώς και τα άρθρα αυτά δεν τα λαμβάνουμε υπόψιν στις μετρήσεις μας.

Για να καταλάβουμε όμως καλύτερα και να μετρήσουμε τη σχετικότητα των αποτελεσμάτων του δείγματος αυτού υπολογίσαμε τα μεγέθη precision και recall. Το precision εστιάζει στην ακρίβεια, δηλαδή πόσα από αυτά που επιστρέφονται είναι σωστά.

Ισούται με τον αριθμό των σωστών αποτελεσμάτων που επιστράφηκαν προς το συνολικό αριθμό των αποτελεσμάτων. Το recall από την άλλη δείχνει πρακτικά τι χάσαμε, γιατί μπορεί η διαδικασία ταιριάσματος να είναι πολύ αυστηρή και τα αποτελέσματα να είναι πολύ ακριβή αλλά μην επιστρέφεται ένα μέρος των σωστών αποτελεσμάτων. Ισούται τον αριθμός των σωστών αποτελεσμάτων που επιστράφηκαν προς τον συνολικό αριθμό των σωστών. Οι τύποι υπολογισμού των δύο μεγεθών είναι οι παρακάτω:

$$precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|}$$

$$recall = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{relevant\ documents\}|}$$

Συνολικά λοιπόν θέλουμε να βρούμε τα άρθρα που επιστρέφει ως σωστά η δική μας μηχανή αναζήτησης και αυτά που είναι όντως σωστά. Το δεύτερο όμως μπορεί να γίνει μόνο αν εξετάζαμε ένα προς ένα όλα τα διαθέσιμα άρθρα μόνοι μας. Επειδή αυτό ήταν πρακτικά αδύνατο, θεωρήσαμε ότι τα σωστά είναι αυτά που έχουν το αντίστοιχο MeSH αν προσθέσουμε όμως και τα επιπλέον άρθρα που επιστρέφει η μηχανή αναζήτησής μας και τα οποία εξετάσαμε. Συνεπώς τα σχετικά αρχεία που αναμένουμε είναι τα 334 που επιστρέφει το Elasticsearch και περιέχουν σωστή πληροφορία, αλλά και τα 31 επιπλέον άρθρα του MeSH που βρήκαμε. Για να υπολογίσουμε το precision και recall αντικαταστήσαμε στους τύπους τα κατάλληλα νούμερα:

$$|\{relevant\ documents\} = 365$$

$$|\{retrieved\ documents\} = 336$$

$$|\{relevant\ documents\} \cap \{retrieved\ documents\}| = 334$$

$$precision = \frac{334}{336} = 0.99$$

$$recall = \frac{334}{365} = 0.91$$

Βλέπουμε ότι και το recall και το precision είναι πολύ υψηλά. Αυτό θεωρητικά σημαίνει ότι τα αποτελέσματά μας ήταν ακριβή και μάλιστα επιστρέφονται τα περισσότερα από αυτά που θα θέλαμε να επιστραφούν. Βέβαια δεν μπορούμε να είμαστε απόλυτοι λόγω των υποθέσεων που κάναμε.

Τέλος, ένα μέτρο που συνδιάζει το precision και το recall είναι το F-measure, που πρόκειται ουσιαστικά για τον αρμονικό μέσο των δύο αυτών μεγεθών. Αντικαταστήσαμε στον τύπο του τις τιμές των μεγεθών που βρήκαμε προηγουμένως και έχουμε:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} = 2 \cdot \frac{0.99 \cdot 0.91}{0.99 + 0.91} = 0.95$$

