Επιρροή Σημασιολογίας στην Επίδοση της Ανάκτησης Ιατρικής Πληροφορίας

Διπλωματική Εργασία

Αν μπορούσαμε να εκμεταλλευτούμε την σημασιολογία των αναζητούμενων δεδομένων τότε θα μπορούσαμε να βελτιώσουμε σημαντικά τις αναζητήσεις μας.

Σταμελάκη Αδαμαντία-Ελένη [3050249]
Ακαδ. Έτος: 2009-2010

Επιρροή Σημασιολογίας στην Επίδοση της Ανάκτησης Ιατρικής Πληροφορίας.

Περίληψη.

Η πετυχημένη ανάκτηση πληροφοριών είναι ένα καθημερινό πρόβλημα που συναντάται στα συστήματα βάσεων δεδομένων. Όταν αυτή σχετίζεται με ιατρική πληροφορία είναι ιδιαίτερα σημαντική, μιας και τα άμεσα, γρήγορα και σωστά αποτελέσματα ενός τέτοιου μηχανισμού θα μπορούσαν να κερδίσουν πολύτιμο χρόνο σε κρίσιμες περιπτώσεις. Το πρόβλημα αυτό βασίζεται στην εύρεση κατάλληλων μεθόδων και τεχνικών, ώστε να βελτιωθούν τα αποτελέσματα των αναζητήσεων μας. Αυτό μπορεί να επιτευχθεί μέσα από την εκμετάλλευση της σημασιολογίας των αναζητούμενων δεδομένων. Σκοπός της αυτής της αναφοράς είναι η ανάλυση των θεμάτων που σχετίζονται με την υλοποίηση και αξιολόγηση διαφόρων μηχανισμών που εκμεταλλεύονται την σημασιολογία των υποβαλλόμενων ερωτημάτων κατά την ανάκτηση τους. Στόχος μας είναι να εντοπιστεί ένας μηχανισμός που εφαρμοζόμενος σε διαφορετικά περιβάλλοντα θα μπορούσε να επιτύχει αποτελέσματα που χωρίς την χρήση του δεν ήταν εφικτά. Προκειμένου να επιτευχθεί αυτό χρησιμοποιήθηκαν δύο ιατρικές βάσεις (OHSUMED και ImageCLEF2009) στις οποίες εφαρμόστηκε μια σειρά από πειράματα. Στην αναφορά αναλύονται η κάθε μια από τις αυτές τις βάσεις, ο τρόπος με τον οποίο ανακτήθηκαν τα στοιχεία τους και καταχωρήθηκαν σε ευρετήρια, όλα τα πειράματα που έγιναν στις βάσεις και το μέρος των αποτελεσμάτων που κρίθηκε απαραίτητο. Επίσης αναλύεται το εργαλείο Metamap και πως αυτό χρησιμοποιήθηκε στα πειράματα που έγιναν. Τέλος, η διαδικασία κατά την οποία προστέθηκαν κατηγορίες στην βάση του ImageCLEF2009, καθώς και όλα τα εργαλεία, πακέτα που χρησιμοποιήθηκαν για την επίτευξη και αξιολόγηση των παραπάνω πειραμάτων.

Abstract.

The successful retrieval of information is a daily problem encountered in database systems. When it relates to medical information is especially important, since the direct, fast and correct results of such a mechanism could gain valuable time in critical situations. The problem is based on finding appropriate methods and techniques to improve our search results. This can be achieved by taking advantage of the meaning of search data. The purpose of this report is to analyze issues related to implementation and evaluation of various schemes that exploit the semantics of questions submitted during the retrieval. Our goal is to identify a mechanism that can be applied to different environments and could achieve results that without its usage was not ever be feasible. In order to achieve that, a series of experiments were implemented in two medical databases (OHSUMED and ImageCLEF2009). In this report we analyze each of these bases, the way their data retrieved and their indexes created, all the experiments in them and part of the results. Moreover, Metamap tool and how it was used in the experiments were made. Finally, the process by which categories were added to the base ImageCLEF2009, and all the tools, kits used to achieve and evaluate these experiments.

Εισαγωγή.

Τα Συστήματα Ανάκτησης Πληροφοριών (Information Retrieval Systems) επιτρέπουν την πρόσβαση σε μεγάλους όγκους πληροφοριών αποθηκευμένων με ποικίλες μορφές. Σκοπός των συστημάτων αυτών είναι η ανάκτηση μόνο εκείνων των πληροφοριών-εγγράφων που είναι συναφή με ότι αναζητεί ο χρήστης. Για να επιτευχθεί αυτό πρέπει να αντιμετωπίσουν η αβεβαιότητα ως προς το τι πραγματικά αναζητείτε και ποιο το θέμα ενός εγγράφου. Αν λοιπόν επιτυγχανόταν με κάποιο μηχανισμό η εισαγωγή και γενικότερα η εκμετάλλευση της σημασιολογίας των υποβαλλόμενων ερωτημάτων στις αναζητήσεις του χρήστη τότε αναμένεται να βελτιωθεί σημαντικά η επίδοση όσον αφορά τα ζητούμενα αποτελέσματα. Κάτω από αυτά τα πλαίσια λοιπόν έγιναν διάφορα πειράματα σε δύο συλλογές ιατρικών δεδομένων (την OHSUMED και την βάση ImageCLEF2009) προκειμένου είτε να εκμεταλλευτούμε υπάρχουσες κατηγορίες των κειμένων αυτών, είτε να προσαρτηθούν σε αυτά κατηγορίες και να ελεγχθεί κατά πόσο βελτιώθηκαν οι επιτυχίες των αναζητήσεων ή όχι. Στα ίδια πλαίσια έγιναν και πειράματα λειτουργίας του μηχανισμού που κατασκευάστηκε όταν τα κατηγοριοποιημένα κείμενα υπόκεινται ελεγχόμενο σφάλμα όσον αφορά τις κατηγορίες τους. Σκοπός ήταν η διερεύνηση κατά πόσο θα μπορούσε ένας τέτοιος μηχανισμός να βοηθήσει σε ένα περιβάλλον αυτόματης κατηγοριοποίησης των κειμένων.

Η συλλογή OHSUMED.

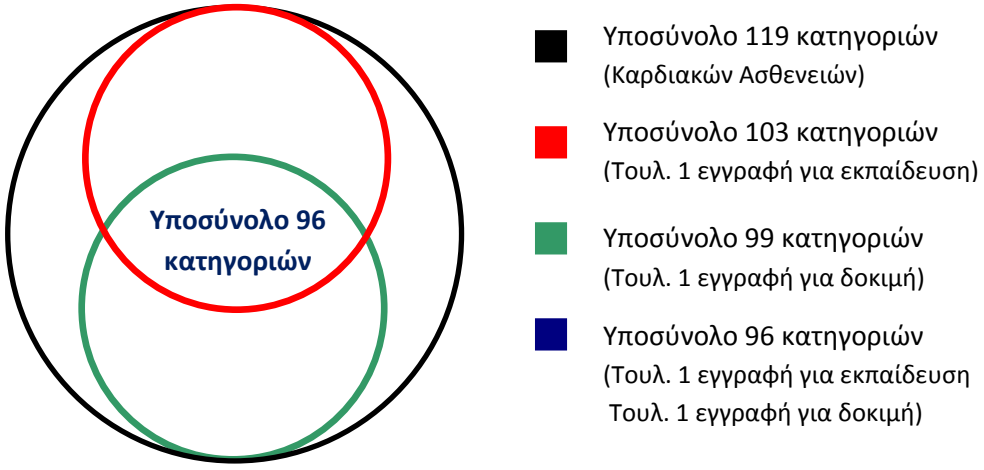
Μια ιατρική συλλογή που χρησιμοποιήσαμε στα πειράματα ήταν η OHSUMED. Η συλλογή αυτή δημιουργήθηκε από τον William Hersh και άλλους, το 1994[i]. Ήταν η πρώτη μεγάλη βάση δεδομένων που ήταν διαθέσιμη για δοκιμές στον τομέα της κατηγοριοποίησης κειμένων. Το αρχικό σώμα της αποτελείται από περίπου 350.000 έγγραφα που προέρχονται από 270 ιατρικά περιοδικά κατά τη διάρκεια των ετών 1987-1991. Ένα μικρότερο σύνολο δεδομένων, χρησιμοποιούμενα συνήθως στη έρευνα, περιέχει 20.000 έγγραφα της αρχικής συλλογής. Τα έγγραφα αυτά αποτελούνται από έναν τίτλο, συγγραφέα και άλλες πληροφορίες. Ακόμα μπορεί να έχουν ευρετηριοποιηθεί με βάση όρους του MeSH (Medical Subject Headings) οι οποίοι έχουν επιλεγεί χειρονακτικά από ειδικούς στο National Library of Medicine (NLM). Τα έγγραφα με id από 100.001 έως 110.000 χρησιμοποιούνται ως συλλογή εκπαίδευσης και τα έγγραφα με id από 110.001 έως 120.000 χρησιμοποιούνται ως δοκιμαστική συλλογή. Επιπλέον, τα έγγραφα που δεν αντιστοιχούν σε κάποια κατηγορία αποκλείονται και διατηρούνται μόνο εκείνα που ανήκουν σε μία από τις 23 κατηγορίες του υποδέντρου του MeSH που αφορά καρδιακές ασθένειες (Ohsumed-23 dataset).

Να σημειωθεί ότι δεν είναι όλα τα έγγραφα χρήσιμα, εξαιτίας του γεγονότος ότι μερικά από αυτά δεν έχουν είτε τίτλο, είτε απόσπασμα είτε δεν έχει οριστεί κάποιος MeSH όρος για αυτά. Το υποσύνολο των απόλυτα λειτουργικών κειμένων, δηλαδή αυτών που περιέχουν όλα τα χαρακτηριστικά, τίτλο, κείμενο και κάποια κατηγορία αποτελείτε από 233.455 κείμενα και εκπονήθηκε από τον D. Lewis, και άλλους κατά την διάρκεια της έρευνα του πάνω στην κατηγοριοποίηση κειμένων[ii].

Το σύνολο αυτό που αποτελείτε από 233.455 εγγραφές περιέχει τα έγγραφα, με τίτλο, περίληψη και την αντίστοιχη κατηγορία ή κατηγορίες που έχουν καταχωρηθεί για αυτά.

Από αυτά 183.299 έγγραφα από τα έτη 1987-1990 αποτελούν το σύνολο εκπαίδευσης και 50.216 έγγραφα από το έτος 1991 αποτελούν το σύνολο δοκιμών που έχουν τεθεί. Οι κατηγορίες αποτελούνται από 119 κατηγορίες του υποδέντρου Καρδιακών Ασθενειών της δομής Καρδιαγγειακά Νοσήματα στο δέντρο του MeSH. Ωστόσο, μόνο 103 από αυτές τις 119 κατηγορίες έχουν έγγραφα στο σύνολο εκπαίδευσης και μόνο 96 έχουν τουλάχιστον ένα έγγραφο για την εκπαίδευση και ένα για τις δοκιμές που έχουν τεθεί. Έτσι περιορίζονται τα πειράματά μας σε αυτές τις 96 κατηγορίες (Ohsumed-96 dataset). Το σύνολο των 96 κατηγοριών έχει υποδιαιρεθεί σε δύο υποσύνολα.

Το επόμενο διάγραμμα παρουσιάζει πως αυτές οι κατηγορίες επικαλύπτονται μεταξύ τους:



Στις προηγούμενες παραγράφους περιγράψαμε τα τρία μεγάλα υποσύνολα κατηγοριών του συνόλου δεδομένων της OHSUMED. Αυτό σημαίνει ότι περιγράψαμε τις τρις δεσμίδες που περιλαμβάνουν ετικέτες κατηγοριών, αλλά δεν αναφέραμε τίποτα για τα έγγραφα που περιέχουν αυτά τα υποσύνολα. Είναι φυσικό να υποθέσουμε ότι κάθε υποσύνολο κατηγοριών περιέχει μόνο τα έγγραφα που ανήκουν σε κάποια από τις κατηγορίες του υποσύνολου. Για παράδειγμα, το υποσύνολο με τις 96 κατηγορίες θα περιέχει μόνο τα έγγραφα που ανήκουν σε κάποια από αυτές τις 96 κατηγορίες και δεν θα περιέχει κανένα άλλο έγγραφο, που είτε ανήκει σε μηδέν κατηγορίες ή ανήκει σε άλλες κατηγορίες. Η μέθοδος αυτή μειώνει το μέγεθος του προκύπτοντος υποσύνολου εγγράφων, διότι εξαιρεί τα έγγραφα που δεν ανήκουν σε καμία από τις κατηγορίες. Αυτός είναι ο λόγος για τον οποίο δημιουργήθηκαν υποσύνολα εγγράφων και αναφερόμαστε σε αυτά ως "x" σύνολα δεδομένων ("x" από το " exclusive", αποκλειστικό).

Ένα υποσύνολο των 49 μεγαλύτερων κατηγοριών, αυτά που περιέχουν τουλάχιστον 75 έγγραφα για εκπαίδευση και ένα υποσύνολο των 28 κατηγοριών με 15 έως 74 έγγραφα που αποτελούν το συνδυασμό εκπαίδευσης (Ohsumed-49 και Ohsumed-28 datasets). Από τα τελευταία τρία σύνολα δεδομένων που έχουμε παράχθηκαν άλλα τρία σύνολα δεδομένων, δηλαδή, OhsumedBIG-96-x, OhsumedBIG-49-x και OhsumedBIG-28-x, που περιέχουν έγγραφα αποκλειστικά στην αντίστοιχη κατηγορία.

