



Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών  
HY463 – Συστήματα Ανάκτησης Πληροφοριών  
2018-2019 Ανοιξιάτικο Εξάμηνο  
Βοηθοί: Συντονιστής ο Γιώργος Καντηλιεράκης (συνδρομή όλοι οι βοηθοί)

### Εργασία Μαθήματος (project)

Άτομα ανά ομάδα: 1 ή 2

Αξία: 25% του τελικού σας βαθμού

Φάση Α: 15 Μαρτίου – 22 Απριλίου (60%)

Φάση Β: 22 Απριλίου – 14 Μαΐου (40%)

## Υλοποίηση ενός Συστήματος Ανάκτησης Πληροφοριών και Δοκιμή και Αξιολόγησή του σε Συλλογή Βιοϊατρικών Άρθρων

Σκοπός αυτής της εργασίας είναι να κατανοήσετε κάποιες βασικές έννοιες και τεχνικές, φτιάχνοντας εξ' αρχής τον λειτουργικό πυρήνα ενός δικού σας Συστήματος Ανάκτησης Πληροφοριών (ΣΑΠ) - στα αγγλικά Information Retrieval System. Συνάμα καλείστε να δοκιμάσετε το σύστημα που θα φτιάξετε σε μια πραγματική **συλλογή βιοϊατρικών άρθρων**. Στην δεύτερη φάση της εργασίας θα πρέπει να αξιολογήσετε την αποτελεσματικότητα του συστήματός σας, ακολουθώντας την επιστημονικά καθιερωμένη μεθοδολογία.

### Σχόλια

**Σ1.** Η εργασία «σπάει» εύκολα σε 2 άτομα. Αυτό πέραν του ότι μειώνει τον απαιτούμενο φόρτο, επιτρέπει την παράλληλη εργασία των δύο ατόμων. Συγκεκριμένα το 1<sup>ο</sup> μέλος της ομάδας μπορεί να ασχοληθεί κυρίως (όχι αποκλειστικά) με τη Φάση Α και το 2<sup>ο</sup> κυρίως (όχι αποκλειστικά) με τη Φάση Β. Λάβετε υπόψη ότι η Φάση Α απαιτεί περισσότερη δουλειά από τη Β. Η έναρξη της Φάσης Β δεν προϋποθέτει ολοκλήρωση της Φάσης Α. Μάλλον θα βοηθήσει αν τα δύο μέλη διαβάσουν προσεκτικά ολόκληρη την εκφώνηση και συμφωνήσουν εξ αρχής στο interface μέσω του οποίου ο κώδικας του συστήματος τη Φάσης Β θα στέλνει ερωτήματα και θα λαμβάνει απαντήσεις από το σύστημα της Φάσης Α.

**Σ2.** Καλείστε να δηλώσετε τις ομάδες έως τη Παρασκευή 22 Μαρτίου (στο κατάλληλο σημείο στο elearn). Κατά τη διάρκεια της προφορικής εξέτασης θα ζητηθεί να γίνουν αλλαγές στο σύστημα από αμφότερα τα μέλη της κάθε ομάδας τα οποία θα πρέπει να είναι σε θέση να τις κάνουν. Για το σκοπό αυτό η αναφορά της κάθε φάσης πρέπει να αναφέρει συγκεκριμένα τι έκανε το κάθε μέλος μιας ομάδας.

### ΦΑΣΗ Α (60 μονάδες)

Για να φτιάξετε το ζητούμενο σύστημα ανάκτησης βιοϊατρικών άρθρων μπορείτε να ακολουθήσετε τα εξής βήματα:

#### Διαδικασία Ευρετηρίασης (B1-B6)

**B1)** (2) Γράψτε ένα πρόγραμμα σε Java το οποίο να διαβάζει τα περιεχόμενα των ετικετών ενός XML αρχείου (που στην ουσία αντιπροσωπεύει ένα βιοϊατρικό άρθρο) σε UTF-8 κωδικοποίηση (ώστε να υπάρχει υποστήριξη πολυγλωσσικότητας) και να τυπώνει το πλήθος των διαφορετικών λέξεων, και την κάθε διαφορετική λέξη συνοδευόμενη από τις ετικέτες στις οποίες εντοπίστηκε και το πλήθος εμφανίσεών της στην κάθε ετικέτα<sup>1</sup> (πληροφορίες για τη συλλογή, τα XML αρχεία και τις ετικέτες που μας ενδιαφέρουν θα βρείτε στο **Παράρτημα Α**). Το πρόγραμμα σας θα πρέπει να αγνοεί τις λέξεις αποκλεισμού (περιλαμβάνονται στα αρχεία **stopwordsEn.txt** και **stopwordsGr.txt**, για αγγλικά και ελληνικά αντίστοιχα) και τα σημεία στίξης που πιθανόν να εμφανίζονται. Βοηθητικός κώδικας σε Java για ανάγνωση των ετικετών που μας

<sup>1</sup> Μπορείτε να δείτε (ή να θυμηθείτε) τα παραδείγματα που υπάρχουν στο <http://www.csd.uoc.gr/~hy252/Lectures07/pdf/CS252CollectionClassesInterfaces07.pdf>

ενδιαφέρουν από ένα βιοϊατρικό άρθρο, καθώς και για τον διαχωρισμό ενός αλφαριθμητικού σε λέξεις, υπάρχει στο **Παράρτημα Β**. Δοκιμάστε την υλοποίησή σας σε ένα αρχείο της συλλογή **“Medical Collection”**.