Συνολικά παρατηρούμε ότι τα αποτελέσματα είναι σε πολύ υψηλό βαθμό ακριβή και σχετικά. Μικρές αποκλίσεις υπάρχουν λόγω των παράπλευρων παραγόντων που αναλύθηκαν προηγουμένως σε συνδιασμό βέβαια με τον όχι απόλυτο τρόπο υπολογισμού του score στο Elasticsearch. Παρόλα αυτά το Elasticsearch μάς παρέχει μεγάλο όγκο επιπλέον πληροφορίας. Βέβαια δεν μπορούμε να κρίνουμε πόσο σημαντική είναι αυτή η πρόσθετη πληροφορία για τους χρήστες. Η αναζήτησή μας στηρίζεται στο abstract κείμενο με την υπόθεση ότι αν ο όρος υπάρχει στο κείμενο τότε σχετίζεται με το θέμα του. Για αυτό επιλέξαμε ειδικούς κλινικούς όρους ώστε να μην εμφανίζονται σε κείμενα που δεν είναι σχετικά. Συνεπώς ο στόχος της βελτιωμένης αναζήτησης επιτεύχθει με βάση το συγκεκριμένο σκεπτικό.

Κεφάλαιο 6

Επόμενα Βήματα

6.1 Εύρεση Συνακόλουθων Όρων (Patterns)

Στο προηγούμενο κεφάλαιο εστιάσαμε στους όρους που υπάρχουν και στο MeSH για να μπορούμε να συγκρίνουμε τα αποτελέσματα που επιστρέφει η δική μας μηχανή αναζήτησης με τα άρθρα που περιέχουν το MeSH. Ωστόσο η προσέγγιση που ακολουθήθηκε και η μηχανή αναζήτησης, βασισμένη στο Elasticsearch, που υλοποιήθηκε, θα μπορούσε να χρησιμοποιηθεί και για την αναζήτηση άρθρων που περιέχουν κάποιο όρο που δεν υπάρχει στο MeSH. Έτσι δίνεται η δυνατότητα στους επιστήμονες να μελετήσουν το σύνδρομο Sjogren αξιοποιώντας το σύστημά μας, πέραν της υπάρχουσας επιλογής του MeSH. Ένα ενδιαφέρον θέμα είναι η εύρεση συνακόλουθων όρων και χρειάζεται περισσότερη έρευνα προς αυτήν την κατεύθυνση.

Μία πρώτη προσπάθεια έγινε, όπως φαίνεται παρακάτω, όπου προσπαθήσαμε να εντοπίσουμε κάποιο μοτίβο στους όρους που χρησιμοποιήθηκαν στην αξιολόγηση του Elasticsearch. Οι κλινικοί όροι που χρησιμοποιήθηκαν συνολικά ήταν 197 και εμφανίστηκαν 9164 σε abstract κείμενα (ή κάποιο συνώνυμο). Στον παρακάτω πίνακα βλέπουμε τους 10 πρώτους κλινικούς όρους ως προς τον αριθμό εμφανίσεων τους.

Κλινικός όρος	**Αριθμός εμφανίσεων**
Homocysteine	1706
Leukocytes	1227
Multiple Myeloma	991
Rheumatoid Factor	812
Central nervous system	551
Antinuclear Antibodies	544
Platelets	450
Periphereal nervous system	442
Cancer	278
Lymphoma	259

Πίνακας 6.1: Οι 10 κορυφαίοι κλινικοί όροι που προκύπτουν από το Elasticsearch.

Συνεχίζουμε την ανάλυση βρίσκοντας για κάθε έναν από αυτούς τους όρους με ποιους άλλους όρους εμφανίζεται συνήθως. Από τις μετρήσεις το πιο σημαντικό που προκύπτει ότι οι όροι "Homocysteine" και "Leukocytes" εμφανίζονται και οι δύο μαζί με άλλους όρους σε υψηλά σχετικά ποσοστά. Ξεκινάνε από 30% και μπορούν να φτάσουν μέχρι και το 50%.