Έτσι στα πειράματα μας μπορούμε να χρησιμοποιούμε έξι διαφορετικά υποσύνολα εγγραφών. Τα στατιστικά στοιχεία των συνόλων των στοιχείων παρουσιάζονται στον επόμενο πίνακα.

Όνομα Βάσης	**96**	**96x**	**49**	**49x**	**28**	**28x**
Αριθμός κειμένων εκπαίδευσης	183,229	12,822	183,229	12,445	183,229	953
Αριθμός κειμένων δοκιμών	50,216	3,760	50,216	3,632	50,216	274
Αριθμός όρων εκπαίδευσης	98,893	17,323	98,893	17,031	98,893	5,201
Αριθμός όρων δοκιμών	57,497	10,842	57,497	10,625	57,497	3,153

Για τα πειράματα μας χρησιμοποιήθηκε η συλλογή OHSUMED με το σύνολο από τα 233,455 αρχεία τα οποία αποτελούνται από έναν τίτλο, ένα απόσπασμα και τις κατηγορίες στις οποίες αυτό ανήκει. Οι κατηγορίες τους αποτελούνται από 119 υποκατηγορίες που σχετίζονται με Καρδιακές παθήσεις (Heart Diseases) που, όπως ήδη αναφέραμε, αποτελεί υποκατηγορία του δέντρου Cardiovascular Diseases της δομή του MeSH. Εκτός από την συλλογή έχουμε και 105 ερωτήσεις που παρήχθησαν από γιατρούς παθολόγους. Κάθε ερώτηση αποτελεί μια δήλωση για τον ασθενή και την απαιτούμενη πληροφορία. Για όλες αυτές τις ερωτήσεις έχουμε και τις αντίστοιχες απαντήσεις τους προκειμένου να είναι εφικτή η αξιολόγηση των αποτελεσμάτων μας μετά από την διεξαγωγή τους. Η αξιολόγηση γίνετε με χρήση του Text REtrieval Conference (TREC).

Πειραματική διερεύνηση της επιρροής της σημασιολογίας στην επίδοση της ανάκτησης.

Με χρήση λοιπόν της παραπάνω συλλογής έγινε μια σειρά πειραμάτων προκειμένου να εντοπιστεί κατά πόσο μπορεί να επηρεαστούν τα αποτελέσματα μιας αναζήτησης αν σε αυτήν πέρα από τα στοιχεία τον κειμένων είχαμε και τις θεματικές ενότητες/κατηγορίες των κειμένων.

Προεργασία.

Parsing.

Προκειμένου να χρησιμοποιηθούν τα στοιχεία της βάσης ήταν απαραίτητο να δημιουργηθούν οι κατάλληλες συνθήκες επεξεργασίας των δεδομένων. Για τον λόγο αυτό χρησιμοποιήθηκε το Digester Component της Apache Software Foundation. Με την βοήθεια αυτού του πακέτου και την υλοποίηση που έγινε κατασκευάστηκε ένας parser που διαχώριζε τα στοιχεία της βάσης και τα έδινε στο lucene προκειμένου να κατασκευαστούν τα ευρετήρια. Ο parser αυτός θα μπορούσαμε να πούμε ότι αποτελείτε από δύο μέρη, μια τάξη η οποία χρησιμοποιείτε προκειμένου να μεταφέρουμε την οντότητα του XML αρχείου σε ένα αντικείμενο Java που το έχουμε ονομάσει Record. Καθώς και μια μέθοδο που είναι υπεύθυνη να ορίσει ποιο στοιχείο της οντότητας αντιστοιχεί σε ποιο πεδίο/χαρακτηριστικό του αντικειμένου μας.

Κώδικας τάξης αντικειμένου που αντιπροσωπεύει ένα έγγραφο κειμένου.

```
public class Record {
    private String ohsumedID;
    private Vector<String> categories;
    private String title;
    private String abstruct;
    public Record() {
        categories = new Vector();
    }
. . .
}
```

Κώδικας μεθόδου που κάνει parse στο xml.

```
protected void digest(File xmlFile) {
    try {
       Digester digester = new Digester();
       //Push the current object onto the stack
       digester.push(this);
        //Creates a new instance of the Record class
        digester.addObjectCreate("TEXT/RECORD", Record.class);
        // βάζω τα πεδία του xml σε αυτά της Record
        digester.addBeanPropertySetter("TEXT/RECORD/OHSUMED-ID",
"ohsumedID");
        digester.addBeanPropertySetter("TEXT/RECORD/TITLE", "title");
        digester.addBeanPropertySetter("TEXT/RECORD/ABSTRACT",
"abstruct");
        //Call Method addCategory that takes a single parameter
        digester.addCallMethod("TEXT/RECORD/CATEGORIES/CATEGORY",
"addCategory", 1);
       //Set value of the parameter for the addCategory method
       digester.addCallParam("TEXT/RECORD/CATEGORIES/CATEGORY", 0);

       //Move to next record
       digester.addSetNext("TEXT/RECORD", "addRecord");

       DigestRecordsIndex ds = (DigestRecordsIndex)
digester.parse(xmlFile);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

Indexing.

Για την δημιουργία των ευρετηρίων χρησιμοποιήθηκε ο παρακάτω κώδικας και ο StandardAnalyzer του πακέτου Lucene.

Κώδικας αρχικοποίησης ευρετηρίων.

```
// που θα γίνει το index
File indexDir = new File("OHSUMED"+File.separator+"index");
//Create instance of Directory where index files will be stored
Directory fsDirectory = FSDirectory.open(indexDir);
//Create instance of analyzer, which will be used to tokenize the input data
Analyzer standardAnalyzer = new StandardAnalyzer(Version.LUCENE_30);
//Create a new index
writer = new IndexWriter(fsDirectory, standardAnalyzer, true, new KeepOnlyLastCommitDeletionPolicy(), IndexWriter.MaxFieldLength.UNLIMITED);


DigestRecordsIndex digestRecords = new DigestRecordsIndex();
File xmlFolder = new File("OHSUMED" + File.separator + "Dataset");
if (xmlFolder.isDirectory()) {
    File[] xmlsToIndexed = SharedMethods.getXMLFiles(xmlFolder);
    for (int i = 0; i < xmlsToIndexed.length; i++) {
        System.out.println("File " + xmlsToIndexed[i].getName());
        digestRecords.digest(xmlsToIndexed[i]);
    }
}
writer.optimize();
writer.close();
```

Πληροφορίες και κώδικας δημιουργίας διαφορετικών πεδίων στα ευρετήρια.

Δημιουργήθηκαν τρία ευρετήρια τα οποία και χρησιμοποιήθηκαν για την πραγματοποίηση αναζητήσεων σε 105 διαφορετικά ερωτήματα, τα αποτελέσματά τους αξιολογήθηκαν με το trec_eval. Τα ευρετήρια διαφοροποιούνταν όσον αφορά τα πεδία αναζήτησης τα οποία αυτά περιείχαν. Το πρώτο ευρετήριο αποτελούταν από ένα πεδίο που περιείχε τα στοιχεία εκείνα της βάσης που σχετίζονταν με τον τίτλο και την περιγραφή της εγγραφής μας.

```
public void addRecord(Record rec) throws CorruptIndexException, IOException {
    Document recdoc = new Document();
    recdoc.add(new Field("docID", rec.getOhsumedID(), Field.Store.YES, Field.Index.NOT_ANALYZED_NO_NORMS));
    recdoc.add(new Field("contents", rec.getContect(), Field.Store.YES, Field.Index.ANALYZED));
    writer.addDocument(recdoc);
}
```

Το επόμενο αποτελούταν από δυο πεδία ένα για τίτλο και περιγραφή και ένα τις κατηγορίες που περιείχε η βάση.

```
    public void addRecord(Record rec) throws CorruptIndexException,
IOException {
        Document recdoc = new Document();
        recdoc.add(new Field("docID", rec.getOhsumedID(), Field.Store.YES,
Field.Index.NOT_ANALYZED_NO_NORMS));
        recdoc.add(new Field("contents", rec.getContect(),
Field.Store.YES, Field.Index.ANALYZED));
    // Categories
        Vector<String> cat = rec.getCategories();
        for (int i = 0; i < cat.size(); i++) {
            recdoc.add(new Field("category", cat.get(i), Field.Store.YES,
Field.Index.ANALYZED));
        }
        writer.addDocument(recdoc);
    }
```

Το τελευταίο αποτελούταν από τρία διαφορετικά πεδία, δηλαδή ένα για τον τίτλο, ένα για τα περιεχόμενα του κειμένου και τέλος ένα ακόμα για τις κατηγορίες.

```
    public void addRecord(Record rec) throws CorruptIndexException,
IOException {
        Document recdoc = new Document();
        recdoc.add(new Field("docID", rec.getOhsumedID(), Field.Store.YES,
Field.Index.NOT_ANALYZED_NO_NORMS));
        recdoc.add(new Field("title", rec.getTitle(), Field.Store.YES,
Field.Index.ANALYZED));
        recdoc.add(new Field("text", rec.getAbstruct(), Field.Store.YES,
Field.Index.ANALYZED));
    // Categories
        Vector<String> cat = rec.getCategories();
        for (int i = 0; i < cat.size(); i++) {
            recdoc.add(new Field("category", cat.get(i), Field.Store.YES,
Field.Index.ANALYZED));
        }
        writer.addDocument(recdoc);
    }
```

Με την ολοκλήρωση της δημιουργίας των ευρετηρίων κρίθηκε απαραίτητο να κατασκευαστεί ο κατάλληλος μηχανισμός αναζήτησης στα παραπάνω ευρετήρια.

Searching.

Για την αναζήτηση έγινε και πάλι χρήση του StandardAnalyzer του πακέτου Lucene. Για τον σκοπό αυτό δημιουργήθηκε μια επαναληπτική μέθοδος που έτρεχε για κάθε μια από τις

105 ερωτήσεις και πρόσθετε σε ένα αρχείο τα αποτελέσματα της αναζήτησης τα οποία σε επόμενο στάδιο θα χρησιμοποιηθούν προκειμένου να αξιολογηθούν με το trec_eval.

Γενικευμένος κώδικας αναζήτησης.