**B2) (3)** Επεκτείνετε το σύστημα ώστε να μπορεί να διαβάσει όχι μόνο ένα, αλλά πολλά αρχεία (π.χ. όσα βρίσκονται σε ένα συγκεκριμένο φάκελο του λειτουργικού συστήματος). Το σύστημα πρέπει τώρα για κάθε διαφορετική λέξη να τυπώνει τα αρχεία στα οποία εμφανίζεται καθώς και (όπως στο B1) τη συχνότητα εμφάνισής της σε κάθε ετικέτα του εκάστοτε αρχείου. Για πρόσβαση σε όλα τα αρχεία ενός φακέλου (συμπεριλαμβανομένων των αρχείων σε υποφακέλους αναδρομικά) δείτε το **Παράρτημα Γ**. Δοκιμάστε την υλοποίησή σας στη συλλογή **“Medical Collection”**.

**B3) (3)** Στη συνέχεια θα πρέπει να υποστηρίξετε τη διαδικασία του *stemming* των λέξεων που διαβάζετε στα B1 και B2, δηλαδή να βρείτε τις ρίζες των λέξεων (π.χ. ‘ending’ -> ‘end’). Για να το επιτύχετε αυτό θα πρέπει να χρησιμοποιήσετε το **Stemmer.jar**, μια βιβλιοθήκη που προσφέρει stemming σε ελληνικές και αγγλικές λέξεις και φτιάχτηκε στα πλαίσια εργασιών παλαιότερων ετών για τη μηχανή *mitos*. Δείτε το **Παράρτημα Δ** για το πώς θα τη χρησιμοποιήσετε. Ενδεικτικά μπορείτε να τρέξετε και τη γραφική διεπαφή του Stemmer με `“java -jar Stemmer.jar”`.

**B4) (5)** Επεκτείνετε το σύστημα ώστε να δημιουργεί ένα φάκελο **CollectionIndex** στον οποίο να δημιουργεί ένα αρχείο με όνομα **VocabularyFile.txt** που θα καταγράφει όλες τις διαφορετικές λέξεις σε αύξουσα (λεξικογραφική) σειρά. Δίπλα σε κάθε λέξη να καταγράφεται το πλήθος των εγγράφων στα οποία εμφανίζεται (document frequency, df).

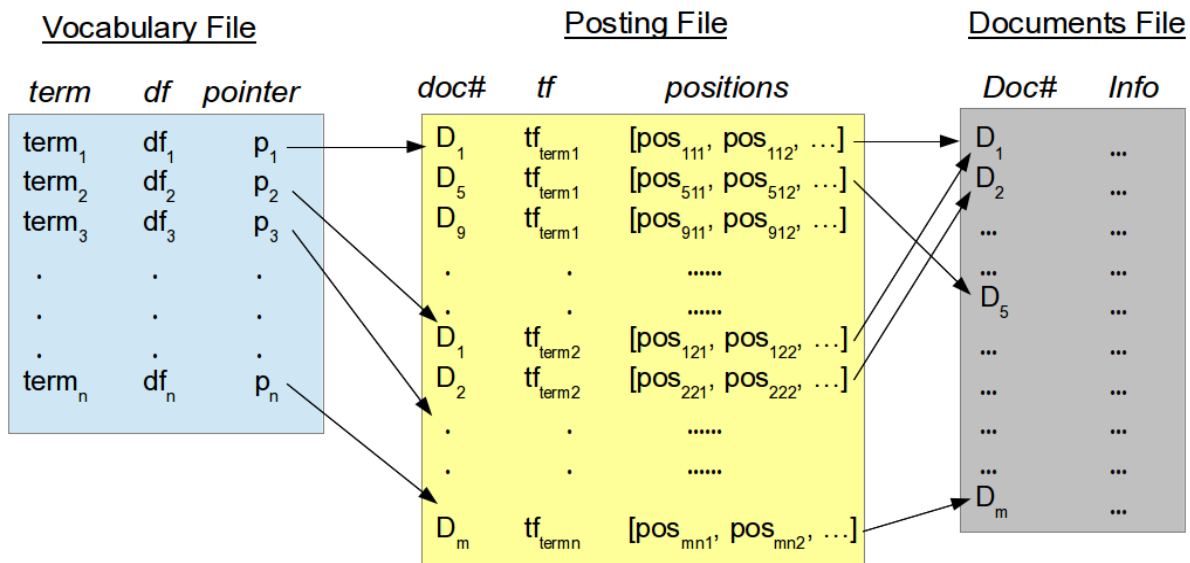
**B5) (7)** Επεκτείνετε το σύστημα ώστε να δημιουργεί ένα ακόμη αρχείο με όνομα **DocumentsFile.txt** στον φάκελο **CollectionIndex** που θα καταγράφει για κάθε αρχείο της συλλογής μία τριάδα αποτελούμενη από ένα μοναδικό αριθμητικό αναγνωριστικό, το πλήρες μονοπάτι του αρχείου, και το μήκος του διανύσματος (νόρμα) του εγγράφου όπως έχουμε αναφέρει στις αντίστοιχες διαλέξεις (ώστε να επιταχυνθεί αργότερα η διαδικασία της αποτίμησης επερωτήσεων). Οι εγγραφές αυτές να είναι καταγεγραμμένες σε αύξουσα σειρά ως προς το αναγνωριστικό του εγγράφου.

**B6) (10)** Επεκτείνετε το σύστημα ώστε να μπορεί να κρατά πληροφορία σχετική με τα έγγραφα στα οποία εμφανίζεται η κάθε λέξη. Μπορείτε να ακολουθήσετε μία από τις δύο επιλογές που περιγράφονται παρακάτω:

1. Δημιουργείτε ένα αρχείο για κάθε λέξη (που εμφανίζεται στο **VocabularyFile.txt**) και για κάθε έγγραφο στο οποίο εμφανίζεται η λέξη να έχετε μία εγγραφή που θα περιέχει:
  - a. το **αναγνωριστικό** του εγγράφου στο οποίο εμφανίζεται η λέξη,
  - b. το **tf** της λέξης στο αντίστοιχο έγγραφο,
  - c. τις **θέσεις εμφάνισης** της λέξης στο έγγραφο,
  - d. ένα **δείκτη** προς τις αντίστοιχες πληροφορίες του συγκεκριμένου εγγράφου στο **DocumentsFile.txt**.