6.2 Βελτιώσεις

Η παρούσα διπλωματική εργασία θα μπορούσε να αποτελέσει τη βάση για περαιτέρω έρευνα στο μέλλον. Η αξιοποίησή της μπορεί να γίνει με διάφορους τρόπους. Πρώτον να βελτιωθεί το σύστημα για καλύτερα αποτελέσματα. Αυτό μπορεί να γίνει με τη δημιουργία καλύτερου analyzer λαμβάνοντας υπόψιν και άλλους παράγοντες (π.χ. keywords). Επίσης, μπορεί να βελτιωθεί η διαδικτυακή εφαρμογή για περισσότερες επιλογές αναζήτησης αλλά και καλύτερη παρουσίαση των αποτελεσμάτων. Ένας άλλος τρόπος έχει να κάνει με τη δοκιμή του συστήματος ώστε να εντοπιστούν προβλήματα και να προταθούν βελτιώσεις. Οι δοκιμές μπορούν να γίνουν με διαφορετικές βάσεις κι όχι μόνο άρθρα σχετικά με το σύνδρομο Sjogren, που είναι αρκετά ειδική περίπτωση. Επιπλέον μπορεί να γίνουν και έλεγχοι με τη βοήθεια χρηστών για την αξιολόγηση των επιστρεφώμενων αποτελεσμάτων. Τέλος, μία πιο εναλλακτική επιλογή θα ήταν να χρησιμοποιηθεί το σύστημα ή γενικά ο τρόπος λειτουργίας του σε καταστάσεις διαφορετικού πλαισίου από το τρέχον. Αντί να χρησιμοποιειταί για αναζήτηση βιοϊατρικής βιβλιογραφίας, να προσαρμοστεί σε τελείως διαφορετικές περιπτώσεις όπου και πάλι μπορεί να αξιοποιηθεί η αναζήτηση σε ελεύθερα κείμενα. Ειδικά σε μεγάλα κείμενα ώστε να εκμεταλλευτεί στο μέγιστο την ταχύτητα και ακρίβεια του Elasticsearch.

Κεφάλαιο 7

Σύνοψη

Η αύξηση της βιοϊατρικής βιβλιογραφίας έχει οδηγήσει στη δυσκολία αξιοποίησής της. Παρότι γίνεται προσπάθεια αντιμετώπισης αυτού του προβλήματος με διάφορους τρόπους, στην τρέχουσα διπλωματική επιλέχθηκε μία εναλλακτική μέθοδος, το Elasticsearch. Μετά την κατασκευή της βάσης δεδομένων για έναν συγκεκριμένο θέμα, το σύνδρομο Sjogren, για να περιορίσουμε τον όγκο των δεδομένων όσο είναι δυνατό, χρησιμοποιήσαμε το Elasticsearch για να αναζητήσουμε στα abstract κείμενα των άρθρων κλινικούς όρους. Η αξιολόγηση των αποτελεσμάτων που έγινε έδειξε ότι το Elasticsearch αποδίδει σε πολύ υψηλό βαθμό τόσο σε ακρίβεια όσο και σε σχετικότητα για αυτόν τον τρόπο αναζήτησης. Τα περιεχόμενα της βάσης αλλά και η δυνατότητα χρήσης του Elasticsearch είναι διαθέσιμα σε μία διαδικτυακή εφαρμογή. Φυσικά τα αποτελέσματα δεν είναι απόλυτα αλλά με περαιτέρω έρευνα και βελτιώσεις μπορεί να αξιοποιηθεί ακόμα περισσότερο στο μέλλον.

Βιβλιογραφία

[1] Densen, Peter. "Challenges and opportunities facing medical education"Transactions of the American Clinical and Climatological Association vol. 122 (2011): 48-58 .

[2] Can too much science be a bad thing? Growth in scientific publishing as a barrier to science communication, http://geekpsychologist.com/can-too-much-science-be-a-bad-thing-growth-in-scientific-publishing-as-a-barrier-to-science-communication/

[3] PubMed: https://www.ncbi.nlm.nih.gov/pubmed/.

[4] PubMed Tutorial: `https://www.nlm.nih.gov/bsd/disted/pubmedtutorial/cover.html`.

[5] Andreas Doms, Michael Schroeder; GoPubMed: exploring PubMed with the Gene Ontology, Nucleic Acids Research, Volume 33, Issue suppl_2, 1 July 2005.