```
    private static void search(int question, String queryStr) throws
ParseException, CorruptIndexException, IOException {
        // που βρίσκεται το index
        File indexDir = new File("OHSUMED" + File.separator + "index");
        // Create instance of Directory where index files will be stored
        Directory fsDirectory = FSDirectory.open(indexDir);
        // Create instance of analyzer, which will be used to tokenize the
input data
        Analyzer standardAnalyzer = new
StandardAnalyzer(Version.LUCENE_30);
        String querystr = queryStr.length() > 0 ? queryStr : "lucene";
        Map<String, Float> boostPerField = new HashMap<String, Float>();
        . . .
    (δημιουργούνται τα στοιχεία στο map ανάλογα με την αναζήτηση)
        . . .
        String[] productFields = (στοιχεία και πάλι με βάση το είδος της
αναζήτησης)
        QueryParser parser = new MultiFieldQueryParser(Version. LUCENE_30,
productFields, standardAnalyzer, boostPerField);
        Query q = parser.parse(querystr);
        // Search
        int hitsPerPage = 1000;
        IndexSearcher searcher = new IndexSearcher(fsDirectory, true);
        TopScoreDocCollector collector =
TopScoreDocCollector.create(hitsPerPage, true);
        searcher.search(q, collector);
        ScoreDoc[] hits = collector.topDocs().scoreDocs;
        // Display
        int k = 1;
        for (int i = 0; i < hits.length; ++i) {
            int docId = hits[i].doc;
            Document d = searcher.doc(docId);
            System.out.print(question + "\t1\t" + d.get("docID") + "\t" +
k + "\t" + hits[i].score + "\t" + "search" + "\r\n");
            k++;
        }
    }
```

Τα πειράματα χωρίστηκαν στις παρακάτω κατηγορίες:

Αποτελέσματα με βάση τον Τίτλο και τα περιεχόμενα του κειμένου.

Κατασκευάστηκε ευρετήριο με τα περιεχόμενα του τίτλου και των περιεχόμενων των κειμένων και έπειτα εκτελέστηκαν σε αυτά οι αναζητήσεις.

Total number of relevant documents over all queries	4837
Total number of relevant documents retrieved over all queries	2805
Mean Average Precision (MAP)	0.1652
R-Precision (Precision after R (= num-rel for topic) documents retrieved)	0.2131
Precision after 5 docs retrieved	0.4038
Precision after 10 docs retrieved	0.3486
Precision after 15 docs retrieved	0.3105
Precision after 20 docs retrieved	0.289

Αποτελέσματα με βάση τον Τίτλο, περιεχόμενα του κειμένου και κατηγορίες στις οποίες αυτά ανήκουν.

Τα ευρετήρια που κατασκευάστηκαν είχαν δυο διαφορετικές μορφές. Το πρώτο με δυο πεδία ένα με τους τίτλους και τα περιεχόμενα των κειμένων και ένα με της κατηγορίες τους. Το δεύτερο ευρετήριο αποτελείτε από τρία διαφορετικά πεδία το ένα να έχει τον τίτλο, το άλλο τα περιεχόμενα του κειμένου και το τελευταίο της κατηγορίες στις οποίες αυτό ανήκει. Εκτελώντας τις αναζητήσεις και στα δυο ευρετήρια παρατηρείται μια βελτίωση στα αποτελέσματα του ευρετηρίου με τα τρία πεδία.

	Τίτλος και περιεχόμενα στο ίδιο πεδίο	***Τίτλος και περιεχόμενα διαφορετικά***
Total number of relevant documents over all queries	4837	4837
Total number of relevant documents retrieved over all queries	2896	2917
Mean Average Precision (MAP)	0.177	0.185
R-Precision (Precision after R (= num-rel for topic) documents retrieved)	0.2268	0.2396
Precision after 5 docs retrieved	0.4152	0.4324
Precision after 10 docs retrieved	0.3495	0.3733
Precision after 15 docs retrieved	0.3187	0.3435
Precision after 20 docs retrieved	0.2962	0.3195

Κατηγορίες με περισσότερη βαρύτητα.

Στο πείραμα αυτό έγινε ακόμα προσπάθεια να πριμοδοτηθούν κάποιες κατηγορίες με boost έναντι των υπολοίπων. Έτσι εκτελέστηκε μια σειρά από δοκιμές για διάφορες τιμές του boost και καταλήγοντας στο ότι η τιμή του boost στις κατηγορίες μπορεί να κυμαίνεται κάπου ανάμεσα στις τιμές 1-1.02 και ανάλογα να επιτυγχάνουμε αναζητήσεις με καλύτερο recall ή precision, χάνοντας όμως 1-4 κείμενα στην ανάκτηση. Μετά την απλή προσθήκη του όρου Title (δηλαδή περίπτωση του δεύτερου ευρετηρίου), με εκτέλεση των αναζητήσεων για boost 1.0 στα 3 πεδία παρατηρείται ανάκτηση περισσότερων κειμένων σε σχέση με όλες τις προηγούμενες περιπτώσεις, ακόμα παρατηρείται μια αύξηση στα αποτελέσματα στα χαμηλά επίπεδα του Interpolated Recall - Precision (0.00 - 0.30) η οποία καθώς ανεβαίνουμε γίνεται τελικά χειρότερη σε σχέση με προηγούμενες αναζητήσεις λιγότερων πεδίων. Το Precision όμως είναι πολύ πιο βελτιωμένο από πριν σε όλο το εύρος των κειμένων. Δοκιμάζοντας διάφορα βάρη για τις τιμές του όρου Title έχοντας κρατήσει βάρος για τα Categories=1.01 τα αποτελέσματα δεν δείχνουν να βελτιώνονται, συνεπώς το boost στον τίτλο δεν βοηθάει τόσο. Λαμβάνοντας τώρα όλους τους ελέγχους υπό όψιν θα διακρίναμε δυο περιπτώσεις:

α) ***(T+T)^1+C^1.01*** ή ***(T+T)^1+C^1.005:*** όπου T+T συμβολίζονται ο τίτλος και τα περιεχόμενα του κειμένου ενώ με C συμβολίζονται οι κατηγορίες. Η χρήση τις παρένθεσης γίνεται για να γίνει σαφές ότι τα δύο παραπάνω στοιχεία αποτελούν μέρος του ίδιου πεδίου.

β) ***T+T+C^1.01:*** Το πρώτο T αντιπροσωπεύει τον τίτλο (Title), το δεύτερο τα περιεχόμενα του κειμένου (Text) και τέλος το C αντιπροσωπεύει τις κατηγορίες (Categories).

Για τις 2 υποπεριπτώσεις της α έχουμε τις ίδιες τιμές με μόνη διαφορά το τελικό R-Precision της πρώτης είναι ελαφρά καλύτερο από αυτό της δεύτερης, αλλά η δεύτερη με την σειρά της έχει σε κάποιες επί μέρους τιμές καλύτερα αποτελέσματα (Interpolated Recall - Precision Averages at 0.50 καθώς και Precision at 20-500 doc).

τύπος	***(T+T)^1+C^1.01***	***(T+T)^1+C^1.005***	***T+T+C^1.01***
Queryid (Num):	105	105	105
Total number of documents over all queries			
Retrieved:	103972	103972	103972
Relevant:	4836	4836	4836
Rel_ret:	2894	2894	2917
Interpolated Recall - Precision Averages:			
at 0.00	0.6514	0.6466	0.6999
at 0.10	0.4287	0.4288	0.4589
at 0.20	0.326	0.326	0.3693
at 0.30	0.2659	0.2659	0.2837
at 0.40	0.2033	0.2034	0.2004
at 0.50	0.1519	0.1521	0.1517
at 0.60	0.0838	0.0838	0.0773
at 0.70	0.0478	0.0478	0.0408
at 0.80	0.0204	0.0204	0.0154
at 0.90	0.0051	0.0051	0.0039
at 1.00	0.0006	0.0006	0.0002
Average precision (non-interpolated) over all rel docs			
	0.1763	0.1762	0.185
Precision:			
At 5 docs:	0.4114	0.4114	0.4324
At 10 docs:	0.3495	0.3495	0.3733
At 15 docs:	0.3181	0.3181	0.3422
At 20 docs:	0.2948	0.2952	0.3205
At 30 docs:	0.2562	0.2568	0.279
At 100 docs:	0.1489	0.149	0.1523
At 200 docs:	0.0954	0.0955	0.0971
At 500 docs:	0.0486	0.0487	0.0496
At 1000 docs:	0.0276	0.0276	0.0278
R-Precision (precision after R (= num_rel for a query) docs retrieved):			
Exact:	0.2258	0.2256	0.2403

Οπότε ανάλογα με το τι θέλουμε θα μπορούσαμε να επιλέξουμε μια από τις 2 περιπτώσεις. Αν και η T+T+C^1.01 είναι η μόνη δοκιμή που κατάφερε να ανακτήσει τα περισσότερα σχετικά κείμενα, για αυτό και έχει τέτοια ευστοχία.

Ανάδραση Συνάφειας (Relevance Feedback).

Τα ερωτήματα της αναζήτησης επεκτάθηκαν με τέτοιο τρόπο ώστε να χρησιμοποιηθεί μια ανάδραση σχετικά με την συνάφεια των κατηγοριών. Σε αυτό το στάδιο, αρχικά τα ερωτήματα υποβλήθηκαν στην βάση και από τα αποτελέσματα τους χρησιμοποιήθηκαν οι κατηγορίες, οι οποίες προηγουμένως είχαν αξιολογηθεί με βάση την συχνότητα εμφάνισης τους. Οι κατηγορίες αυτές προσαρτήθηκαν στο ερώτημα το οποίο υποβλήθηκε και πάλι στην βάση. Ο μηχανισμός αυτός λειτούργησε σαν έναν εσωτερικό μηχανισμό ανάδρασης πριμοδοτώντας τις καλύτερες και πιο συναφείς κατηγορίες.

Αναλυτικότερα, σχετικά με την λειτουργία του παραπάνω μηχανισμού. Αρχικά, υποβάλλεται ένα ερώτημα στην OHSUMED και έστω ότι ανακτά τα κείμενα δ1, δ2, ..., δN. Κρατάμε τα k πρώτα (δ_1, δ_2, ...,δ_k) κείμενα της απάντησης (με k παραμετροποιημένο). Τα κείμενα αυτά έχουν και τις κατηγορίες τους δηλαδή κάθε κείμενο έχει τρία πεδία (Τίτλο, Κείμενο, Κατηγορία). Υπολογίζουμε τις κατηγορίες με την υψηλότερη συχνότητα εμφάνισης στα κείμενα δ_1, δ_2, ...,δ_k. Τις κατηγορίες αυτές τις ενώνουμε με το αρχικό ερώτημα και το επανυποβάλλουμε στη βάση. Η αξιολόγηση των αποτελεσμάτων θα γίνει από τα επεκταμένα ερωτήματα.

Παράδειγμα.

Έστω k=5. Υποβάλλουμε το ερώτημα Q το οποίο ανακτά τα κείμενα:

d_1 --> c_1 c_2 c_4 c_6
d_2 --> c_2 c_3 c_5 c_6
d_3 --> c_1 c_4 c_5 c_6
d_4 --> c_3 c_4 c_9 c_{11}
d_5 --> c_2 c_5 c_6 c_7 c_8
d_6
d_7
κτλ.

Από τα 5-πρώτα ανακτηθέντα κείμενα υπολογίζουμε τις συχνότητες των όρων -κατηγοριών

c_1 --> 2
c_2 --> 3
c_3 --> 2
c_4 --> 3
c_5 --> 3
c_6 --> 4
c_7 --> 1
c_8 --> 1
c_{11} --> 1

Έπειτα, επιλέγουμε τις κατηγορίες με την υψηλότερη συχνότητα εμφάνισης στα k-πρώτα κείμενα της απάντησης {c_6, c_4, c_5, c_2} και κατασκευάζουμε ένα νέο ερώτημα που περιλαμβάνει το αρχικό Q και τους όρους {c_6, c_4, c_5, c_2} δηλαδή

Qnew = Q + {c_6, c_4, c_5, c_2} και αξιολογούμε την ανάκτηση από τα νέα ερωτήματα. Στην αξιολόγηση να τυπώνουμε και το μέτρο MAP (Mean Average Precision).

Αλγόριθμος της μεθόδου.

Για κάθε εγγραφή της βάσης

παίρνω τις κατηγορίες, έστω {C1,C2,C3,C4,C5,C6,C7,C8};

βρίσκω το πόσες πρέπει να αλλάξω, έστω το 30%;

Πλήθος κατηγοριών που θα αλλάξουν=0,3*8=⌈2,4⌉=3 (άνω φράγμα);

Τρεις κατηγορίες από τις {C1,C2,C3,C4,C5,C6,C7,C8} επιλέγονται τυχαία: έστω ότι είναι οι κατηγορίες: {C2, C5, C7};

επιλέγω από όλες μείον αυτές που ήδη είχα τον αριθμό αυτών που θέλω να αλλάξω; Δηλαδή από όλες τις κατηγορίες (όλων των κειμένων της συλλογής)επιλέγω τυχαία τρεις νέες κατηγορίες οι οποίες είναι διαφορετικές από τις {C1,C2,C3,C4,C5,C6,C7,C8}; Έστω για παράδειγμα ότι επέλεξα τυχαία τις: {C13, C20, C55};

αντικαθιστώ το απαιτούμενο ποσοστό παλιών με τις νέες που βρήκα;

το αποτέλεσμα θα είναι {C1,C3,C4,C6,C8,C13,C20,C55};

ανανεώνω την εγγραφή μου;

Λογική Υλοποίησης.

Κατά την υλοποίηση χρειάστηκε πριν την αναζήτηση μας να προστεθεί μια προκαταρτική αναζήτηση έτσι ώστε να εντοπιστούν οι απαραίτητες κατηγορίες έπειτα αυτές προσαυξήθηκαν στο ερώτημα και τέλος εκτελέστηκε η αναζήτηση με βάση το νέο ερώτημα.

Στην διαδικασία της προκαταρτικής αναζήτησης για κάθε ένα επιστρεφόμενο έγγραφο πήραμε όλα τα στοιχεία του στο πεδίο κατηγορίες και τα περάσαμε μέσα από ένα μηχανισμό αξιολόγησης συχνοτήτων, προκειμένου να εντοπιστούν αυτά με τις περισσότερες εμφανίσεις. Δηλαδή, οι κατηγορίες που έχουν λογική συνάφεια με το αρχικό ερώτημα. Η προκαταρτική λοιπόν αναζήτηση σε αντίθεση με την κανονική δεν προσθέτει τα αποτελέσματα σε ένα αρχείο προκειμένου να αξιολογηθούνε με το trec_eval, αλλά αντίθετα επιστρέφει ένα αντικείμενο τύπου String που περιέχει όλες εκείνες τις καλά αξιολογημένες κατηγορίες. Η έξοδος της προκαταρτικής αναζήτησης χρησιμοποιείτε έπειτα ως είσοδος στην κανονική αναζήτηση, που δημιουργεί τα αρχεία τα οποία και αξιολογούνται από το εργαλείο trec_eval.

Ο μηχανισμός αυτός δοκιμάστηκε για διαφορετικά πλήθη ανακτηθέντων κειμένων καθώς και για διάφορες συχνότητες εμφάνισης κειμένων. Τα καλύτερα αποτελέσματα επιτεύχθηκαν για την συλλογή κατηγοριών που είχαν συχνότητα εμφάνισης μεγαλύτερη ή ίση του τρία στα πέντε πρώτα κείμενα.

Total number of relevant documents over all queries	4837
Total number of relevant documents retrieved over all queries	3064
Mean Average Precision (MAP)	0.2022
R-Precision (Precision after R (= num-rel for topic) documents retrieved)	0.2512
Precision after 5 docs retrieved	0.459
Precision after 10 docs retrieved	0.3905
Precision after 15 docs retrieved	0.3524
Precision after 20 docs retrieved	0.33

Ανοχή στην χρήση εσφαλμένων κατηγοριών (Fault tolerance Analysis).

Δημιουργήθηκε πρόγραμμα το οποίο χρησιμοποιώντας την βάση της oshumed, αντικαθιστούσε με τυχαιοποιημένο τρόπο το απαιτούμενο ποσοστό κατηγοριών με άλλες κατηγορίες που επιλέγονταν τυχαία από της κατηγορίες που ήταν διαθέσιμες. Έτσι κατασκευάστηκαν πέντε νέες βάσεις η κάθε μια από τις οποίες είχε ένα διαφορετικό ποσοστό τυχαιοποιημένων κατηφοριών. Με χρήση αυτών των νέων βάσεων δημιουργήθηκαν νέα ευρετήρια. Σε όλα τα ευρετήρια έγινε χρήση τριών πεδίων ένα που αντιπροσωπεύει τον τίτλο, ένα για το περιεχόμενο του κειμένου και τέλος ένα για τις κατηγορίες αυτού. Στο πείραμα αυτό χρησιμοποιήθηκε και η ανάδραση συνάφειας με βάση τα αποτέλεσμα των καλύτερων τιμών που παρατηρήθηκαν στο προηγούμενο πείραμα.

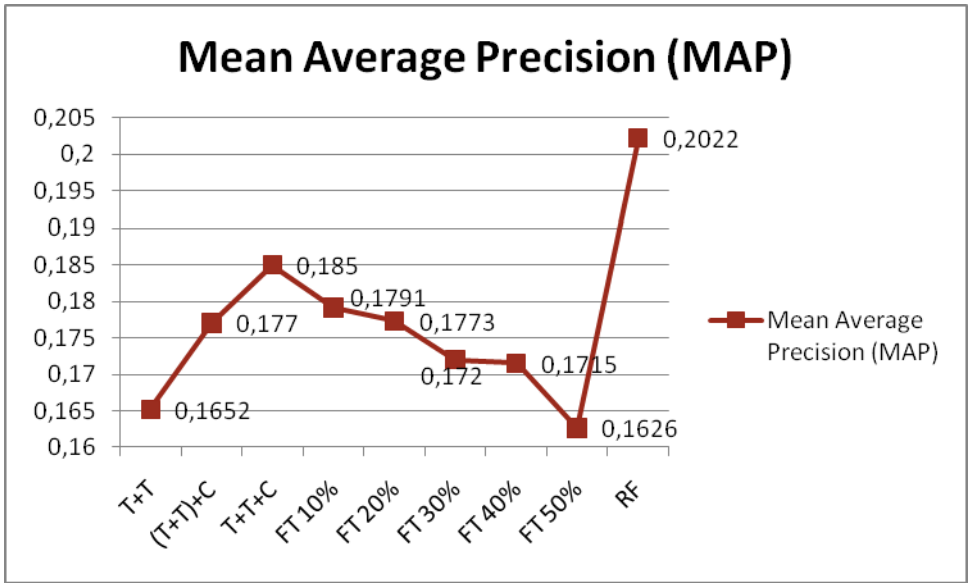
	10%	***20%***	***30%***	***40%***	***50%***
Total number of relevant documents over all queries	4837	4837	4837	4837	4837
Total number of relevant documents retrieved over all queries	2877	2863	2840	2822	2772
Mean Average Precision (MAP)	0.1791	0.1773	0.172	0.1715	0.1626
R-Precision (Precision after R (= num-rel for topic) documents retrieved)	0.232	0.2301	0.2304	0.2211	0.2126
Precision after 5 docs retrieved	0.4343	0.44	0.4267	0.4514	0.4286
Precision after 10 docs retrieved	0.3695	0.3667	0.3629	0.3695	0.3495
Precision after 15 docs retrieved	0.339	0.3371	0.3283	0.3289	0.3181
Precision after 20 docs retrieved	0.311	0.3186	0.311	0.31	0.2943

Παρατηρείται ότι σε σχέση με την απλή αναζήτηση έχουμε καλύτερα αποτελέσματα σε όλες τις περιπτώσεις πλην της 50% σε σχέση πάντα με την απλή αναζήτηση χωρίς την χρήση κατηγοριών. Όλα τα πειράματα έγιναν με επισύναψη των κατηγοριών στο αρχικό query όταν μετά από μια πρωταρχική αναζήτηση προσθέταμε στο αρχικό Q (query) τις κατηγορίες που μας έδιναν τα πέντε πρώτα ανακτημένα κείμενα και είχαν συχνότητα μεγαλύτερη ή ίση του τρία. Αξιοσημείωτο είναι το γεγονός ότι στην περίπτωση του 20% είχαμε βελτίωση στα αποτελέσματα από ότι του 10% αυτό μπορεί να εξηγηθεί ως επόμενο της τυχαίας αλλαγής των κατηγοριών δηλαδή έτυχε να αλλαχτούν μη-σημαντικές κατηγορίες για τις αναζητήσεις μας.

Γενικά, ο μηχανισμός με τον οποίο επισυνάπτονται οι κατηγορίες στο query με βάση δηλαδή την συχνότητα φαίνεται να λειτουργεί ιδιαίτερα αποτελεσματικά γιατί ακόμα και αν βάλουμε λάθος κατηγορίες αυτό δεν σημαίνει ότι θα τύχει σε όλα τα σχετικά κείμενα να συμπέσουν οι ίδιες λάθος κατηγορίες, οπότε και τα αποτελέσματα μας δεν χαλάνε ιδιαίτερα. Έτσι έχουμε κατασκευάσει έναν ιδιαίτερα καλό μηχανισμό που θα μπορεί να λειτουργήσει και σε κείμενα τα οποία θα έχουν υποστεί αυτόματη κατηγοριοποίηση, της οποίας το ποσοστό επιτυχίας θα είναι της τάξης του 70%, για να είμαστε μέσα σε ανεκτά όρια σφαλμάτων πάντα.

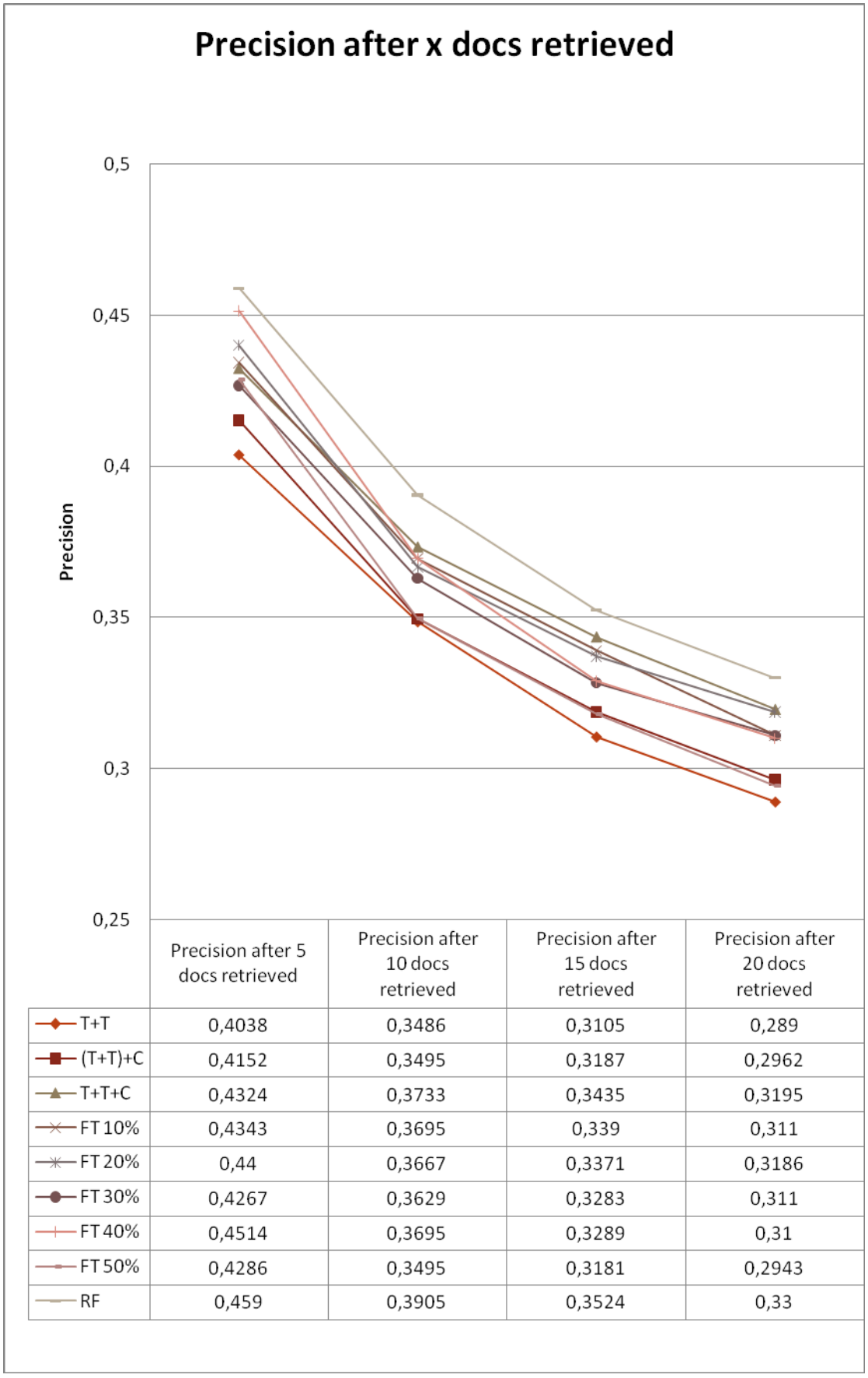
Συμπεράσματα.

Η χρήση των κατηγοριών σε μια αναζήτηση μπορεί να βελτιώσει τα αποτελέσματα της εκάστοτε αναζήτησης ενώ ακόμα και η μερικώς εσφαλμένη επιλογή κατηγοριών σε ποσοστό ακόμα και 40% μπορεί να μην επηρεάσει αρνητικά τις αναζητήσεις μας. Αξιοσημείωτη είναι η βελτίωση που παρατηρείτε με τον μηχανισμό ανάδρασης.



Πίνακας Δεδομένων.

Περιεχόμενα	***Συμβολισμός***	***MAP***
Τίτλος και απόσπασμα	T+T	0.1652
Τίτλος, απόσπασμα (μαζί) και κατηγορίες	(T+T)+C	0.177
Τίτλος, απόσπασμα, κατηγορίες (διαφορετικά πεδία)	T+T+C	0.185
Εσφαλμένες κατηγορίες σε ποσοστό 10%	FT 10%	0.1791
Εσφαλμένες κατηγορίες σε ποσοστό 20%	FT 20%	0.1773
Εσφαλμένες κατηγορίες σε ποσοστό 30%	FT 30%	0.172
Εσφαλμένες κατηγορίες σε ποσοστό 40%	FT 40%	0.1715
Εσφαλμένες κατηγορίες σε ποσοστό 50%	FT 50%	0.1626
Ανάδραση συνάφειας (relevance feedback)	RF	0.2022



	Precision after 5 docs retrieved	Precision after 10 docs retrieved	Precision after 15 docs retrieved	Precision after 20 docs retrieved
T+T	0,4038	0,3486	0,3105	0,289
(T+T)+C	0,4152	0,3495	0,3187	0,2962
T+T+C	0,4324	0,3733	0,3435	0,3195
FT 10%	0,4343	0,3695	0,339	0,311
FT 20%	0,44	0,3667	0,3371	0,3186
FT 30%	0,4267	0,3629	0,3283	0,311
FT 40%	0,4514	0,3695	0,3289	0,31
FT 50%	0,4286	0,3495	0,3181	0,2943
RF	0,459	0,3905	0,3524	0,33

MetaMap.

Το εργαλείο MetaMap, δημιουργήθηκε από τον Dr. Alan (Lan) Aronson στο National Library of Medicine (NLM) και είναι ένα ιδιαίτερα παραμετροποιήσιμο εργαλείο με σκοπό την χαρτογράφηση βιοϊατρικού κειμένου στην UMLS Metathesaurus ή ισοδύναμα να εντοπίσει Metathesaurus έννοιες που περιέχονται μέσα στο κείμενο. Το εργαλείο αυτό χρησιμοποιεί μια αναλυτική προσέγγιση που βασίζεται σε συμβολική ή επεξεργασία φυσικής γλώσσας μέσα από υπολογιστικές και γλωσσολογικές τεχνικές. Εκτός του ότι χρησιμοποιείται για ανάκτηση πληροφοριών και εφαρμογές εξόρυξης γνώσης το MetaMap είναι μια από τις θεμελιώδεις εφαρμογές του Medical Text Indexer (MTI) η οποία χρησιμοποιείται είτε για ημιαυτόματη είτε για πλήρως αυτοματοποιημένη ευρετηριοποίηση βιοϊατρικής βιβλιογραφίας στην Εθνική Ιατρική Βιβλιοθήκη (NLM).

Στην περίπτωση μας χρησιμοποιήθηκε προκειμένου να εξαχθούν με αυτοματοποιημένο μηχανισμό κατηγορίες στα κείμενα μας, είτε αυτά αποτελούσαν δεδομένα της βάσης, είτε ερωτήματα προς αυτήν. Οπότε ως είσοδο δεδομένων είχαμε τα ερωτήματα της βάσης και η έξοδος επιλέχθηκε να είναι xml αρχείο, μια από τις πολλές επιλογές του συγκεκριμένου εργαλείου. Το αρχείο αυτό επεξεργάστηκε κατάλληλα μέσα από κατάλληλο πρόγραμμα που δημιουργήθηκε στην γλώσσα java έτσι ώστε να εξαχθούν/φιλτραριστούν οι απαιτούμενες κατηγορίες.

Λειτουργία και Χρήση.

Το εργαλείο αυτό εγκαταστάθηκε και λειτούργησε στον server του εργαστηρίου Επεξεργασίας Πληροφοριών. Προκειμένου να τρέξει ήταν απαραίτητο να ενεργοποιηθούν οι servers Tagger και WSD που τρέχουν αυτόματα στο παρασκήνιο με την εκκίνηση τους:

1. *The SKR/Medpost Part-of-Speech Tagger Server:*

