Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών
HY463 – Συστήματα Ανάκτησης Πληροφοριών
Άνοιξη 2022
Διδάσκων: Γιάννης Τζίτζικας
Υπεύθυνοι Βοηθοί: Όλοι

Εργασία Μαθήματος (project)
Άτομα ανά ομάδα: 1 ή 2
Αξία: 25% του τελικού σας βαθμού
Φάση Α: 8 Απριλίου – 10 Μαΐου (60%)
Φάση Β: 10 Μαΐου – 28 Μαΐου (40%)
Bonus: 10% επί του βαθμού σε όποιον παραδώσει τουλάχιστον 2 μέρες πριν την προθεσμία (ισχύει και για τις δύο φάσεις)

Αυτό είναι το Πρώτο Σκέλος της εκφώνησης που περιγράφει τα γενικά βήματα. Το **Δεύτερο Σκέλος της εκφώνησης,** που έχει συγκεκριμένες πληροφορίες για τη συλλογή που θα χρησιμοποιήσετε, περιγράφεται στο αρχείο **project_2022_B_BioDocumentspdf**.

Α. Δημιουργία ενός Συστήματος Ανάκτησης Πληροφορικών και Υποσυστήματος για την Αξιολόγησή του: Τα Γενικά Βήματα

Σκοπός αυτής της εργασίας είναι να κατανοήσετε τις βασικές έννοιες και τεχνικές, φτιάχνοντας εξ' αρχής το λειτουργικό πυρήνα ενός δικού σας Συστήματος Ανάκτησης Πληροφοριών (ΣΑΠ) (Information Retrieval System). Στην **Φάση Α** καλείστε να φτιάξετε το σύστημα, ενώ στη **Φάση Β** να το δοκιμάσετε και να το αξιολογήσετε σε μια πραγματική συλλογή. Πληροφορίες για τη συλλογή υπάρχουν στο **Δεύτερο Σκέλος της Εκφώνησης**.

Σχόλια για την εκπόνηση της εργασίας από ομάδες
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 1ο μέλος της ομάδας μπορεί να ασχοληθεί κυρίως (όχι αποκλειστικά) με τη Φάση Α και το 2ο κυρίως (όχι αποκλειστικά) με τη Φάση Β. Λάβετε υπόψη ότι η Φάση Α απαιτεί περισσότερη δουλειά από τη Φάση 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 εξαρχής στο interface μέσω του οποίου ο κώδικας του συστήματος της Φάσης Β θα στέλνει ερωτήματα και θα λαμβάνει απαντήσεις από το σύστημα της Φάσης Α.

Σ2. Καλείστε να δηλώσετε τις ομάδες έως τη Τετάρτη 7 Απριλίου (πληροφορίες υπάρχουν στην αντίστοιχη ανακοίνωση στο elearn). Κατά τη διάρκεια της προφορικής εξέτασης (τον Ιούνιο) 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): Μηχανισμός Ευρετηρίασης και Μηχανισμός Απάντησης Ερωτήσεων

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

Διαδικασία Ευρετηρίασης (Β1-Β6)

Β1) (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[1] (πληροφορίες για τη συλλογή, τα XML αρχεία και τις ετικέτες που μας ενδιαφέρουν θα βρείτε στο **Β' Σκέλος της Εκφώνησης**. 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 ενδιαφέρουν από ένα έγγραφο

[1] Μπορείτε να δείτε (ή να θυμηθείτε) τα παραδείγματα που υπάρχουν στην ενότητα Java Collection Framework του HY252, μια παλαιότερη έκδοση υπάρχει εδώ: http://www.csd.uoc.gr/~hy252/Lectures07/pdf/CS252CollectionClassesInterfaces07.pdf

της συλλογής υπάρχει **Β' Σκέλος της Εκφώνησης** ενώ πληροφορίες για το διαχωρισμό ενός αλφαριθμητικού σε λέξεις, υπάρχει στο **Παράρτημα Α-1**. Δοκιμάστε την υλοποίησή σας σε ένα αρχείο της συλλογής. ("**Medical Collection**")

Β2) (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 Β1) 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 Α-2.** Δοκιμάστε την υλοποίησή σας στη συλλογή εγγράφων ("**Medical Collection**").

Β3) (3) Στη συνέχεια θα πρέπει να υποστηρίξετε τη διαδικασία του *stemming* των λέξεων που διαβάζετε στα Β1 και Β2, 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 Α-3** για το πώς θα τη χρησιμοποιήσετε. Ενδεικτικά μπορείτε να τρέξετε και τη γραφική διεπαφή του Stemmer με "java –jar Stemmer.jar". Μπορείτε όμως να χρησιμοποιήσετε και οποιοδήποτε άλλο κώδικα θέλετε για stemming της αγγλικής γλώσσας.

Β4) (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).