[6] Medical Subject Headings (MeSH): `https://www.nlm.nih.gov/mesh/meshhome.html`.

[7] Zhiyong Lu,Won Kim, W. John Wilbur; Evaluation of query expansion using MeSH in PubMed, Information Retrieval, 2009.

[8] Elasticsearch: https://www.elastic.co.

[9] Zhiyong Lu; PubMed and beyond: a survey of web tools for searching biomedical literature, Database, Volume 2011, 1 January 2011, baq036, https://doi.org/10.1093/database/baq036

[10] Olivier Bodenreider; The Unified Medical Language System (UMLS): integrating biomedical terminology, Nucleic Acids Research, Volume 32, Issue suppl_1, 1 January 2004, Pages D267–D270, https://doi.org/10.1093/nar/gkh061.

[11] Sayers E. A General Introduction to the E-utilities. In: Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2010-.

[12] A Practical Guide on Elasticsearch Scoring and Revelancy. https://qbox.io/blog/practical-guide-elasticsearch-scoring-relevancy.

[13] Theory Behind Revelance Scoring. `https://opensourceconnections.com/blog/2015/09/18/the-simple-power-of-elasticsearch-analyzers/`.

[14] The Simple Power of Elasticsearch Analyzers. https://opensourceconnections.com/blog/2015/09/18/the-simple-power-of-elasticsearch-analyzers/.

[15] Elasticsearch from the Bottom Up, Part 1.https://www.elastic.co/blog/found-elasticsearch-from-the-bottom-up.

[16] Biomedical Literature Search Tools: `https://www.ncbi.nlm.nih.gov/CBBresearch/Lu/search/`.

[17] MEDLINE®PubMed® XML Element Descriptions and their Attributes. https://www.nlm.nih.gov/bsd/licensee/elements_descriptions.html.

[18] Apache Lucene. http://lucene.apache.org/.

[19] Logstash. https://www.elastic.co/products/logstash.

Παράρτημα

Τεχνολογίες

Για την δημιουργία του λογισμικού χρησιμοποιήθηκε κατά κύριο λόγο η γλώσσα προγραμματισμού **Java**. Προτιμήθηκε για πολλούς λόγους :

- Αντικειμενοστρέφεια. Βασίζεται σε κλάσεις και αντικείμενα. Έτσι γίνεται πιο ευέλικτη, μπορεί να διαχειριστεί και συντηρηθεί πιο εύκολα ακόμα και από τρίτους.
- Διαλειτουργικότητα. Είναι φορητή και μπορεί να χρησιμοποιηθεί ανεξάρτητα πλατφόρμας. Γράφεις τον κώδικα μία φορά και μετά το compiled πρόγραμμα μπορεί να αξιοποιηθεί από οποιονδήποτε με Java virtual machine (JVM) χωρίς ανάγκη για κάποια αλλαγή στο hardware, στο λειτουργικό σύστημα ή αναπροσαρμογής στο software. Επίσης, μπορεί να τρέξει σε προσωπικό υπολογιστή ή κινητό τηλέφωνο μέχρι servers ή πελάτες σε άλλο δίκτυο.
- Κληρονομικότητα. Κάποιες κλάσεις μπορεί να έχουν κοινά χαρακτηριστικά.
- Ασφάλεια ακόμα και αν λειτουργεί σε ένα δίκτυο. Επίσης μπορούμε να αποκρύψουμε όποια πεδία δεν θέλουμε ή δεν πρέπει να φανούν.
- Κλιμακωσιμότητα είτε οριζόντια είτε κάθετα.
- Σταθερότητα.
- Στιβαρότητα. Έχει δυνατό σύστημα διαχείρισης μνήμης. Χρησιμοποιείται η στοίβα (heap) και garbage collection.
- Just-In-Time compile.
- Υψηλή απόδοση. Χρήση νημάτων.
- API. Υπάρχουν πολλά εργαλεία διαθέσιμα σχετικά με I/O, δικτυώσεις, XML parsing, σύνδεση σε βάση δεδομένων με τη βοήθεια του JDBC (Java Database Connectivity) και πολλά άλλα.
- Υπάρχουν διαθέσιμα πολλά και ισχυρά εργαλεία ανάπτυξης όπως το Eclipse και το Netbeans, που βοηθάνε κατά τη διαδικασία συγγραφής κώδικα, ενώ κάνουν πιο εύκολο και το debugging.
- Συνίσταται στην ανάπτυξη Web applications.
- Είναι γλώσσα ανοιχτού κώδικα οπότε μπορεί να χρησιμοποιηθεί δωρεάν.