```
% ./bin/skrmedpostctl start
```

2. *Word Sense Disambiguation (WSD) Server (optional):*

```
% ./bin/wsdserverctl start
```

Αντίστοιχα όταν θέλουμε να σταματήσουμε τους servers εκτελούμε τις εντολές:

1. *The SKR/Medpost Part-of-Speech Tagger Server*

```
% ./bin/skrmedpostctl stop
```

2. *Word Sense Disambiguation (WSD) Server (optional)*

```
% ./bin/wsdserverctl stop
```

Προκειμένου τώρα να βρούμε τις κατηγορίες ενός κειμένου με χρήση του παραπάνω εργαλείου δίνουμε την εντολή:

```
public_mm/bin/metamap09v2 -% format input.txt output.txt
```

έτσι με είσοδο το αρχείο input.txt που περιέχει την ερώτηση ή το κείμενο του οποίου θέλουμε να εντοπίσουμε τις κατηγορίες παράγουμε το αρχείο output.txt. Δεδομένου ότι το input.txt περιέχει μόνο μια φράση για μετατροπή και όχι πολλές φράσεις διαχωρισμένες με

αλλαγές γραμμών το output.txt είναι ένα αρχείο xml έτοιμο για επεξεργασία. Αλλιώς το αρχείο output.txt περιέχει πολλά xml αρχεία ενοποιημένα σε ένα.

Παράδειγμα Αρχείου Εισαγωγής.