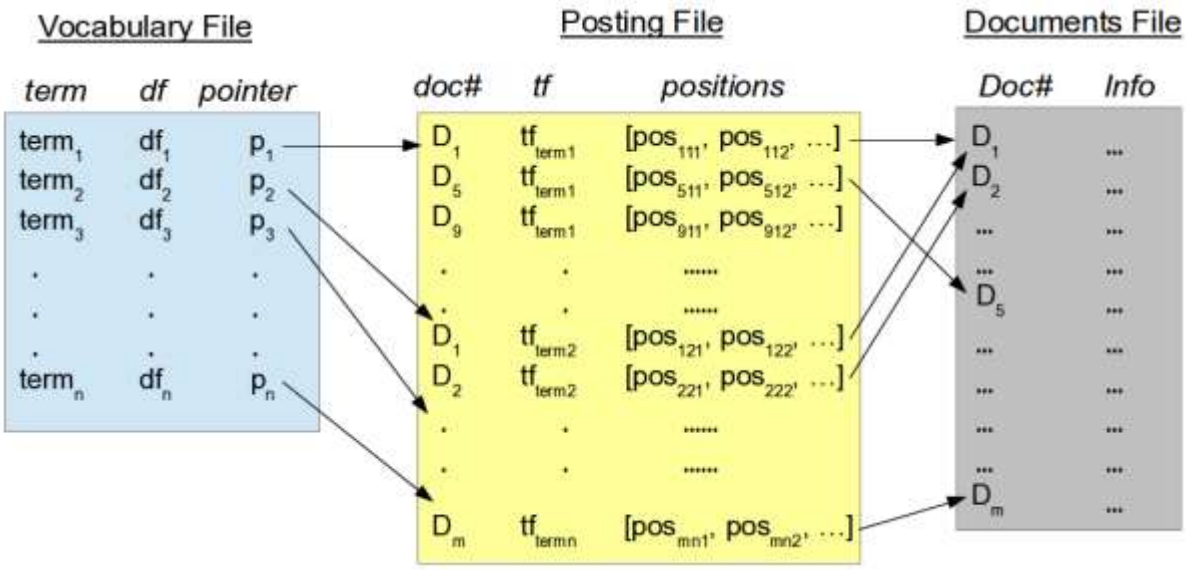
Β5) (7) Επεκτείνετε το σύστημα ώστε να δημιουργεί ένα ακόμη αρχείο με όνομα **DocumentsFile.txt** στον φάκελο **CollectionIndex** που θα καταγράφει για κάθε αρχείο της συλλογής μία τριάδα αποτελούμενη από ένα μοναδικό αριθμητικό αναγνωριστικό, το πλήρες μονοπάτι του αρχείου, και το μήκος του διανύσματος (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.

Β6) (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. Δημιουργείστε 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. Δημιουργείστε 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 $\mathbf{t}_i$ 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 $\mathbf{t}_i$ (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. 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.

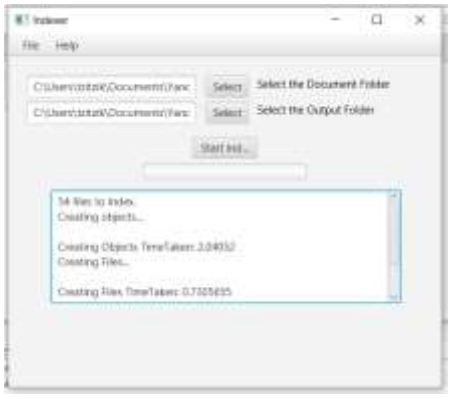


Εικόνα 1. *Το ανεστραμμένο ευρετήριο*

Συνίσταται η επιλογή (2), η οποία απαιτεί τη δημιουργία ενός μοναδικού PostingFile καθώς και η χρήση *random access files*[2] με σκοπό την αποφυγή σειριακής σάρωσης του αρχείου (δείτε **Παράρτημα Α-4**) .

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 απόδοσή** 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 αλλάζεται 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). Ενδεικτικά μια γραφική διεπαφή για τον Indexer ακολουθεί. Επιτρέπει στο χρήση να διαλέξει το φάκελο στον οποίο είναι τα έγγραφα προς ευρετηρίαση, το φάκελο στο οποίο θα φτιαχτεί το ευρετήριο, και να ξεκινήσει τη διαδικασία της ευρετηρίασης και στο τέλος να δει διάφορα στατιστικά για το τι έγινε (πόσα έγγραφα ευρετηριάστηκαν, πόσο χρόνο πήρε, κ.α.).



[2] http://www.csd.uoc.gr/~hy252/html/Lectures2012/assist2012/CS252FilesStreams12.pdf
http://docs.oracle.com/javase/6/docs/api/java/io/RandomAccessFile.html
http://www.dailyfreecode.com/code/reads-writes-random-access-file-1204.aspx

Σημείωση: Δείτε τις διαφάνειες για τη δημιουργία και το merging of partial indexes (και αυτά που κάναμε στις διαλέξεις και τις επιπλέον διαφάνειες που υπάρχουν στην ενότητα του project στο elearn). Χωρίς αυτόν τον τρόπο θα καταφέρετε να ευρετηριάσετε τη μικρή συλλογή, αλλά όχι τη μεγάλη.