- Είναι δημοφιλής και ευρέως χρησιμοποιημένη σε εφαρμογές της καθημερινότητας αλλά και σε επιστημονικά-ερευνητικά έργα .
- Υπάρχει τεράστιος όγκος documentation και community support.
- Ευκολία εκμάθησης.

Για το σχεδιασμό, τη δημιουργία και επεξεργασία της βάσης δεδομένων χρησιμοποιήθηκε η **MySQL**. Πρόκειται για ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDMS). Τα πλεονεκτήματά της είναι τα εξής:

- Ασφάλεια, αξιοπιστία και στιβαρότητα. Παρέχει ισχυρούς μηχανισμούς ασφάλειας και ομαλής λειτουργίας. Οι δοσοληψίες (transactions) είναι πάντα ατομικές, συνεπείς, απομονωμένες και μόνιμες. Έτσι εξασφαλίζεται η ακεραιότητα των δεδομένων, που είναι απαραίτητη ειδικά σε περιπτώσεις οικονομικών συναλλαγών.
- Επιτρέπει σε πολλούς χρήστες να έχουν πρόσβαση σε πολλές βάσεις.
- Είναι σχεδιασμένη για Web εφαρμογές που χρειάζεονται μια βάση για αποθήκευση στο backend.
- Λειτουργεί με πολλές γλώσσες προγραμματισμού, όπως η Java.
- Υψηλή απόδοση.
- Υψηλή διαθεσιμότητα. 24 ώρες το 24ωρο λειτουργεί αδιάκοπα.
- Είναι ανοιχτού κώδικα και δωρεάν.
- Υπάρχει μεγάλη κοινότητα που συμβάλλει στη συντήρηση, debuggind και αναβάθμισή της. Επίσης υπάρχει και το λεγόμενο community support.
- Χρησιμοποιείται ευρέως σε εφαρμογές όπως το WordPress μέχρι και σε ορισμένους από τους πιο πετυχημένους οργανισμούς όπως η Google και το Facebook.
- Λειτουργεί σε όλες τις πλατφόρμες και λειτουργικά συστήματα.
- Το κατέβασμα και η εγκατάσταση γίνεται σε μικρό χρονικό διάστημα.
- Υπάρχουν και οπτικά εργαλεία διαχείρισης όπως το phpMyAdmin και το MySQL Workbench.

Για τη δημιουργία του Web-based application έγινε χρήση Java Server Pages (**JSP**). Πρόκειται για μια server-side τεχνολογία που επιτρέπει στους προγραμματιστές να δημιουργήσουν δυναμικά σελίδες. Πρακτικά είναι HTML σελίδες που περιέχουν κώδικα Java στο εσωτερικό τους. Επίσης μπορεί να περιέχουν και Javascript. Ουσιαστικά περιμένουν διαρκώς να "ακούσουν" κάποιο request και να απαντήσουν με response. Ένα JSP μετατρέπεται σε Servlet πριν εκτελεστεί. Τα θετικά είναι τα παρακάτω:

- Ευκολία στο γράψιμο κώδικα και κατ' επέκταση και στην ανάγνωσή του.
- Υπάρχει διαχωρισμός μεταξύ στατικού περιεχόμενο της HTML και δυναμικού της Java.