```
Heart attack.


Lung cancer.
```

Ανάλυση σημασίας αρχείου εισαγωγής.

Το αρχείο εισαγωγής είναι ένα αρχείο κειμένου που περιέχει τις λέξεις ή φράσεις των οποίων θέλουμε να βρούμε τους ιατρικούς τους όρους. Όταν έχουμε πολλές ανεξάρτητες φράσεις για τις οποίες θέλουμε να εκτελεστεί το πρόγραμμα τότε τις γράφουμε αφήνοντας μεταξύ τους κενές γραμμές, έτσι το output αποκτά την παρακάτω μορφή.

Παράδειγμα Αρχείου Εξαγωγής.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MMO PUBLIC "-//NLM//DTD MetaMap Machine Output//EN"
"http://ii-public.nlm.nih.gov/DTD/MMOtoXML_v2.dtd">
<MMOlist>
  <MMO>

    . . . XML output for Heart attack. . . .


  </MMO>
</MMOlist>


<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MMO PUBLIC "-//NLM//DTD MetaMap Machine Output//EN"
"http://ii-public.nlm.nih.gov/DTD/MMOtoXML_v2.dtd">
<MMOlist>
  <MMO>

    . . . XML output for Lung cancer. . . .


  </MMO>
</MMOlist>
```

Ανάλυση σημασίας xml αρχείου εξαγωγής.

<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE MMOlist PUBLIC "-//NLM//DTD MetaMap Machine Output//EN" "http://ii- public.nlm.nih.gov/DTD/MMOtoXML.dtd">	Εισαγωγικά στοιχεία
<MMOlist> <MMO>	Περιγραφή ενός Metamap Machine Output
<Args> <Command>MetaMap -Z 08 -% format h</Command> <Options Count="4"> <Option> <OptName>mm_data_year</OptName> <OptValue>08</OptValue> </Option> <Option> <OptName>XML</OptName> <OptValue>format</OptValue> </Option> <Option> <OptName>infile</OptName> <OptValue>h</OptValue> </Option> <Option> <OptName>outfile</OptName> <OptValue>h.out</OptValue> </Option> </Options> </Args>	Παράμετροι και στοιχεία της εισόδου (Arguments)
<AAs Count="0" />	Όλα τα δεδομένα που σχετίζονται με τα Ακρωνύμια
<Negations Count="0" />	Όλα τα δεδομένα που σχετίζονται με αρνήσεις
<Utterances Count="1"> <Utterance> <PMID>00000000</PMID> <Location>tx</Location> <SeqNo>1</SeqNo> <UText>heart.</UText> <UStartPos>0</UStartPos> <USpanLen>7</USpanLen>	Utterances: Όλες οι εκφράσεις PMID: PubMed ID
<Phrases Count="1"> <Phrase> <PText>heart.</PText>	Φράση από τα Utterances
<Tags Count="2"> <Tag> <Type>head</Type> <LexMatch>heart</LexMatch> <InputMatch>heart</InputMatch> <POS>noun</POS> <Tokens Count="1"> <Token>heart</Token> </Tokens> </Tag> <Tag> <Type>punc</Type> <InputMatch>.</InputMatch> <Tokens Count="0" /> </Tag>	LexMatch: Λεκτικά σύνολα που αντιστοιχούν με την συντακτική μονάδα. InputMatch: Λέξεις που εισάγαμε και κατασκευάζουν την συντακτική μονάδα. POS: Μέρος του λόγου Tokens: κομμάτια του παραπάνω τύπου.

</Tags>	
<PStartPos>0</PStartPos> <PSpanLen>6</PSpanLen>	Σημείο στο οποίο αρχίζει η φράση, και μήκος αυτής
<Candidates Count="2"> <Candidate>	Υποψήφια στοιχεία για κατηγορίες.
<NegScore>-1000</NegScore> <UMLSCUI>C0018787</UMLSCUI> <UMLSConcept>Heart</UMLSConcept> <UMLSPreferred>Heart</UMLSPreferred>	Το σκορ με το οποίο αξιολογήθηκαν UMLSCUI: Unified Medical Language System Concept Unique Identifier (μοναδικός κωδικός για το συγκεκριμένη λέξη ή φράση στην Ιατρική)
<MatchedWords Count="1"> <MatchedWord>heart</MatchedWord> </MatchedWords>	Ταίριασμα της λέξης
<STs Count="1"> <ST>bpoc</ST> </STs>	Ο σημασιολογικός τύπος (-ες) του υποψηφίου
<MatchMaps Count="1"> <MatchMap> <TWMatchPosS>1</TWMatchPosS> <TWMatchPosE>1</TWMatchPosE> <CWMatchPosS>1</CWMatchPosS> <CWMatchPosE>1</CWMatchPosE> <Variation>0</Variation> </MatchMap> </MatchMaps> <IsHead>yes</IsHead> <IsOverMatch>no</IsOverMatch>	Τρόπος-χάρτης ταιριάσματος
<Sources Count="20"> <Source>AIR</Source> <Source>HL7V2.5</Source> <Source>ICNP</Source> <Source>LCH</Source> <Source>LNC</Source> <Source>MSH</Source> <Source>MTH</Source> <Source>NCI</Source> <Source>OMIM</Source> <Source>PSY</Source> <Source>RCD</Source> <Source>SNM</Source> <Source>SNOMEDCT</Source> <Source>UWDA</Source> <Source>CCPSS</Source> <Source>SNMI</Source> <Source>AOD</Source> <Source>CSP</Source> <Source>BI</Source> <Source>PNDS</Source> </Sources>	Το λεξιλόγιο UMLS στο οποίο διαπιστώθηκε ο όρος
<Spans Count="1"> <Span> <StartPos>0</StartPos> <SpanLen>5</SpanLen> </Span> </Spans> </Candidate>	Που εκτείνεται το στοιχείο
<Candidate> . . .	Υπόλοιπα candidates

</Candidate> </Candidates>	
<Mappings Count="2"> <Mapping> <MapNegScore>-1000</MapNegScore> <Candidates Count="1"> <Candidate> <NegScore>-1000</NegScore> <UMLSCUI>C1281570</UMLSCUI> <UMLSConcept>Heart</UMLSConcept> <UMLSPreferred>Entire heart</UMLSPreferred> <MatchedWords Count="1"> <MatchedWord>heart</MatchedWord> </MatchedWords> <STs Count="1"> <ST>bpoc</ST> </STs>	Χαρτογράφηση των υποψήφιων στοιχείων
<MatchMaps Count="1"> <MatchMap> <TWMatchPosS>1</TWMatchPosS> <TWMatchPosE>1</TWMatchPosE> <CWMatchPosS>1</CWMatchPosS> <CWMatchPosE>1</CWMatchPosE> <Variation>0</Variation> </MatchMap> </MatchMaps> <IsHead>yes</IsHead> <IsOverMatch>no</IsOverMatch> <Sources Count="2"> <Source>MTH</Source> <Source>SNOMEDCT</Source> </Sources>	
<Spans Count="1"> <Span> <StartPos>0</StartPos> <SpanLen>5</SpanLen> </Span> </Spans> </Candidate> </Candidates> </Mapping>	
<Mapping> . . . </Mapping> </Mappings>	Χαρτογράφηση των υπόλοιπων στοιχείων.
</Phrase> </Phrases> </Utterance> </Utterances> </MMO> </MMOlist>	Τέλος των παραπάνω χαρακτηριστικών

Από το παραπάνω XML ως κατηγορίες χρησιμοποιήθηκαν τα UMLSPreferred των Candidates και ως σκορ αυτών όπου αυτό κρίθηκε απαραίτητο το NegScore κατ' απόλυτη τιμή.

Τρόπος Αξιοποίησης.

Το εργαλείο χρησιμοποιήθηκε για να παράγουμε με αυτοματοποιημένο τρόπο κατηγορίες στα ερωτήματα τα οποία είχαμε να θέσουμε στην βάση μας. Προκειμένου να επιτευχθεί αυτό πριν θέσουμε κάθε ερώτηση στην βάση χρησιμοποιώντας το metamap βρίσκαμε της κατηγορίες στις οποίες αυτή ανήκει. Μετά φτιάχναμε ένα σύνθετο ερώτημα και το εκτελέσαμε στην βάση.

Υλοποίηση 1.

Σε αυτή την φάση απλά προστέθηκαν στο query με απλό concatenate οι κατηγορίες των που εντοπίστηκαν. Τα στοιχεία που χρησιμοποιήσαμε ήταν το Query (δηλαδή ότι είχαμε και πριν) και το candidatePreferred (ο όρος που προτιμάται για μια συγκεκριμένη κατηγορία) καθώς επίσης και το sources[iii]

Παρατηρήσεις.

Σε κάποιες περιπτώσεις ειδικά όταν δεν είχαμε κανένα αποτέλεσμα με τις προηγούμενες περιπτώσεις τώρα έχουμε αποτελέσματα αλλά σε άλλες που αποτελούν και την πλειοψηφία έχουμε χειρότερα το οποίο έχει ως αποτέλεσμα τα συνοπτικά αποτελέσματα να είναι χειρότερα. Τα αποτελέσματα αυτά οφείλονται και στο γεγονός ότι έχουμε αποτελέσματα σε 103 αντί για 105 ερωτήσεις, οι 2 που χάνει είναι αυτές που έχουν 0 hits στις αναζητήσεις ενώ κανονικά υπάρχουν κείμενα που απαντάνε σε αυτές απλά ο μηχανισμός αδυνατεί να τα εντοπίσει.

Υλοποίηση 2.

Στην υλοποίηση αυτή έγινε ότι και στην προηγούμενη με μόνη διαφορά ότι αυτή την φορά δεν κρατήθηκαν τα sources, μιας και η μορφή τους δεν ήταν ιδιαιτέρα χρηστική όσον αφορά τα περιεχόμενα των ευρετηρίων μας. Όπως ήταν αναμενόμενο παρατηρήθηκε μικρή βελτίωση σε όλα σχεδόν τα αποτελέσματα σε σχέση με την προηγούμενη. Έτσι έχουμε καλύτερες τιμές στις 2598 τιμές έναντι των χειρότερων που είναι 53. Το συγκεκριμένο αποτέλεσμα τεκμηριώνει και πειραματικά ότι ο όρος sources δεν βοηθάει, οπότε και καταργήθηκε.

Υλοποίηση 3.

Παραλλαγή 1.

Στην τρίτη και τελευταία υλοποίηση έγινε μια ακόμα προσπάθεια βελτίωσης των αποτελεσμάτων. Προκειμένου να επιτευχθεί αυτό έγινε μια σύνθετη αναζήτηση έτσι ώστε να είναι πιο συγκεκριμένη σε αυτό που θέλουμε. Δηλαδή το αρχικό query αναζητήθηκε στο πεδίο text και title και οι κατηγορίες οι οποίες βρήκαμε για αυτό αναζητήθηκαν μόνο στο πεδίο κατηγοριών. Με αυτόν τον τρόπο ακόμα επιτεύχθηκε να έχουμε απαντήσεις και στις 105 ερωτήσεις. Έτσι τα αποτελέσματα ήταν καλύτερα από όλες τις παραπάνω υλοποιήσεις. Στα επιμέρους λοιπόν αποτελέσματα έχουμε στις 2651 μετρήσεις παρατηρείτε να υπάρχει αύξηση σε 1146 και από αυτές μείωση σε 724 ενώ παραμένουν σταθερές οι 781. Αλλά ακόμα δεν επιτεύχθηκε τα συνοπτικά να είναι καλύτερα από την αντίστοιχη υλοποίηση που είχαμε πριν την χρήση του metamap.

Στα πλαίσια αυτής της προσπάθειας έγιναν και κάποιες ακόμα δοκιμές μήπως και τελικά επιτευχθεί η αναμενόμενη βελτίωση. Σε σχέση με τις προηγούμενες τιμές της αναζήτησης

παρατηρείται μια βελτίωση αλλά ακόμα δεν έχουμε βελτίωση σε σχέση με την χρήση ή όχι του metamap, ειδικά όσον αφορά τα συνολικά αποτελέσματα.

Παραλλαγή 2.

Προσπαθήσαμε να αναζητήσουμε το αρχικό query σε όλα τα πεδία, δηλαδή και στο πεδίο των κατηγοριών, ενώ στο πεδίο των κατηγοριών αναζητήθηκαν εκτός από το αρχικό query και οι κατηγορίες οι οποίες είχαμε αντλήσει από το metamap. Δυστυχώς, τα αποτελέσματα που πήραμε ήταν χειρότερα από αυτά που δεν είχαν αναζήτηση της αρχικής ερώτησης στο πεδίο των κατηγοριών, μιας και αποδείχτηκε ότι προσθέτουν φιλτράρισμα σε στοιχεία που δεν θα έπρεπε.

Παραλλαγή 3.