Τα αρχεία αυτά θα δημιουργούνται στο φάκελο **CollectionIndex/InvertedLists**.

2. Δημιουργείτε ένα μόνο αρχείο (ας το πούμε **PostingFile.txt**) που θα περιέχει τις παραπάνω πληροφορίες για όλες τις λέξεις. Στην περίπτωση αυτή, θα χρειαστεί να καταγράφετε για κάθε λέξη  $t_i$  στο αρχείο **VocabularyFile.txt** άλλον έναν αριθμό ο οποίος θα περιγράφει τη θέση στο αρχείο **PostingFile.txt** από την οποία αρχίζουν τα στοιχεία που αφορούν τη λέξη  $t_i$  (πεδίο **pointer** στο Vocabulary File στην Εικόνα 1). Σε αυτήν την περίπτωση μπορεί κάποιος για ευκολία να θεωρήσει ότι τα postings όλων των όρων μπορούν να κρατηθούν στην μνήμη πριν αυτά γραφούν στο Posting File στο τέλος της ευρετηρίασης. Όμως αυτό περιορίζει τις δυνατότητες του συστήματος σας και την απόδοσή του για μεγάλες συλλογές όπως αυτή που σας δίνεται. **Συστήνεται να δοκιμάσετε τις τεχνικές partial indexing and merging** όπως είδαμε στο μάθημα. Αν κάποιος τις λάβει υπόψη του από την αρχή, το σύστημα του θα μπορεί να εφαρμοστεί και σε πολύ μεγάλες συλλογές εγγράφων, και δεν θα συναντήσει καμία δυσκολία για την ευρετηρίαση της συλλογής εγγράφων που σας δίνεται, διαφορετικά η διαδικασία ευρετηρίασης σε συστήματα με λίγη μνήμη θα είναι πάρα πολύ αργή.



Εικόνα 1. Το ανεστραμμένο ευρετήριο

Συνίσταται η επιλογή (2), η οποία απαιτεί τη δημιουργία ενός μοναδικού PostingFile καθώς και η χρήση *random access files*<sup>2</sup> με σκοπό την αποφυγή σειριακής σάρωσης του αρχείου (δείτε Παράρτημα Ε) .

Να σημειωθεί ότι οι παραπάνω πληροφορίες είναι οι **ελάχιστες** που πρέπει να αποθηκεύσουμε για τη δημιουργία του InvertedFile. Είστε ελεύθεροι να κάνετε όποιες προσθήκες κρίνετε χρήσιμες/αναγκαίες τόσο για την **επιτάχυνση** του συστήματος σας, όσο και για την **καλύτερη απόδοση** του στο συγκεκριμένο πρόβλημα ανάκτησης βιοιατρικών άρθρων. Για παράδειγμα, μπορείτε να δώσετε μεγαλύτερη βαρύτητα στις λέξεις του τίτλου ή της σύνοψης. Αυτό συνεπάγεται ότι μπορείτε να **επινοήσετε έναν διαφορετικό τρόπο υπολογισμού του TF** ο οποίος να λαμβάνει υπόψη τις ετικέτες. Θα μπορούσατε επίσης στο Posting File να αποθηκεύετε για κάθε όρο όχι μόνο τις θέσεις του **αλλά και την αντίστοιχη ετικέτα στην οποία βρέθηκε** για την περίπτωση που θέλετε να μπορείτε να αλλάζετε δυναμικά (στη φάση της αξιολόγησης) τον τρόπο υπολογισμού του TF χωρίς να επανασαρώσετε τα έγγραφα. Βέβαια μια τέτοια επιλογή απαιτεί περισσότερο αποθηκευτικό χώρο.

Το σύστημα σας πρέπει να τρέχει από command line (π.χ. "java -jar indexer.jar") και η γραφική διεπαφή είναι προαιρετική, αλλά θα ληφθεί υπόψη στην τελική βαθμολόγηση της εργασίας. Ο χρόνος εκτέλεσης για τη διαδικασία της ευρετηρίασης της συλλογής κειμένων που θα σας δοθεί πρέπει να είναι ο ταχύτερος δυνατός (θα ληφθεί και αυτό υπόψη στη βαθμολόγηση).

### Αποτίμηση επερωτήσεων (B7- B10)

**B7) (5)** Γράψτε ένα νέο πρόγραμμα σε Java το οποίο θα χρησιμοποιεί το ανεστραμμένο ευρετήριο που κατασκευάσατε για την αποτίμηση επερωτήσεων. Με αυτόν τον τρόπο θα επιβεβαιώσετε ότι έχετε κατασκευάσει σωστά το ευρετήριο σας. Προς το παρόν δεν ενδιαφερόμαστε για την κατάταξη των εγγράφων (βάσει κάποιου συγκεκριμένου μοντέλου ανάκτησης), απλώς θέλουμε να ανακτούμε τα έγγραφα στα οποία εμφανίζεται **μία λέξη** που δίνει ο χρήστης ως επερώτηση.