- Αξιοποιεί τη λειτουργία και δυνατότες της Java.
- Πρόσβαση σε όλα τα Java APIs, όπως το JDBC για πρόσβαση σε βάσεις δεδομένων.
- Οι περισσότερες Web εφαρμογές θέλουν σαν απάντηση HTML σελίδες. Το JSP λειτουργεί σαν web service.
- Υπάρχουν επαναχρησιμοποιούμενα κομμάτια κώδικα.
- Η JSP σελίδα είναι precompiled πριν επεξεργαστεί από το server (με εξαίρεση την πρώτη φόρα).

Προκειμένου να τρέξουμε τον κώδικα Java στο Web και συγκεκριμένα τα JSP, χρειάζεται ένας web server που να μας παρέχει ένα κατάλληλο περιβάλλον. Αυτός που μας το παρέχει αυτό και εφαρμόζει Java Servlets και JSP είναι ο **Apache Tomcat**. Πρόκειται για ένα web server και servlet container που τροφοδοτεί πολλές και μεγάλης κλίμακας εφαρμογές, ακόμα και κρίσιμης σημασίας, ενός εύρους οργανισμών. Τρέχει ένα Java virtual machine (JVM) και κάθε HTTP request από τον browser στο Tomcat επεξεργάζεται ξεχωριστά.

- Είναι ελαφρύς. Παρέχει μόνο τα απολύτως απαραίτητα για να τρέξει ένας server.
- Ευελιξία.
- Σταθερότητα.
- Ασφάλεια.
- Υψηλή απόδοση του website.
- Η χρήση του πρωτοκόλλου HTTP, που είναι στο επίπεδο εφαρμογής και χρησιμοποιεί TCP/IP πακέτα, οδηγεί στην αξιοποίηση των παροχών του IP. Το IP παρέχει υποστήριξη για δρομολόγηση και διευθυνσιοδότηση (κάθε μηχάνημα έχει μοναδική IP διεύθυνση).
- Στην παραγωγή μπορεί να χρησιμοποιηθεί μέσω εργαλέιων όπως το Eclipse και το Netbeans.
- Εύκολη εγκατάσταση και ρύθμιση.
- Είναι ανοιχτού κώδικα και δωρεάν.

Η συγγραφή κώδικα Java έγινε πάνω στο ενοποιημένο περιβάλλον ανάπτυξης (IDE) **Eclipse**. Διαθέτει πολλά χαρακτηριστικά και επεκτάσεις, ενώ τα εργαλεία του το κάνουν ιδανικό για την ανάπτυξη εφαρμογών. Κάποια από τα πλεονεκτήματα του είναι:

- Το debugging γίνεται ευκολότερα.
- Refactoring με το find/replace λέξεων.
- Η πλοήγηση είναι ευκολότερη εφόσον υπάρχουν όλα τα αρχεία και οι φάκελοι σε μία μόνο οθόνη.

- Οργάνωση των imports.
- Αυτόματη συμπλήρωση κώδικα. Αποφυγή χάσιμου χρόνου ψάχνοντας στο documentation όλες τις μεθόδους.
- Κατέβασμα πακέτων γρήγορα και εύκολα.
- Έλεγχος για συντακτικά λάθη την ώρα που πληκτρολογείς κώδικα.
- Εξοικονόμιση χρόνου και κόπου συνολικά.
- Είναι ανοιχτού κώδικα και δωρεάν.
- Υποστηρίζει και άλλες γλώσσες πέρα από Java.
- Χρησιμοποιείται και σε βιομηχανικό επίπεδο.

Η διαχείριση της MySQL έγινε με το web οπτικό εργαλείο **phpMyAdmin**. Είναι για ένα δωρεάν εργαλείο γραμμένο σε PHP που παρέχει πλήθος λειτουργιών πάνω σε MySQL που πραγματοποιούνται μέσω γραφικού περιβάλλοντος. Ο σχεδιασμός και η δημιουργία της βάσης, η εισαγωγή και η επεξεργασία των δεδομένων θα γίνει με το εργαλείο αυτό. Οι λόγοι επιλογής του είναι οι εξής:

- Παρέχει Web διεπαφή.
- Διαχειρίζεται MySQL.
- Υποστηρίζει όλα τα διαθέσιμα queries.
- Import/Export SQL. Βοηθά στη μεταφορά βάσεων.
- Δημιουργία διαγραμμάτων.
- Σταθερότητα και αναθαθμίζεται συχνά.
- Πλοήγηση μεταξύ των βάσεων.
- Είναι ανοιχτού κώδικα και δωρεάν.
- Λειτουργεί σε διάφορα λειτουργικά συστήματα.
- Πλούσιο documentation.