Τα προηγούμενα πειράματα μας οδήγησαν στο συμπέρασμα ότι η γενικότητα των κατηγοριών αποπροσανατολίζει κατά κάποιο τρόπο την αναζήτηση μας, έτσι έγινε μια προσπάθεια προκειμένου οι κατηγορίες αυτές να χρησιμοποιηθούν με κάποιο βάρος, προκειμένου να μην επηρεάζουν τα αποτελέσματα μας περισσότερο από όσο θα θέλαμε. Με αυτό τον τρόπο θα μπορούσαμε να φιλτράρουμε μόνο τις σημαντικότερες από τις μέτρια σημαντικές κατηγορίες. Οπότε στον μηχανισμό αυτό χρησιμοποιήσαμε αναζήτηση της αρχικής ερώτησης στα πεδία το κειμένου (τίτλος και περιεχόμενα) και στο πεδίο των κατηγοριών έγινε αναζήτηση των κατηγοριών του metamap με βάρος το score που είχαμε βρει πολλαπλασιασμένο με -0.001 έτσι ώστε να βρίσκεται στο διάστημα [0, 1]. Σε αυτή την περίπτωση παρατηρήθηκε μικρή βελτίωση στα συνολικά αποτελέσματα, ιδιαίτερα επιτεύχθηκε η καλύτερη ως τώρα Rel_ret τιμή, δηλαδή πετύχαμε να αντλήσουμε τα περισσότερα σχετικά κείμενα στα ανακτημένα μας, αλλά σε σχέση με τις προηγούμενες περιπτώσεις δεν έχουμε καλύτερα αποτελέσματα.

Αποτελέσματα.

1. Προσθήκη του candidatePreferred

2. candidatePreferred χωρίς τον όρο sources

3.1. query->title+text, candidatePreferred ->categories

3.2. query->title+text+categories, candidatePreferred ->categories

3.3. categories^score

Υλοποίηση	***Απλή***	***1.***	***2.***	***3.1.***	***3.2***	***3.3.***
Queryid (Num):	105	103	103	105	105	105
Total number of documents over all queries						
Retrieved:	103972	103000	102055	104330	103917	104330
Relevant:	4836	4694	4694	4836	4836	4836
Rel_ret:	2917	2669	2737	2874	2844	2928
Interpolated Recall - Precision Averages:						
at 0.00	0.6997	0.2566	0.5691	0.6031	0.5994	0.613
at 0.10	0.4591	0.2255	0.4065	0.441	0.4319	0.4521
at 0.20	0.3693	0.1815	0.3165	0.3223	0.3195	0.3365
at 0.30	0.2833	0.1576	0.2599	0.2545	0.2573	0.2612
at 0.40	0.2006	0.1222	0.1954	0.1807	0.1831	0.1889
at 0.50	0.1515	0.0955	0.1408	0.1229	0.1276	0.1295
at 0.60	0.0778	0.0566	0.0848	0.0722	0.074	0.0727
at 0.70	0.0406	0.0263	0.0349	0.0362	0.0369	0.0372
at 0.80	0.0154	0.0112	0.0152	0.0119	0.0121	0.0131
at 0.90	0.0039	0.0013	0.0021	0.0019	0.0018	0.0018
at 1.00	0.0002	0.0002	0.0002	0.0002	0.0002	0.0001
Average precision (non-interpolated) over all rel docs						
	0.185	0.0919	0.166	0.1667	0.1669	0.1714
Precision:						
At 5 docs:	0.4324	0.1204	0.3748	0.4229	0.4095	0.4305
At 10 docs:	0.3733	0.1524	0.334	0.3524	0.3514	0.3638
At 15 docs:	0.3435	0.1638	0.3016	0.3162	0.3181	0.3321
At 20 docs:	0.3195	0.1597	0.2796	0.2881	0.2905	0.2995
At 30 docs:	0.2794	0.1524	0.2411	0.2533	0.2559	0.2594
At 100 docs:	0.1523	0.1076	0.1392	0.1434	0.1431	0.145
At 200 docs:	0.0971	0.0776	0.09	0.0924	0.0912	0.095
At 500 docs:	0.0497	0.0441	0.0463	0.0477	0.047	0.0486
At 1000 docs:	0.0278	0.0259	0.0266	0.0274	0.0271	0.0279
R-Precision (precision after R (= num_rel for a query) docs retrieved):						
Exact:	0.2396	0.1392	0.2096	0.2134	0.2164	0.2176

ImageCLEF 2009 medical retrieval.

Η συλλογή των δεδομένων.

Πρόκειται για μια συλλογή παρόμοια με αυτήν του 2008 αλλά με μεγαλύτερο αριθμό εικόνων. Η συλλογή περιέχει όλες τις εικόνες από άρθρα που δημοσιεύτηκαν στο «Radiology and Radiographics» μαζί με την περιγραφή της εικόνας και έναν σύνδεσμο html προς το συγκεκριμένο άρθρο. Πρόκειται για ένα σύνολο πάνω από 70.000 εγγραφών. Η συλλογή που χρησιμοποιήθηκε για τα πειράματα μας, είχε μια ακόμα βελτιστοποίηση, δηλαδή όλες οι σχετικές εικόνες που περιείχαν την ίδια περιγραφή είχαν ενοποιηθεί ως εγγραφές προσθέτοντας ένα ακόμα πεδίο που αφορούσε όλες τις σχετικές εικόνες. Την βάση μας αποτελούσε λοιπόν ένα xml αρχείο της μορφής:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<imageclef>
    <record>
        <figureID>27977</figureID>
        <figureURL>
            http://radiology.rsnajnls.org/cgi/content/full/210/1/28/F1
        </figureURL>
        <caption>
            &lt;B&gt;Figure 1. &lt;/B&gt;
        </caption>
        <title>
            Seymour H. Levitt, MD--President Radiological Society of
North America, 1999
        </title>
        <pmid>9885582</pmid>
        <articleURL>
            http://radiology.rsnajnls.org/cgi/content/full/210/1/28
        </articleURL>
        <imageLocalName>27977.jpg</imageLocalName>
        <link></link>
    </record>
    <record>
        . . .
    </record>
    . . .
</imageclef>
```

Όπου οι διαφορετικές εγγραφές ορίζονται ως record, με χαρακτηριστικά:

- figureID που είναι ο κωδικός της κάθε μιας,
- figureURL την τοποθεσία στην οποία είναι αποθηκευμένη η εικόνα,
- caption την περιγραφή της,
- title τον τίτλο του άρθρου που περιέχει την εικόνα,
- pmid PubMed Identifier, ο μοναδικός κωδικός που αποδίδεται ως παραπομπή στα επιστημονικά άρθρα των βιοεπιστημών και της βιοϊατρικής,
- articleURL την διαδικτυακή τοποθεσία στην οποία βρίσκεται το συγκεκριμένο άρθρο αποθηκευμένο,
- imageLocalName το όνομα της εικόνας τοπικά,
- link πιθανές εικόνες σχετιζόμενες με αυτήν.

Προεπεξεργασία συλλογής (Parsing).

Κατασκευάστηκε ένας parser έτσι ώστε να μπορούν να χρησιμοποιηθούν τα απαραίτητα κάθε φορά δεδομένα από το xml αρχείο. Αυτό επιτεύχθηκε και πάλι με την βοήθεια του πακέτου Digester Component της Apache Software Foundation. Οπότε και δημιουργήθηκε μια τάξη έτσι ώστε οι εγγραφές μας να είναι αντικείμενα αυτής.

Κώδικας τάξης αντικειμένου που αντιπροσωπεύει ένα record του xml.

```
public class Record implements Comparable {
    private String figureID;
    private String figureURL;
    private String caption;
    private String title;
    private String pmid;
    private String articleURL;
    private String imageLocalName;
    private String link;
    private Vector<String> categories;
    public Record() {
        categories = new Vector<String>();
    }
. . .
}
```

Κώδικας μεθόδου που κάνει parse στο xml.

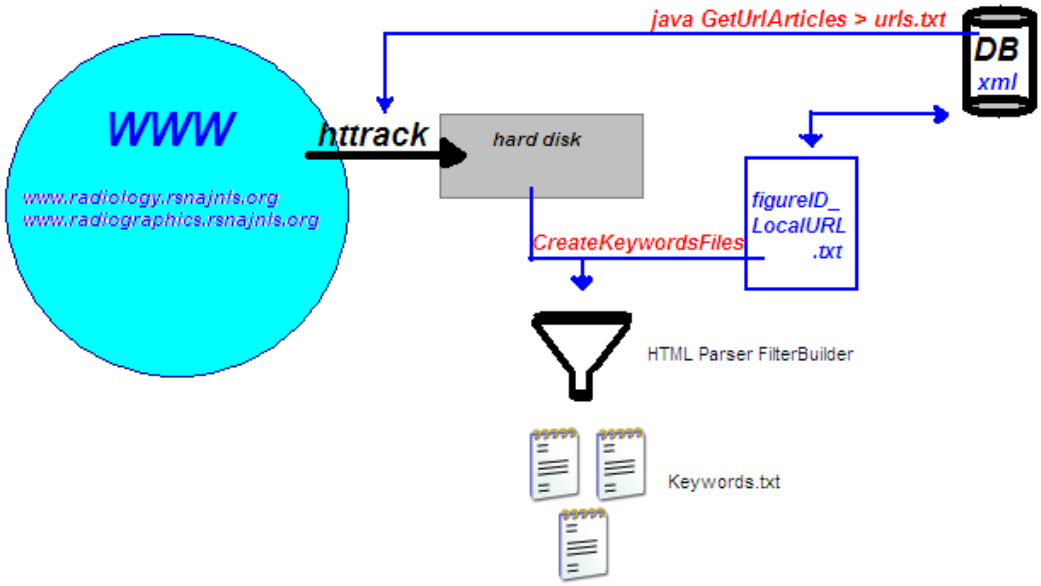
```
public static void main(String[] args) {
    System.err.println("Program starting ... ");
    Digester digester = new Digester();
    digester.setValidating(false);
    digester.addObjectCreate("imageclef", Imageclef.class);
    digester.addObjectCreate("imageclef/record", Record.class);
    digester.addBeanPropertySetter("imageclef/record/figureID",
"figureID");
    digester.addBeanPropertySetter("imageclef/record/figureURL",
"figureURL");
    digester.addBeanPropertySetter("imageclef/record/caption",
"caption");
    digester.addBeanPropertySetter("imageclef/record/title", "title");
    digester.addBeanPropertySetter("imageclef/record/pmid", "pmid");
    digester.addBeanPropertySetter("imageclef/record/articleURL",
"articleURL");
     digester.addBeanPropertySetter("imageclef/record/imageLocalName",
"imageLocalName");
```

```
            digester.addBeanPropertySetter("imageclef/record/link", "link");
            digester.addSetNext("imageclef/record", "addRecord");
            File inputfile = new File("image2009.xml");
            Imageclef imgclef = null;
            try {
               System.err.println("Parsing imageclef starting ... ");
               imgclef = (Imageclef) digester.parse(inputfile);
            } catch (IOException ex) {

Logger.getLogger(ImageclefDigester.class.getName()).log(Level.SEVERE, null,
ex);
            } catch (SAXException ex) {

Logger.getLogger(ImageclefDigester.class.getName()).log(Level.SEVERE, null,
ex);
            }


            if (imgclef != null) {
                System.err.println(inputfile.getAbsolutePath()   +   "   parsed
succesufully");


                addCategories(imgclef);
                createCategoriesXML(imgclef);
            } else {
                System.err.println("ERROR:      CANNOT      PARSE      "      +
inputfile.getAbsolutePath());
            }