Πριν αρχίσετε την αποτίμηση επερωτήσεων, βεβαιωθείτε ότι το σύστημα ευρετηριασμού (που έχετε φτιάξει στα **B1-B6**) έχει δημιουργήσει τα απαραίτητα αρχεία στο φάκελο CollectionIndex. Το σύστημα αποτίμησης επερωτήσεων πρέπει αρχικά να **φορτώνει το λεξιλόγιο** (Vocabulary File) **στη μνήμη**. Τα postings όμως κάθε όρου (τα αρχεία στον φάκελο Collection/InvertedIndex ή εάν έχετε μόνο ένα αρχείο το Posting File) καθώς και πληροφορίες σχετικά με τα έγγραφα της

<sup>2</sup> <http://www.csd.uoc.gr/~hy252/html/Lectures2012/assist2012/CS252FileStreams12.pdf>  
<http://docs.oracle.com/javase/6/docs/api/java/io/RandomAccessFile.html>  
<http://www.dailymfrecode.com/code/reads-writes-random-access-file-1204.aspx>

συλλογής (Documents File) θεωρούμε ότι έχουν μεγάλο μέγεθος και **δεν πρέπει να κρατηθούν στην μνήμη**, γι' αυτό και παραμένουν στα αντίστοιχα αρχεία στον δίσκο (άρα δεν πρέπει να φορτωθούν στην κύρια μνήμη).

**B8) (10)** Επεκτείνετε το σύστημα που φτιάξατε έτσι ώστε να διαβάζει μία ή περισσότερες λέξεις από την κονσόλα (δηλαδή μια επερώτηση σε φυσική γλώσσα) και να τυπώνει την απάντηση ως προς το **διανυσματικό μοντέλο** (vector space model)<sup>3</sup>. Βασικό κομμάτι του συγκεκριμένου βήματος είναι και ο Επεξεργαστής Επερωτήσεων, ο οποίος θα αφαιρεί τα *stopwords*, τα σημεία στίξης και θα κάνει το *stemming* της επερώτησης (γενικά ό,τι έχετε κάνει και στη δημιουργία του ευρετηρίου).

**B9) (15)** Σας δίνεται μια συλλογή από 30 **ιατρικά θέματα** (αρχείο **topics.xml**). Κάθε ιατρικό θέμα αναπαριστά τα πρακτικά μιας επίσκεψης ασθενή σε γιατρό και περιέχει πληροφορίες όπως το ιστορικό του ασθενή, τα συμπτώματά του, τυχόν εξετάσεις που έκανε, κτλ. Τα θέματα έχουν επισημανθεί σύμφωνα με τους 3 πιο συνηθισμένους **τύπους** κλινικών ερωτήσεων:

- **Διάγνωση** – Ποια είναι η διάγνωση του ασθενή;
- **Εξέταση** – Τι εξετάσεις πρέπει να κάνει ο ασθενής;
- **Θεραπεία** – Ποια αγωγή πρέπει να ακολουθήσει ο ασθενής;

Παραδείγματος χάριν, για ένα θέμα «Διάγνωση», το σύστημά σας πρέπει να ανακτήσει άρθρα που θα φανούν χρήσιμα στον γιατρό για να διαγνώσει την αρρώστια του ασθενή. Αντίστοιχα, για ένα θέμα τύπου «Εξέταση», το σύστημά σας πρέπει να ανακτήσει σχετικά άρθρα που προτείνουν εξετάσεις για την διάγνωση του ασθενή. Τέλος, για ένα θέμα τύπου «Θεραπεία», το σύστημά σας πρέπει να ανακτήσει άρθρα που προτείνουν στον γιατρό τη καλύτερη θεραπεία που πρέπει να ακολουθήσει ο ασθενής.

Για κάθε θέμα, δίνεται μια **περιγραφή** και μια (μικρότερη) **σύνοψη**. Για την δημιουργία της αντίστοιχης επερώτησης μπορείτε να χρησιμοποιήσετε είτε τη περιγραφή, είτε τη σύνοψη, όμως **όχι** και τα 2 μαζί ταυτόχρονα (αυτός ήταν κανόνας του διαγωνισμού TREC<sup>4</sup> το 2015). Επίσης, μπορείτε να **επεξεργαστείτε** τα θέματα όπως εσείς νομίζετε (ώστε να αποδίδει καλύτερα το σύστημά σας), **είτε με αυτόματο τρόπο είτε με παρέμβαση χρήστη**. Σε κάθε περίπτωση **πρέπει να αναφέρετε** αν ο κάθε τρόπος που προτείνετε χρειάζεται παρέμβαση χρήστη ή είναι πλήρως αυτόματος.

Σας ζητείται να **επεκτείνεται το πρόγραμμά σας** (το διανυσματικό μοντέλο, τον επεξεργαστή επερωτήσεων, κτλ.) ώστε να ζητάει από τον χρήστη να δώσει τα στοιχεία ενός ιατρικού θέματος (τον **τύπο** του και την **περιγραφή ή τη σύνοψη** του) και να ανακτά **σχετικά άρθρα**. Για κάθε στοιχείο της απάντησης πρέπει να επιστρέφεται το **μονοπάτι του αρχείου** (path) και ο **βαθμός ομοιότητας** (score) που υπολόγισε το σύστημα σας. Στην αναφορά σας, εκτός των άλλων, πρέπει **να αναφέρετε** τις αλλαγές που κάνατε στο σύστημά σας καθώς και τον λόγο που τις κάνατε.

Το πρόγραμμά σας πρέπει να τρέχει από command line (π.χ. "**java -jar queryevaluator.jar**").