Το PubMed επιστρέφει τα δεδομένα που θέλουμε (το κάθε άρθρο με τις σχετικές πληροφορίες) σε μορφή **XML**. Η XML είναι μια γλώσσα σήμανσης που περιέχει ένα σύνολο κανόνων για τη κωδικοποίηση εγγράφων. Είναι σχεδιασμένη για την αποθήκευση και ανταλλαγή δεδομένων. Μπορεί να αναγνωστεί τόσο από άνθρωπο όσο και από υπολογιστή. Χρησιμοποιεί ιεραρχημένες ετικέτες (tag) που μπορούν να προσαρμοστούν στις ατομικές ανάγκες του χρήστη δίχως κανέναν περιορισμό. Η XML στηρίζεται στην απλότητα και χρησιμοποιείται καθολικά ακομά και μεταξύ διαφορετικών εφαρμογών.

Το Elasticsearch επιστρέφει τα αποτελέσματά του (τα ζητούμενα άρθρα στη σωστή σειρά) σε μορφή **JSON**. Το JSON είναι απλό κείμενο με συγκεκριμένο συντακτικό

και χρησιμοποιείται για την αποθήκευση και ανταλλαγή πληροφορίας. Είναι ιδιαίτερα χρήσιμο, ειδικά στην περίπτωση της ανταλλαγής δεδομένων μεταξύ browser και server όπου επιτρέπονται τα δεδομένα να είναι μόνο κείμενο. Είναι ασφαλές, ευέλικτο, γρήγορο, εύκολο στη γραφή και στην κατανόηση και μικρότερου μεγέθους από το αντίστοιχο XML. Η χρήση του στις Web εφαρμογές ενισχύεται και από το γεγονός ότι λειτουργεί καλύτερα με JavaScript, καθώς και ότι είναι ευκολότερη η ανάλυση και ο χειρισμός των response. Επίσης είναι ανεξάρτητο γλώσσας και υποστηρίζει και πίνακες.

Σε δύο περιπτώσεις χρειάστηκε να ανακτήσουμε δεδομένα από web server. Πρώτα όταν θέλαμε τα xml αρχεία από το PubMed που είναι απαραίτητα για τη βάση δεδομένων και δεύτερον όταν αναζητάγαμε στο Elasticsearch. Για να το πετύχουμε αυτό χρησιμοποιήθηκε το **HTTP request**, όπου ουσιαστικά πρόκειται για ένα μήνυμα που ζητάει ο πελάτης πληροφορία από το server και αυτό γίνεται με το πρωτόκολλο εφαρμογής HTTP. Με αυτό το πρωτόκολλο, που είναι stateless, γίνεται η μεταφορά κειμένων και εικόνων από το web server στον υπολογιστή. Μόλις ζητηθούν τα δεδομένα με το HTTP request, ο server απαντάει στο browser με HTML σελίδα ή οποιοδήποτε άλλο αρχείο ζητήθηκε, αν όλα πήγαν καλά και δόθηκε η άδεια. Το HTTP request έχει τα εξής στοιχεία : Method, Request-URI (καθορίζει τους πόρους που ζητούνται) και το Protocol Version. Η μέθοδος (method) καθορίζει τι θα γίνει στους πόρους του Request-UI. Οι βασικές είναι οι GET, HEAD, POST, PUT, DELETE. Η μέθοδος που χρησιμοποιείται για ανάκτηση πληροφορίας από το server μέσω του URI χωρίς να επηρεάζει τα δεδομένα είναι η GET. Αυτή πρόκειται να χρησιμοποιηθεί στην δική μας περίπτωση.