        }
```

Προσθήκη λέξεων κλειδιών με βάση τις κατηγορίες των κειμένων.

Με χρήση του παραπάνω parser αντλήθηκαν από το xml αρχείο τα απαραίτητα urls στα οποία βρίσκονταν τα κείμενα των εικόνων. Έπειτα από αυτά αποθηκεύτηκαν οι ιστοσελίδες στις οποίες αναφέρονταν οι εικόνες της βάσης μας. Για να επιτευχθεί αυτό χρησιμοποιήθηκε ένα βοηθητικό πρόγραμμα το οποίο και υλοποίησα προκειμένου να εξαχθούν οι διευθύνσεις εκείνες στις οποίες υπήρχαν τα κείμενα των εικόνων. Έπειτα με χρήση του HTTrack (http://www.httrack.com/) αποθηκεύτηκαν οι σελίδες αυτές σε τοπικό υπολογιστή για περεταίρω επεξεργασία και εξαγωγή των λέξεων κλειδιών που περιείχαν. Μετά την ολοκλήρωση της αποθήκευσης των κατάλληλων στοιχείων υλοποιήθηκε πρόγραμμα το οποίο χρησιμοποιώντας την βάση δεδομένων την οποία είχαμε και τα κείμενα που είχαμε αποθηκεύσει πραγματοποιούσε εξαγωγή των λέξεων κλειδιών. Αναλυτικότερα, αρχικά έγινε μια αντιστοίχηση των στοιχείων/κειμένων που αποθηκεύτηκαν στον υπολογιστή με τα figureID των εικόνων. Έπειτα με χρήση ενός HTML Parser FilterBulder δημιουργήθηκε ένα φίλτρο που ενσωματώθηκε στον κώδικα του προγράμματος που υλοποιήθηκε το οποίο εξήγαγε τις λέξεις κλειδιά από τις αποθηκευμένες ιστοσελίδες και δημιουργούσε αρχεία με όνομα το figureID των εικόνων στις οποίες αναφέρονταν. Μετά την ολοκλήρωση της διαδικασίας αυτής πραγματοποιήθηκε έλεγχος, σχετικά με το αν είχαν αποθηκευτεί όλες οι σελίδες και αν είχαν ανακτηθεί όλες οι λέξεις κλειδιά, εντοπίστηκε ότι έλειπαν κάποια αρχεία που παρουσίασαν ιδιαιτερότητες σε σχέση με την γενική μορφή η οποία είχε ληφθεί υπό όψιν οπότε και υλοποιήθηκε ένα ακόμα πρόγραμμα προκειμένου να έχουμε το απαιτούμενο αποτέλεσμα και σε αυτή την περίπτωση. Στο νέο αυτό πρόγραμμα απλά αλλάχτηκε το φίλτρο εξαγωγής των λέξεων κλειδιών έτσι ώστε να είναι κατάλληλο για την άλλη περίπτωση. Τέλος, οι λέξεις κλειδιά προστέθηκαν σε ένα νέο xml αρχείο το οποίο εκτός από τα πεδία του αρχικού περιέχει και ένα ακόμα categories με χαρακτηριστικά category όπου και αποθηκεύονται ξεχωριστά κάθε μια από τις κατηγορίες στις οποίες ανήκει η συγκεκριμένη εικόνα.

Σχηματική αναπαράσταση της εξαγωγής λέξεων κλειδιών.



Αναζήτηση με χρήση των κατηγοριών.

Indexing.

Για την δημιουργία των ευρετηρίων χρησιμοποιήθηκε το πακέτο lucene, ο parser που περιγράψαμε παραπάνω, αφού πρώτα έγινε μια μικρή τροποποίηση έτσι ώστε να αναγνωρίζει και τις κατηγορίες που έχουν αποθηκευτεί στην νέα βάση μας.

Πειράματα και Συμπεράσματα.

Για το πείραμα αυτό πραγματοποιήθηκε αναζήτηση σε τρία διαφορετικά ευρετήρια, η διαφοροποίηση μεταξύ των ευρετηρίων έχει να κάνει με τα πεδία τα οποία χρησιμοποιούνται στην αναζήτηση. Στο ένα ευρετήριο είχαμε χρήση ενός πεδίου που περιείχε τίτλο και περιγραφή της εικόνας μαζί, στο επόμενο είχαμε τίτλο και περιγραφή της εικόνας σε ένα πεδίο και σε ένα ξεχωριστό την κατηγορία/keyword στην οποία αναφερόταν η εικόνα και που βρέθηκε με τον τρόπο που περιγράφτηκε νωρίτερα. Το τελευταίο ευρετήριο κατασκευαστικέ με τρία διαφορετικά, έτσι ώστε να επιτευχθεί η πριμοδότηση κάποιων από αυτά, τα πεδία μας ήταν τίτλος, περιεχόμενα κειμένων και κατηγορίες, ενώ πριμοδοτήθηκαν αυτά του τίτλου και των κατηγοριών. Μετά την εκτέλεση των αναζητήσεων για διάφορες τιμές πριμοδότησης παρατηρήθηκε ότι οι κατηγορίες δεν βοηθούσαν στην βελτίωση των αναζητήσεων μας. Κατά πάσα πιθανότητα αυτό οφείλεται στο γεγονός ότι οι λέξεις κλειδιά των κειμένων που περιέχουν τις εικόνες δεν είναι απαραίτητα τόσο σχετικές έτσι ώστε να βοηθήσουν στην ανάκτηση εικόνων. Αυτό γιατί η επιλογή τους δεν ήταν τόσο εύστοχη μιας και οι λέξεις αναφέρονταν στα κείμενα και ήταν ιδιαίτερα γενικές σε κάποιες περιπτώσεις σε σχέση με τα περιεχόμενα των εικόνων. Συνεπώς, αντί να επιτύχουμε την άντληση πιο σχετικών κειμένων σε κάποιες περιπτώσεις «αποπροσανατολίζαμε» εντελώς την κατεύθυνση της ζητούμενης αναζήτησης οπότε και χάναμε την χρήσιμη για εμάς πληροφορία.

Αποτελέσματα.

τύπος	*(T+T)*	*(T+T)+C*	*(T+T)^1+C^1.1*	*T+T+C*	*T^2.01+T+C^2.005*
Fields	contents	contents category	contents category	title text category	title text category
boost/field	1/cont	1/cont 1/cat	1/contents 1.1/category	1/title 1/text 1/cat	1.01/title 1/text 1.005/category
Queryid (Num):	25	25	25	25	25
Total number of documents over all queries					
Retrieved:	21234	24429	24446	24381	21926
Relevant:	2362	2362	2362	2362	2362
Rel_ret:	1027	1034	1031	1022	1019
Interpolated Recall - Precision Averages:					
at 0.00	0.8156	0.7287	0.7	0.692	0.6963
at 0.10	0.6653	0.5116	0.4968	0.4994	0.5177
at 0.20	0.5299	0.395	0.389	0.3935	0.4008
at 0.30	0.345	0.2528	0.2477	0.2406	0.2552
at 0.40	0.2344	0.145	0.1414	0.1239	0.1373
at 0.50	0.064	0.0361	0.0351	0.0297	0.0318
at 0.60	0.0245	0.0096	0.0091	0.0089	0.0091
at 0.70	0.0015	0	0	0.0003	0.0003
at 0.80	0.0007	0	0	0	0
at 0.90	0	0	0	0	0
at 1.00	0	0	0	0	0
Average precision (non-interpolated) over all rel docs					
	0.2123	0.1544	0.1498	0.1519	0.1562
Precision:					
At 5 docs:	0.584	0.432	0.424	0.504	0.488
At 10 docs:	0.56	0.396	0.392	0.448	0.456
At 15 docs:	0.5413	0.408	0.3867	0.424	0.4427
At 20 docs:	0.512	0.382	0.37	0.416	0.426
At 30 docs:	0.4533	0.3653	0.344	0.3693	0.3827
At 100 docs:	0.2388	0.2072	0.2052	0.1984	0.206
At 200 docs:	0.1482	0.1372	0.1354	0.1322	0.137
At 500 docs:	0.0755	0.0736	0.073	0.0712	0.071
At 1000 docs:	0.0411	0.0407	0.0407	0.0405	0.0408
R-Precision (precision after R (= num_rel for a query) docs retrieved):					
Exact:	0.2753	0.231	0.2297	0.2229	0.2255

Όπου με Τ συμβολίζεται ο τίτλος (title) και το κείμενο/περιγραφή (text) της εικόνας, και με C οι κατηγορίες (categories) της εικόνας.

Πειράματα με βάση τους όρους του MeSH.

Συλλογή δεδομένων.

Για αυτό το πείραμα χρησιμοποιήθηκε και πάλι η βάση του ImageClef 2009 αλλά αυτή την φορά με μια άλλη μορφή. Σε αυτή την βάση για κάθε pmid είχαν προστεθεί οι όροι του MeSH που αντιστοιχούν σε αυτό. Οπότε η νέα δοσμένη βάση είχε την παρακάτω μορφή.

Μορφή συλλογής δεδομένων.