#### **B10) BONUS 10 ΜΟΝΑΔΩΝ (5 μονάδες η επιστροφή snippet + 5 μονάδες η γραφική διεπαφή)**

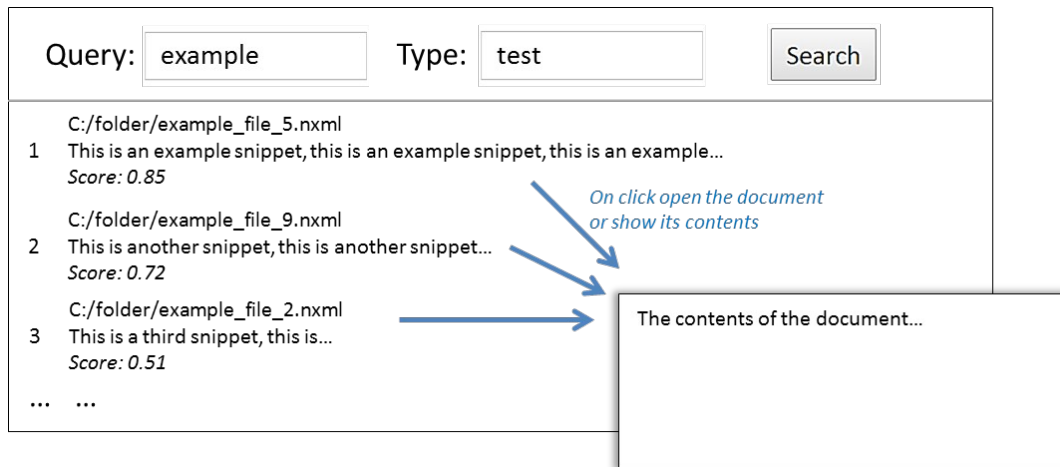
Επεκτείνετε το σύστημά σας ώστε για κάθε στοιχείο της απάντησης να επιστρέφεται και ένα απόσπασμα του άρθρου (snippet) που να περιέχει μία ή περισσότερες λέξεις της επερώτησης. Το snippet μπορείτε να το διαμορφώσετε όπως εσείς νομίζετε, πχ. μπορείτε να χρησιμοποιήσετε κάποια έτοιμη υλοποίηση ενός string matching/searching algorithm. Επίσης, δημιουργήστε μια υποτυπώδη γραφική διεπαφή (ένα παράδειγμα δίνεται στην Εικόνα 2).

➔ **ΠΑΡΑΔΟΤΕΑ:** Ένα αρχείο <<AM1-AM2>>.zip το οποίο θα ανεβάσετε στο moodle το οποίο πρέπει να περιέχει:

- /doc/report.{doc|pdf}: Μια γραπτή αναφορά σχετικά με το τι κάνατε και τι όχι (αναφέρετε και το χρόνο ευρετηρίασης και το χρόνο απόκρισης)
- /src: Με τον κώδικά σας (**ΠΡΟΣΟΧΗ: Να περιέχει τα .java αρχεία, όχι τα .class και να είναι maven project**)
- /dist: Με τα αρχεία .jar τα οποία πρέπει να επαρκούν για την εκτέλεση του ευρετηριαστή και του αποτιμητή επερωτήσεων. Είναι σημαντικό για την τελική βαθμολόγηση της εργασίας σας η ομαλή και εύκολη εκτέλεσή της.

<sup>3</sup> Μπορείτε αρχικά να χρησιμοποιήσετε μία μικρή συλλογή κειμένων, ώστε να σιγουρέψετε ότι τα μοντέλα σας έχουν υλοποιηθεί σωστά.

<sup>4</sup> <https://trec.nist.gov/pubs/call2016.html>, <https://trec-cds.appspot.com/>



Εικόνα 2: Παράδειγμα γραφικής διεπαφής

## ΦΑΣΗ Β (40 μονάδες)

Στη συγκεκριμένη φάση καλείστε να αξιολογήσετε την **αποτελεσματικότητα** του συστήματός σας.

### ΣΥΛΛΟΓΗ ΑΞΙΟΛΟΓΗΣΗΣ: ΕΙΣΑΓΩΓΗ

Η συλλογή αξιολόγησης που θα χρησιμοποιήσετε προέρχεται από το TREC 2014 (θυμηθείτε αυτά που λέγαμε στην ενότητα Αξιολόγηση Αποτελεσματικότητας Ανάκτησης), και όποιος θέλει μπορεί (δεν είναι υποχρεωτικό) να ενημερωθεί για το σκεπτικό αυτού του Clinical Decision Support Track του TREC, διαβάζοντας το άρθρο

Matthew S. Simpson, Ellen M. Voorhees, and William Hersh, Overview of the TREC 2014 Clinical Decision Support Track, URL: <https://pdfs.semanticscholar.org/fcf8/1b7641c0cd7be089051018a53fabfa685da0.pdf>

(Επίσης στην διεύθυνση <https://trec.nist.gov/pubs/trec23/trec2014.html> υπάρχουν papers από ομάδες οι οποίες συμμετείχαν στο διαγωνισμό εκείνης της χρονιάς).

### ΣΥΛΛΟΓΗ ΑΞΙΟΛΟΓΗΣΗΣ: TOPICS and HUMAN JUDGES

Για την αξιολόγηση της αποτελεσματικότητας ανάκτησης (retrieval effectiveness) του συστήματός σας θα χρησιμοποιήσετε τα συνολικά 30 ιατρικά θέματα του αρχείου **topics.xml** (10 από κάθε τύπο κλινικής ερώτησης). Κώδικας για την ανάγνωσή τους δίνεται στο **Παράρτημα ΣΤ**.