Αποσπάσματα κώδικα

Σε αυτήν την ενότητα παροουσιάζονται κάποια αποσπάσματα κώδικα.

SQL ερωτήματα για τη σύμπτιξη όμοιων εγγραφών των keywords

(1) Ενημέρωση του πίνακα της σχέσης του άρθρου με τη λέξη-κλειδί
UPDATE KeywordArticleConnection C0
INNER JOIN Keyword C1 ON C0.Id_Keyword=C1.Id_Keyword
INNER JOIN Keyword C2 ON C1.TheKeyword=C2.TheKeyword
SET C0.Id_Keyword=IF(C1.Id_Keyword<C2.Id_Keyword,
C1.Id_Keyword,C2.Id_Keyword)
WHERE (C1.Id_Keyword<C2.Id_Keyword OR C1.Id_Keyword>C2.Id_Keyword)

(2) Διαγραφή των περιττών εγγραφών Keyword
DELETE FROM Keyword WHERE Id_Keyword NOT IN (SELECT
MIN(Id_Keyword) FROM (SELECT * FROM Keyword) as C0
GROUP BY TheKeyword)

Ανάκτηση του XML αρχείου του άρθρου και μετατροπή του σε Java Object

String url_eutils = "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?

```
db=pubmed&retmode=xml&id=" + temp_id;
try {
URL myurl = new URL(url_eutils);
con2 = (HttpURLConnection) myurl.openConnection();
con2.setRequestMethod("GET");
StringBuilder content;
try (BufferedReader in = new BufferedReader(
new InputStreamReader(con2.getInputStream()))) {
String line;
content = new StringBuilder();
while ((line = in.readLine()) != null)  content.append(line);
content.append(System.lineSeparator()); /
}
}
File tempFile = File.createTempFile("temp_article", ".xml");
java.io.FileWriter fw = new java.io.FileWriter(tempFile);
fw.write(content.toString());
fw.close();
JAXBContext jaxbContext = JAXBContext.newInstance
(PubmedArticleSet.class);
Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();
PubmedArticleSet full_article = (PubmedArticleSet)
jaxbUnmarshaller.unmarshal(tempFile);
PubmedArticle temp_article = (PubmedArticle)
full_article.getPubmedArticleOrPubmedBookArticle().get(0);
```

Ανάκτηση PMID άρθρου

```
int PMID = Integer.parseInt(temp_article.getMedlineCitation().
getPMID().getvalue());
```

Αναζήτηση στο Elasticsearch και ανάκτηση δεδομένων

```
String url = "http://localhost:9200/abstracts/_search?q=abstracttext:" +
term + "&analyzer=hmmy_analyzer&size=5000";
try {
URL myurl = new URL(url);
con = (HttpURLConnection) myurl.openConnection();
con.setRequestMethod("GET");
StringBuilder content;
try (BufferedReader in =
new BufferedReader(new inputStreamReader(con.getInputStream()))) {
String line;
content = new StringBuilder();
while ((line = in.readLine()) != null) {
content.append(line);
content.append(System.lineSeparator()); }
}
```

```
JSONParser jsonParser = new JSONParser();
Object object;
object = jsonParser.parse(content.toString());
JSONObject jsonObject = (JSONObject) object;
long took = (long) jsonObject.get("took");
JSONObject _shards = (JSONObject) jsonObject.get("_shards");
long total = (long) _shards.get("total");
JSONObject hits = (JSONObject) jsonObject.get("hits");
long total_hits = (long) hits.get("total");
JSONArray hits_hits = (JSONArray) hits.get("hits");
int counter = hits_hits.size();
if(counter==0) System.out.println("abort");
for(int i=0; i<counter; i++) {
JSONObject article = (JSONObject) hits_hits.get(i);
double _score = (double) article.get("_score");
JSONObject _source = (JSONObject) article.get("_source");
long PMID = (long) _source.get("pmid");
}
```