```
<?xml version="1.0" encoding="UTF-8"?>
<imageclef>
    <record>
        <pmid>9885582</pmid>
        <articleURL>
            http://radiology.rsnajnls.org/cgi/content/full/210/1/28
        </articleURL>
        <figureIDs>
            <figureIDss>27977</figureIDss>
        </figureIDs>
        <contents>seymour h. levitt, md--president radiological
society of north america, 1999 &lt;b&gt;figure 1. &lt;/b&gt;</contents>
        <meshterms>
            <meshterm>history</meshterm>
            <meshterm>20th</meshterm>
            <meshterm>century</meshterm>
            <meshterm>humans</meshterm>
            <meshterm>radiology</meshterm>
            <meshterm>history</meshterm>
        </meshterms>
    </record>
    <record>
        <pmid/>
        <articleURL>
            http://radiology.rsnajnls.org/cgi/content/full/246/2/630
        </articleURL>
        <figureIDs>
            <figureIDss>27978</figureIDss>
            <figureIDss>48504</figureIDss>
        </figureIDs>
        <contents>william w. olmsted, md, new rsna education editor
&lt;b&gt;figure 1. &lt;/b&gt;</contents>
        <meshterms/>
    </record>
    <record>
    . . .
    </record>
. . .
</imageclef>
```

Όπου όλες οι σχετιζόμενες εικόνες βρίσκονται σε ένα record και το χαρακτηριστικό figureIDs περιέχει όλες τις σχετικές. Η κάθε μια με διαφορετικό figureIDss. Ακόμα έχει προστεθεί και το χαρακτηριστικό meshterms που περιέχει όλα τα meshterm της εικόνας. Τα meshterms της κάθε εικόνας, όπως ήδη αναφέραμε, προέκυψαν από το pmid αυτής.

Parsing & Indexing.

Προκειμένου να εκμεταλλευτούμε την διαφορετική μορφή των στοιχείων του xml αρχείου μας τροποποιήθηκε κατάλληλα ο υπάρχον parser. Οπότε χρησιμοποιήθηκε και πάλι το πακέτο του Digester και προχωρήσαμε στην κατασκευή ευρετηρίων με παρόμοιο τρόπο. Δηλαδή ευρετήρια με πεδία, τίτλου και περιεχομένου ενοποιημένα σε ένα πεδίο και ένα ακόμα πεδίο για τους όρους MeSH. Τα ευρετήρια αυτά αξιοποιήθηκαν προκειμένου να εκτελεστούν οι απαραίτητες αναζητήσεις.

Πειράματα.

Απλή αναζήτηση χωρίς την χρήση MeSH.

Απλή αναζήτηση των ερωτήσεων στο ενοποιημένο πεδίο τίτλου και περιεχομένου. Η αναζήτηση έγινε με αντίστοιχο τρόπο όπως ήδη έχουμε περιγράψει παραπάνω.

Αναζήτηση με χρήση MeSH.

Αναζήτηση των ερωτήσεων στο πεδίο του τίτλου και του περιεχομένου και των όρων του MeSH στο πεδίο με τους MeSH όρους. Χρησιμοποιήθηκε και πάλι κώδικας αντίστοιχος με αυτόν που περιγράφτηκε και παραπάνω.

Αναζήτηση με ανάδραση συνάφειας (relevance feedback) στους όρους MeSH.

Αναζήτηση, όπως και πριν, στο πεδίο του τίτλου και του περιεχομένου της ερώτησης μας και των όρων MeSH στο αντίστοιχο πεδίο. Αρχικά έγινε αναζήτηση της ερώτησης μας και από τα αποτελέσματα αυτής έγινε επιλογή ενός αριθμού κειμένων, από τα κείμενα αυτά ανακτήθηκαν οι όροι MeSH και εν συνεχεία αξιολογήθηκαν με βάση την συχνότητα εμφάνισης τους. Οπότε με βάση την συχνότητα επιλέχτηκαν κάποιοι όροι οι οποίοι προστέθηκαν στον αρχικό ερώτημα το οποίο υποβλήθηκε και πάλι στην βάση για να πάρουμε τα τελικά αποτελέσματα. Η λειτουργία αυτή είναι ακριβώς όπως περιγράψαμε και αναλυτικότερα νωρίτερα, απλά τώρα εφαρμόστηκε σε διαφορετική βάση.

Αποτελέσματα.

Αναζήτηση	***Χωρίς MeSH terms, RF***	***Με MeSH terms, χωρίς RF***	***Με MeSH terms και RF***		
Ανακτηθέντα κείμενα για RF			***5***	***10***	***10***
Συχνότητα όρων στο RF			***3***	***5***	***10***
Total number all queries	25	25	25	25	25
Total number of documents retrieved over all queries	22052	22696	25000	25000	24020
Total number of relevant documents over all queries	2362	2362	2362	2362	2362
Total number of relevant documents retrieved over all queries	1017	1049	921	932	1010
Mean Average Precision (MAP)	0.2332	0.2137	0.144	0.146	0.1765
R-Precision (Precision after R (= num-rel for topic) documents retrieved)	0.3016	0.2877	0.2222	0.2271	0.2562
Precision after 5 docs retrieved	0.696	0.576	0.44	0.456	0.512
Precision after 10 docs retrieved	0.62	0.544	0.376	0.396	0.484
Precision after 15 docs retrieved	0.5973	0.528	0.3413	0.3707	0.456
Precision after 20 docs retrieved	0.564	0.512	0.334	0.348	0.428

Συμπέρασμα.

Παρατηρούμε ότι η απλή αναζήτηση στην βάση του imageClef 2009 αν και δεν πετυχαίνει να ανακτήσει τον αριθμό των κειμένων που επιτυγχάνετε με τις επόμενες μεθόδους καταφέρνει να έχει καλύτερα αποτελέσματα. Αυτό σημαίνει ότι η χρήση των MeSH όρων δεν είναι ιδιαίτερα αποδοτική για την περίπτωση μας μιας και όπως φαίνεται από τα παραπάνω πειράματα αντί να επιτυγχάνεται βελτίωση των αποτελεσμάτων με την χρήση τους αντίθετα αυτά γίνονται χειρότερα. Επίσης όσο προσπαθούμε να τα χρησιμοποιήσουμε περισσότερο στις αναζητήσεις μας τόσο αυτά παρουσιάζουν μείωση των επιθυμητών αποτελεσμάτων. Προσπαθώντας να εντοπιστούν τα βαθύτερα αίτια αυτής της μη αναμενόμενης συμπεριφοράς, παρατηρήθηκε ότι υπήρχαν ερωτήσεις οι οποίες είχαν την τάση να χαλάνε τα αποτελέσματα. Οπότε θα μπορούσαμε να καταλήξουμε στο συμπέρασμα ότι δεν αρκεί ένας μηχανισμός που μπορεί να δουλέψει κάτω από ορισμένες συνθήκες.

Παράρτημα

Βοηθητικά εργαλεία που χρησιμοποιήθηκαν.

The Digester Component.

Για την παροχή προετοιμασίας των διαφόρων αντικειμένων της Java στα πλαίσια του συστήματος πολλά προγράμματα διαβάσουν αρχεία ρυθμίσεων σε μορφή XML. Υπάρχουν διάφοροι τρόποι να γίνει αυτό, και το στοιχείο του *Digester* έχει σχεδιαστεί για να παρέχει μια κοινή εφαρμογή που μπορεί να χρησιμοποιηθεί σε πολλά διαφορετικά σχέδια.

Βασικά, το πακέτο *Digester* επιτρέπει να ρυθμιστεί ένα XML αρχείο σε ένα αντικείμενο χαρτογράφησης της Java, το οποίο πυροδοτεί ορισμένες δράσεις, που ονομάζονται κανόνες, κάθε φορά που ένα συγκεκριμένο πρότυπο XML αναγνωρίζεται. Ένα πλούσιο σύνολο προκαθορισμένων κανόνων είναι στη διάθεσή μας, ή μπορούμε επίσης να δημιουργήσετε το δικό μας. Οι πιο εξειδικευμένες λειτουργίες του *Digester* περιλαμβάνουν:

* Δυνατότητα δημιουργίας ενός διαφορετικού προτύπου αν το προεπιλεγμένο δεν ταιριάζει στις ανάγκες μας.

* Προαιρετικό namespace, ώστε να μπορούμε να καθορίσουμε κανόνες που αφορούν μόνο σε ένα συγκεκριμένο χώρο ονομάτων XML.

* Ενσωμάτωση των κανόνων μας σε σύνολα κανόνων που μπορούν εύκολα και άνετα να επαναχρησιμοποιηθούν σε περισσότερες από μία εφαρμογές που απαιτούν το ίδιο είδος επεξεργασίας.

Apache Lucene.

Το Apache Lucene είναι μια πανίσχυρη βιβλιοθήκη αναζήτησης υψηλής απόδοσης γραμμένη σε Java, η οποία μπορεί να προστεθεί εύκολα σε οποιαδήποτε εφαρμογή. Το γεγονός ότι είναι γραμμένη σε Java την καθιστά ως μια τεχνολογία κατάλληλη για σχεδόν οποιαδήποτε εφαρμογή απαιτεί αναζήτηση πλήρους κειμένου, ειδικά αν αυτή πρόκειται να τρέξει σε πολλές διαφορετικές πλατφόρμες. Κατά τα τελευταία έτη το Lucene έχει γίνει εξαιρετικά δημοφιλής και αποτελεί σήμερα την πιο ευρέως χρησιμοποιούμενη βιβλιοθήκη ανάκτησης πληροφοριών. Έτσι κυριαρχεί πίσω από τις επιλογές αναζήτησης σε πολλές τοποθεσίες Web και εργαλεία υπολογιστών γραφείου. Παρά το γεγονός ότι είναι γραμμένο σε Java, χάρη στη φήμη του και στον ζήλο με το οποίο δουλεύουν οι σχεδιαστές του για την ανάπτυξη του, αυτή την στιγμή υπάρχουν πλέον αρκετά ports και αναβαθμίσεις ένταξης σε άλλες γλώσσες προγραμματισμού (C/C++, C#, Ruby, Perl, Python, PHP, κτλ.). Ένας από τους βασικούς παράγοντες πίσω από δημοτικότητα Lucene είναι η απλότητά του, αλλά παρόλα αυτά από πίσω κρύβεται ένα πανέξυπνο εργαλείο με μεγάλη πολύπλοκα, που εκμεταλλεύεται την κατάσταση των τεχνικών της Ανάκτησης Πληροφοριών ήσυχα και αθόρυβα χωρίς να είναι εμφανές στον χρήστη. Η προσεκτική θέση της ευρετηρίασης και αναζήτησης του API είναι ένα ακόμα σημάδι του πόσο καλά σχεδιασμένο είναι το λογισμικό. Κατά συνέπεια, δεν χρειάζεται εις βάθος γνώσεις σχετικά με τον τρόπο λειτουργίας της ευρετηρίασης και ανάκτησης των πληροφοριών στον μηχανισμό του Lucene, ώστε χρησιμοποιηθεί το εργαλείο αυτό. Επιπλέον, το Lucene API προϋποθέτει μόνο

μια σειρά από τάξεις προκειμένου προχωρήσουμε στην χρήση του. Έτσι το Lucene μπορεί να χρησιμοποιηθεί για μια τυπική εφαρμογή αναζήτησης. Παρόλα αυτά θα ήταν τραγικό να αντιμετωπίσουμε το Lucene σαν να είναι απλώς μια κοινή βιβλιοθήκη αναζήτησης, και θα πρέπει να χειριστούμε τα άλλα συστατικά μιας εφαρμογής αναζήτησης (crawling, φιλτράρισμα κειμένων, server runtime, διεπαφή χρήστη, διαχείριση κλπ.) όπως απαιτεί η εφαρμογή μας. Στην εφαρμογή μας χρησιμοποιήθηκε προκειμένου να κατασκευαστούν τα ευρετήρια και εν συνεχεία να γίνουν όλες οι απαραίτητες αναζητήσεις σε αυτά.

Text REtrieval Conference, TREC.

Το Text REtrieval Conference (TREC) είναι συγχρηματοδοτούμενο από το National Institute of Standards and Technology (NIST) και από το Υπουργείο Άμυνας των ΗΠΑ, ξεκίνησε το 1992 ως μέρος του προγράμματος κειμένου TIPSTER. Σκοπός του ήταν να στηρίξει την έρευνα στους κόλπους της κοινότητας της ανάκτηση πληροφοριών, παρέχοντας την απαραίτητη υποδομή για μεγάλης κλίμακας αξιολόγησης μεθόδων ανάκτησης κειμένου. Ειδικότερα, το εργαστήριο του TREC έχει τους εξής στόχους:

* Να ενθαρρύνει την έρευνα σε πληροφορίες ανάκτησης που βασίζονται σε μεγάλες συλλογές δοκιμών,

* Να αυξηθεί η επικοινωνία μεταξύ της βιομηχανίας, του ακαδημαϊκού κόσμου, και της κυβέρνησης με τη δημιουργία ενός ανοικτού φόρουμ για την ανταλλαγή ερευνητικών ιδεών,

* Να επιταχύνουν τη μεταφορά της τεχνολογίας από τα ερευνητικά εργαστήρια σε εμπορικά προϊόντα με την επίδειξη σημαντικών βελτιώσεων στις μεθόδους ανάκτησης για πραγματικά προβλήματα, και

* Να αυξηθεί η διαθεσιμότητα των απαραίτητων τεχνικών αξιολόγησης οι οποίες χρησιμοποιούνται είτε από τη βιομηχανία είτε από την ακαδημαϊκή κοινότητα, συμπεριλαμβανομένης της ανάπτυξης νέων τεχνικών αξιολόγησης που έχουν μεγαλύτερη εφαρμογή στις μέρες μας.

Το TREC επιβλέπεται από μια επιτροπή προγράμματος αποτελούμενη από εκπροσώπους της κυβέρνησης, της βιομηχανίας και ακαδημαϊκού χώρου. Για κάθε TREC, το NIST παρέχει ένα δείγμα εγγράφων και των αντίστοιχων ερωτήσεων. Οι συμμετέχοντες εκτελούν πειράματα με τα δικά τους συστήματα ανάκτησης δεδομένων, και επιστρέφουν στο NIST κατάλογο των καλύτερα αξιολογημένων εγγράφων. Το NIST έπειτα εξάγει τα επιμέρους αποτελέσματα, κρίνει τα ανακτημένα έγγραφα ως προς την ορθότητα, και αξιολογεί τα αποτελέσματα. Ο κύκλος του TREC τελειώνει με τους συμμετέχοντες να μοιράζουν τις εμπειρίες τους μέσα από ένα φόρουμ.

Σε αυτή την προσπάθεια αξιολόγησης έχει αυξηθεί τόσο ο αριθμός των συμμετεχόντων συστημάτων όσο και ο αριθμός των καθηκόντων κάθε χρόνο. Ενενήντα τρεις ομάδες που εκπροσωπούν 22 χώρες συμμετείχαν στο TREC 2003. Οι συλλογές δοκιμών του TREC και το λογισμικό αξιολόγησης είναι σε μεγάλο βαθμό στη διάθεση της ερευνητικής κοινότητας ανάκτησης, έτσι ώστε οι οργανισμοί να μπορούν να αξιολογούν τα δικά τους συστήματα ανάκτησης οποιαδήποτε στιγμή. Έτσι το TREC έχει ανταποκριθεί επιτυχώς τους στόχους του

για τη βελτίωση της τρέχουσας κατάστασης στην ανάκτηση πληροφοριών και τη διάδοση της τεχνολογίας. Η αποτελεσματικότητας των συστημάτων ανάκτησης σχεδόν διπλασιάστηκε κατά τα πρώτα έξι χρόνια του TREC.

Το TREC έχει υποστηρίξει επίσης την πρώτη μεγάλης κλίμακας αξιολόγηση στην ανάκτηση μη αγγλικών (Ισπανικά και Κινέζικα) εγγράφων, στην ανάκτηση ηχογραφήσεων λόγου, και στην ανάκτησης ανάμεσα σε πολλές γλώσσες. Ακόμα το TREC εισήγαγε αξιολογήσεις για να απαντηθεί το ερώτημα ανοιχτού πεδίου με βάση το περιεχόμενο ανάκτησης ψηφιακού βίντεο. Οι συλλογές δοκιμών του TREC είναι αρκετά μεγάλες έτσι ώστε να μπορούν να παρέχουν ρεαλιστικά μοντέλα. Οι περισσότερες σημερινές μηχανές αναζήτησης που είναι διαθέσιμες στο ευρύ κοινό περιλαμβάνουν τεχνολογία που αρχικά αναπτύχθηκε το TREC. Για τον λόγο αυτό το TREC ήταν ένα από τα εργαλεία που επιλέχθηκαν προκειμένου να αξιολογηθούν τα αποτελέσματα των αναζητήσεων μας.

HTTrack Website Copier.

HTTrack είναι ένα δωρεάν (GPL, δωρεάν / ελεύθερο λογισμικό) βοηθητικό και εύκολο στη χρήση προγράμμα που παρέχει εκτός σύνδεσης περιήγηση. Επιτρέπει στον χρήστη να κατεβάσει ένα World Wide Web site από το Internet σε έναν τοπικό κατάλογο, αναδρομικά με βάση όλους τους καταλόγους, παίρνει HTML, εικόνες και άλλα αρχεία από το διακομιστή στον τοπικό υπολογιστή του χρήστη. Το HTTrack οργανώνει τον αρχικό σύνδεσμο τον στον αντίστοιχο σχετικό σύνδεσμο στην αρχική τοποθεσία της διάρθρωσης όπου το όρισε ο χρήστης. Απλά χρειάζεται ο χρήστης να ανοίξει μια σελίδα από τον ιστότοπο που αποθήκευσε στον browser, και έπειτα μπορεί να περιηγηθεί στο site από σύνδεσμο σε σύνδεση, αντίστοιχο αυτού που υπάρχει online. Το HTTrack μπορεί επίσης να ενημερώσει μια υπάρχουσα ιστοσελίδα ειδώλου, και να συνεχίσει λήψεις που έχουν διακοπεί. Ακόμα, το HTTrack είναι πλήρως παραμετροποιήσιμο, και έχει ένα ολοκληρωμένο σύστημα βοήθειας, έτσι ώστε να μπορεί να ρυθμιστεί από τον εκάστοτε χρήστη. Μπορεί να χρησιμοποιηθεί είτε σε έκδοση για Windows είτε σε κάποια έκδοση για Linux.

Στα πλαίσια της εργασίας χρησιμοποιήθηκε προκειμένου να αποθηκευτούν κάποια από τα κείμενα που ήταν απαραίτητα για την εργασία έτσι ώστε να αναλυθούν με χρήση κάποιων αναλυτών κειμένου (parsers) και εξαχθούν από αυτά οι απαραίτητες πληροφορίες. Η χρήση του προγράμματος έγινε με τέτοιο τρόπο ώστε να μην επιβαρυνθούν οι εξυπηρετητές που της φιλοξενούσαν. Οι αποθήκευση των κειμένων αυτών έγινε χωρίς την αναζήτηση σε επόμενους συνδέσμους, απλά γινόταν αποθήκευση του κειμένου της πρώτης σελίδας.

Βιβλιογραφία.

Hersh WR, Hickam DH, Use of a multi-application computer workstation in a clinical setting, Bulletin of the Medical Library Association, 1994, 82: 382-389.

Hersh WR, Buckley C, Leone TJ, Hickam DH, OHSUMED: An interactive retrieval evaluation and new large test collection for research, Proceedings of the 17th Annual ACM SIGIR Conference, 1994, 192-201.

Gospodnetic, Otis; Erik Hatcher,Michael McCandless (June 28, 2009). *Lucene in Action* (2nd edition), Manning Publications, ISBN 1933988177.

MetaMap: http://metamap.nlm.nih.gov/

Trec: http://trec.nist.gov/

HTTrack: http://www.httrack.com/

Digester Component: http://commons.apache.org/digester/

Apache Lucene: http://lucene.apache.org/

ImageClef 2009: http://www.imageclef.org/2009/medical

PubMed: http://www.ncbi.nlm.nih.gov/pubmed/

---

[i] W. Hersh, C. Buckley, T. J. Leone, and D. Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In W. B. Croft and C. J. V. Rijsbergen, editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–201, Dublin, Ireland, July 1994. ACM, Springer-Verlag

[ii] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In H.-P. Frei, D. Harman, P. Schnabel, and R. Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 298–306, Zürich, CH, 1996. ACM Press, New York, US.

[iii] συντομογραφίες *Derived from the NIH UMLS* (*Unified Medical Language System. http://www.nlm.nih.gov/research/umls/*) όπως φαίνονται και στον πίνακα http://www.fpnotebook.com/ENT/Sx/Otrh.htm