Επίσης για κάθε ιατρικό θέμα στο αρχείο **topics.xml**, σας δίνεται:

- ένα σύνολο εγγράφων της συλλογής που είναι πολύ σχετικά για την απάντηση του αντίστοιχου θέματος
- ένα σύνολο εγγράφων της συλλογής που είναι σχετικά για την απάντηση του αντίστοιχου θέματος
- ένα σύνολο εγγράφων της συλλογής που δεν είναι σχετικά για την απάντηση του αντίστοιχου θέματος

Οι παραπάνω πληροφορίες δίνονται στο TSV (Tab-Separated Values) αρχείο **qrels.txt**. Κάθε γραμμή αυτού του αρχείου περιέχει 4 στοιχεία:

1. *topic number*: αριθμός από 1 έως 30 που αναπαριστά το αντίστοιχο ιατρικό θέμα του αρχείου **topics.xml**
2. *αριθμός 0* (δεν χρησιμοποιείται)
3. *document PMCID*: αναγνωριστικό PMC βιοϊατρικού άρθρου από τη συλλογή **Medical Collection**
4. *relevance score*: η σχετικότητα του βιοϊατρικού άρθρου για την απάντηση του ιατρικού θέματος (0 = μη σχετικό, 1 = σχετικό, 2 = πολύ σχετικό)

Για παράδειγμα, η γραμμή «**1 0 1033658 0**» σημαίνει ότι το έγγραφο της συλλογής με PMCID “1033658 ” δεν είναι σχετικό για το ιατρικό θέμα με αριθμό 1.

## ΤΟ ΣΥΣΤΗΜΑ ΑΥΤΟΜΑΤΗΣ ΑΞΙΟΛΟΓΗΣΗΣ: ΕΙΣΑΓΩΓΗ, ΜΕΘΟΔΟΣ ΚΑΙ ΜΕΤΡΑ

Συγκεκριμένα **δημιουργήστε** ένα πρόγραμμα (μπορείτε να το πείτε **IRQualityEvaluator**) που θα αυτοματοποιεί τον υπολογισμό των παραπάνω μέτρων. Το πρόγραμμα πρέπει να μπορεί να διαβάσει το αρχείο των θεμάτων (topics), να στέλνει επερωτήσεις στο σύστημα που φτιάξατε στη Φάση Α, και κατόπιν να αποθηκεύει τα 1000 κορυφαία αποτελέσματα (όπως αυτά επιστρέφονται από το σύστημά σας) σε ένα αρχείο με όνομα «**results.txt**», στη μορφή:

**TOPIC\_NO Q0 PMCID RANK SCORE RUN\_NAME**

όπου **TOPIC\_NO** είναι ο αριθμός του ιατρικού θέματος, **Q0** μια σταθερά (βάλτε τον αριθμό 0), **PMCID** είναι το PMC ID του εγγράφου που ανακτήθηκε, **RANK** είναι η κατάταξη (αριθμός από 1 έως 1000) του εγγράφου που ανακτήθηκε (το έγγραφο με RANK 1 είναι αυτό με τον υψηλότερο βαθμό ομοιότητας, κ.ο.κ), **SCORE** είναι ο βαθμός ομοιότητας που έδωσε το σύστημά σας στο συγκεκριμένο έγγραφο, και **RUN\_NAME** είναι ένα αναγνωριστικό για το σύστημά σας (αυτό είναι χρήσιμο σε περίπτωση που θέλετε να δοκιμάσετε διαφορετικές παραμετροποιήσεις του συστήματός σας). Το αρχείο πρέπει να είναι **ταξινομημένο** ως προς το **RANK** (δηλαδή στην πρώτη γραμμή θα είναι το έγγραφο με RANK 1, στην δεύτερη το έγγραφο με RANK 2, κ.ο.κ.).

Εν συνεχεία το πρόγραμμα σας πρέπει να διαβάσει το αρχείο με τα μερικά αποτελέσματα συνάφειας (**qrels.txt**) και συγκρίνοντας το με τα αποτελέσματα του συστήματός σας (στο **results.txt**), να υπολογίζει τις τιμές των μέτρων που περιγράφονται παρακάτω **για κάθε ιατρικό θέμα**, και θα τις αποθηκεύει σε ένα TSV αρχείο με όνομα «**eval\_results.txt**» στην μορφή:

**TOPIC\_NO BPREF\_VALUE AVEP\_VALUE NDCG\_VALUE**

Επειδή για πολλά από τα έγγραφα τις συλλογές δεν γνωρίζουμε αν είναι σχετικά ή όχι για την απάντηση ενός ή περισσοτέρων θεμάτων, πρέπει να χρησιμοποιήσετε κάποιες μετρικές αξιολόγησης που είναι σχεδιασμένες για τέτοιες περιπτώσεις. Οι μετρικές που πρέπει να χρησιμοποιήσετε είναι οι παρακάτω:

- bpref [1]
- AveP' [2]
- nDCG' [2]

Για να καταλάβετε και να υλοποιήσετε αυτές τις μετρικές πρέπει να διαβάσετε τα παρακάτω 2 άρθρα:

- [1] Chris Buckley and Ellen M. Voorhees, "Retrieval evaluation with incomplete information.", Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2004.
- [2] Tetsuya Sakai, "Alternatives to bpref.", Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007.

Χρησιμοποιώντας τις παραπάνω μετρικές, θα αξιολογήσετε την αποτελεσματικότητα του συστήματός σας. Παρατηρώντας τα αποτελέσματα της αξιολόγησης αποτελεσματικότητας μπορείτε να κάνετε ό,τι παραλλαγή νομίζετε στο σύστημά σας (π.χ. στη βάρυνση του ευρετηρίου, στον επεξεργαστή επερωτήσεων, στη συνάρτηση υπολογισμού του βαθμού συνάφειας, κλπ.) **ώστε να μεγιστοποιήσετε την αποτελεσματικότητα του συστήματός σας (που είναι το τελικό ζητούμενο) και να αυξήσετε την πιθανότητα να βγείτε νικητές!**

## ΤΟ ΣΥΣΤΗΜΑ ΑΥΤΟΜΑΤΗΣ ΑΞΙΟΛΟΓΗΣΗΣ: ΔΟΜΗΣΗ ΚΩΔΙΚΑ

Μπορείτε αυτό το τμήμα του λογισμικού να το κάνετε γενικό ώστε να μπορεί να χρησιμοποιηθεί για την αξιολόγηση οποιουδήποτε Συστήματος Ανάκτησης Πληροφοριών. Για το σκοπό βολεύει να ορίσετε ένα interface που πρέπει να προσφέρει ένα τέτοιο σύστημα και ο κώδικας που θα γράψετε για τη φάση Β να εξαρτάται μόνο από αυτό το interface. Σχετικό pattern της τεχνολογίας λογισμικού είναι το: ProxyPattern (σχετικό υλικό και παραδείγματα μπορείτε να βρείτε σε πολλά μέρη, ενδεικτικά

[https://en.wikipedia.org/wiki/Proxy\\_pattern](https://en.wikipedia.org/wiki/Proxy_pattern),

[https://www.tutorialspoint.com/design\\_pattern/proxy\\_pattern.htm](https://www.tutorialspoint.com/design_pattern/proxy_pattern.htm),

<https://dzone.com/articles/design-patterns-proxy>).

## ΑΝΑΦΟΡΑ ΜΕ ΑΠΟΤΕΛΕΣΜΑΤΑ ΑΞΙΟΛΟΓΗΣΗΣ

Συντάξτε σχετική αναφορά που να περιγράφει τις καλύτερες επιδόσεις του συστήματός σας και τι κάνατε για να τις επιτύχετε. Επίσης, στην αναφορά πρέπει να αναλύετε τα αποτελέσματα της πειραματικής αξιολόγησης (με ποια ιατρικά θέματα το σύστημά σας τα πήγε καλά, με ποια όχι, που μπορεί να οφείλετε η επιτυχία/αποτυχία, κτλ.). Επιπλέον θα πρέπει να δοθούν σχετικά στατιστικά στοιχεία, π.χ. median/average values, min, max, κ.ο.κ. Για την κατασκευή των γραφημάτων μπορείτε να χρησιμοποιήσετε τις δυνατότητες του excel.

Το καλύτερο σύστημα θα είναι αυτό με τις περισσότερες «νίκες» σε το σύνολο των ιατρικών θεμάτων (για ένα ιατρικό θέμα, το σύστημα που κερδίζει είναι αυτό με το **υψηλότερο άθροισμα** των τιμών των τριών μετρικών, ενώ σε περίπτωση ισοπαλίας, θα υπολογίζονται οι δεύτερες θέσεις, κ.ο.κ.).

➔ **ΠΑΡΑΔΟΤΕΑ:** Ένα αρχείο <<AM1-AM2>>.zip το οποίο να περιέχει

- o /doc/report.{doc|pdf}: Η γραπτή αναφορά στην οποία πρέπει να περιγράψετε τι ακριβώς κάνατε και τα αποτελέσματα της πειραματικής αξιολόγησης (χρησιμοποιείτε το πρότυπο που σας έχει δοθεί - HY463\_Report\_Template\_2019).
- o /doc/eval\_results.txt: Το αρχείο eval\_results.txt όπως παράγεται από το πρόγραμμά σας.
- o /src: Με τον κώδικά σας (**ΠΡΟΣΟΧΗ: Να περιέχει τα .java αρχεία, όχι τα .class και να είναι maven project**)
- o /dist: Με τα αρχεία .jar τα οποία πρέπει να επαρκούν για την εκτέλεση του προγράμματός σας. Είναι σημαντικό για την τελική βαθμολόγηση της εργασίας σας η ομαλή και εύκολη εκτέλεσή της.

*Καλή εργασία!*

## ΠΑΡΑΡΤΗΜΑΤΑ

### ΠΑΡΑΡΤΗΜΑ Α – Η ΣΥΛΛΟΓΗ

Η συλλογή που θα χρησιμοποιήσετε περιέχει άρθρα σχετικά με την βιοϊατρική τα οποία έχουν εξαχθεί από τη ψηφιακή βάση δεδομένων «PubMed Central» (PMC). Κάθε άρθρο της συλλογής αναπαριστάται ως ένα NXML αρχείο (XML αρχείο κωδικοποιημένο με χρήση της βιβλιοθήκης «[NLM Journal Archiving and Interchange Tag Library](#)» και αναγνωρίζεται μοναδικά από τον αριθμό PMCID (το όνομα κάθε αρχείου είναι στην ουσία ο αριθμός PMCID του αντίστοιχου άρθρου).

Κάθε NXML αρχείο περιλαμβάνει πολλές ετικέτες όπως: αναγνωριστικά του άρθρου, αναγνωριστικό περιοδικού, τίτλος περιοδικού, εκδότης περιοδικού, τίτλος άρθρου, περίληψη άρθρου, συγγραφείς, κυρίως σώμα άρθρου, ημερομηνία δημοσίευσης, αναφορές, και πολλά άλλα.

Στα πλαίσια αυτής της εργασίας μας ενδιαφέρουν οι παρακάτω 8 ετικέτες:

- Αναγνωριστικό PMCID
- Τίτλος άρθρου
- Συγγραφείς του άρθρου
- Περίληψη άρθρου
- Κυρίως σώμα άρθρου
- Κατηγορίες άρθρου
- Περιοδικό που δημοσιεύτηκε
- Εκδότης περιοδικού

Στο Παράρτημα Β θα βρείτε κώδικα για την ανάγνωση των παραπάνω ετικετών. Φυσικά, αν το επιθυμείτε, είστε ελεύθεροι να χρησιμοποιήσετε όποιες άλλες ετικέτες θέλετε.

### ΠΑΡΑΡΤΗΜΑ Β – ΑΝΑΓΝΩΣΗ ΒΙΟΪΑΤΡΙΚΩΝ ΑΡΘΡΩΝ | ΔΙΑΧΩΡΙΣΜΟΣ

#### ΑΛΦΑΡΙΘΜΗΤΙΚΟΥ ΣΕ ΛΕΞΕΙΣ

Για την ανάγνωση ενός βιοϊατρικού άρθρου μπορείτε να χρησιμοποιήσετε την βιβλιοθήκη «**BioReader.jar**». Αφού προσθέσετε την βιβλιοθήκη στο πρόγραμμά σας, με τον παρακάτω κώδικα μπορείτε να διαβάσετε τις ετικέτες ενός άρθρου:

```
import gr.uoc.csd.hy463.NXMLFileReader;
import java.io.File;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.HashSet;

public class MYEXAMPLE {

    public static void main(String[] args) throws UnsupportedEncodingException, IOException
    {

        File example = new File("C:\\dataset\\clinic\\3536594.xml");

        NXMLFileReader xmlFile = new NXMLFileReader(example);
        String pmcid = xmlFile.getPMCID();
        String title = xmlFile.getTitle();
        String abstr = xmlFile.getAbstr();
        String body = xmlFile.getBody();
        String journal = xmlFile.getJournal();
        String publisher = xmlFile.getPublisher();
        ArrayList<String> authors = xmlFile.getAuthors();
        HashSet<String> categories =xmlFile.getCategories();

        System.out.println("- PMC ID: " + pmcid);
        System.out.println("- Title: " + title);
        System.out.println("- Abstract: " + abstr);
        System.out.println("- Body: " + body);
        System.out.println("- Journal: " + journal);
        System.out.println("- Publisher: " + publisher);
        System.out.println("- Authors: " + authors);
        System.out.println("- Categories: " + categories);

    }
}
```

Το παρακάτω κομμάτι κώδικα εκτυπώνει όλες τις λέξεις ενός αλφαριθμητικού:

```
String delimiter = "\t\n\r\f ";

String line = "hello, my name is Pavlos, how are you?";
StringTokenizer tokenizer = new StringTokenizer(line, delimiter);
while(tokenizer.hasMoreTokens() ) {
    String currentToken = tokenizer.nextToken();
    System.out.println(currentToken);
}
```

## ΠΑΡΑΡΤΗΜΑ Γ – ΠΡΟΣΒΑΣΗ ΣΤΑ ΑΡΧΕΙΑ ΕΝΟΣ ΦΑΚΕΛΟΥ

Ενδεικτική πρόσβαση σε όλα τα αρχεία ενός φακέλου (συμπεριλαμβανομένων των αρχείων σε υποφακέλους αναδρομικά):

```
import java.io.File;

public class ReadAllFiles {

    public static void main(String[] args) {
```



```

        File folder = new File("C:\\dataset\\clinic\\");
        listFilesForFolder(folder);

    }

    public static void listFilesForFolder(File folder) {
        for (File fileEntry : folder.listFiles()) {
            if (fileEntry.isDirectory()) {
                listFilesForFolder(fileEntry);
            } else {
                System.out.println(fileEntry.getAbsolutePath());
            }
        }
    }
}

```

## ΠΑΡΑΡΤΗΜΑ Δ - STEMMING

Ενδεικτική χρήση του Stemmer της μηχανής αναζήτησης mitos.

```

import mitos.stemmer.Stemmer;

public class TestStemmer {

    public static void main(String[] args){
        Stemmer.Initialize();
        System.out.println(Stemmer.Stem("ending"));
        System.out.println(Stemmer.Stem("συγχωνευμένος"));
    }
}

```

## ΠΑΡΑΡΤΗΜΑ Ε - RANDOM ACCESS FILE

Ενδεικτική ανάγνωση και εγγραφή σε random access αρχείο:

```

import java.io.*;

public class WRFile
{
    public static void main(String[] args)
    {
        RandomAccessFile file = null;
        try {
            file = new RandomAccessFile("rand.txt", "rw");

            //Writing to the file
            file.writeChar('A');
            file.writeChar('B');
            file.writeChar('C');
            file.writeChar('D');

            file.seek(0);    // get first item
            System.out.println(file.readChar());

            file.seek(4);    //get third item (char size = 2 byte, 2*2)
            System.out.println(file.readChar());

            file.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```
        } catch(Exception e) {}  
    }  
}
```

## ΠΑΡΑΡΤΗΜΑ ΣΤ - ΑΝΑΓΝΩΣΗ ΙΑΤΡΙΚΩΝ ΑΝΑΦΟΡΩΝ ΑΠΟ ΑΡΧΕΙΟ

Για την ανάγνωση ενός του αρχείου με τις ιατρικές αναφορές μπορείτε να χρησιμοποιήσετε την βιβλιοθήκη «**BioReader.jar**». Αφού προσθέσετε την βιβλιοθήκη στο πρόγραμμά σας, με τον παρακάτω κώδικα μπορείτε να διαβάσετε τις όλες τις ιατρικές αναφορές:

```
import gr.uoc.csd.hy463.Topic;  
import gr.uoc.csd.hy463.TopicsReader;  
import java.util.ArrayList;  
  
public class MYEXAMPLE {  
  
    public static void main(String[] args) throws Exception {  
  
        ArrayList<Topic> topics = TopicsReader.readTopics("C:\\\\dataset\\\\ topics.xml");  
        for (Topic topic : topics) {  
            System.out.println(topic.getNumber());  
            System.out.println(topic.getType());  
            System.out.println(topic.getSummary());  
            System.out.println(topic.getDescription());  
            System.out.println("-----");  
        }  
  
    }  
}
```