*Αποτίμηση επερωτήσεων (***B*7- B*10)***

B7) (5) Γράψτε ένα νέο πρόγραμμα σε Java (ξεχωριστό πακέτο στο project σας) το οποίο θα χρησιμοποιεί το ανεστραμμένο ευρετήριο που κατασκευάσατε ώστε να υποστηρίξει τη διαδικασία αναζήτησης. 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 συλλογής (Documents File) θεωρούμε (στη γενική περίπτωση) 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)[3]. 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). Μπορείτε εναλλακτικά ή συμπληρωματικά να υποστηρίξετε και κάποιο άλλο μοντέλο ανάκτησης (Okapi BM25 ή άλλο).

B9) (15) Σας δίνεται μια συλλογή από θέματα (topics), για κάθε ένα από αυτά πρέπει να παράγετε μια επερώτηση, δείτε τις πληροφορίες στο **2ο Αρχείο της Εκφώνησης**. Καλείστε να **επεκτείνετε 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 το σύστημα να ανακτά **σχετικά άρθρα**. Για κάθε στοιχείο της απάντησης πρέπει να επιστρέφετε το **μονοπάτι του αρχείου** (path) και ο **βαθμός ομοιότητας** (score) που υπολόγισε το σύστημά σας. Στην αναφορά σας, εκτός των άλλων, πρέπει **να αναφέρετε** τις αλλαγές που κάνατε στο σύστημά σας καθώς και τον λόγο που τις κάνατε.

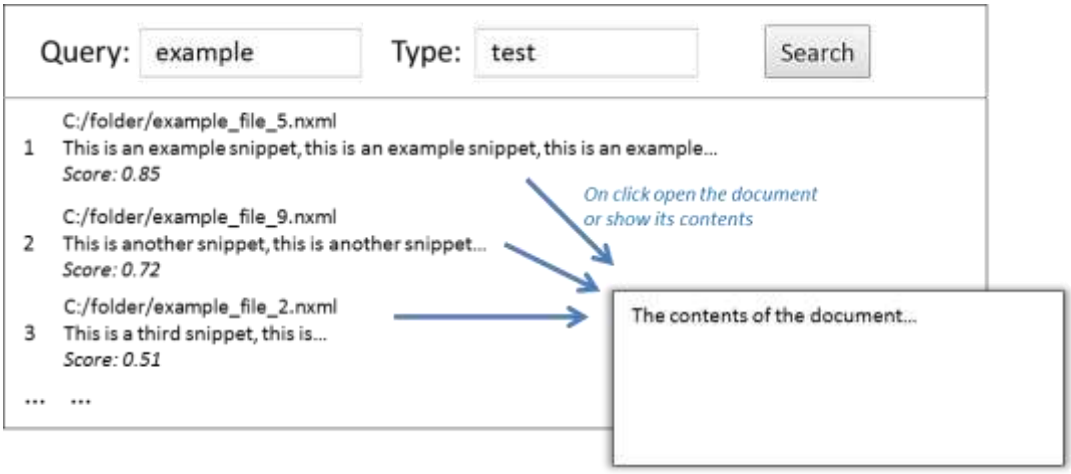
Το πρόγραμμά σας πρέπει να τρέχει από command line (π.χ. "**java -jar queryevaluator.jar**").

B10) <u>BONUS</u> 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)

<u>ΠΑΡΑΔΟΤΕΑ ΦΑΣΗΣ-Α</u>: 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 <u>.java</u> αρχεία, <u>όχι τα .class</u>, ή ιδανικά ολόκληρο το Netbeans ή Eclipse project όπως προκύπτει από το export)**
- /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.

[3] Μπορείτε αρχικά να χρησιμοποιήσετε μία μικρή συλλογή κειμένων, ώστε να σιγουρέψετε ότι τα μοντέλα σας έχουν υλοποιηθεί σωστά.



Εικόνα 2: *Παράδειγμα γραφικής διεπαφής για τον αποτιμητή επερωτήσεων*

ΦΑΣΗ Β (40 μονάδες): Μηχανισμός Αξιολόγησης

Στη συγκεκριμένη φάση καλείστε να εφαρμόσετε και να αξιολογήσετε την **αποτελεσματικότητα** του συστήματός σας σε μια συγκεκριμένη συλλογή και με συγκεκριμένου τύπου πληροφοριακές ανάγκες και συγκεκριμένη συλλογή αξιολόγησης. Εν συντομία καλείστε να **δημιουργήστε** ένα υποσύστημα (μπορείτε να το πείτε **IRQualityEvaluator**) που θα αυτοματοποιεί τον υπολογισμό των μέτρων αξιολόγησης. Το πρόγραμμα πρέπει να μπορεί να διαβάζει το αρχείο των θεμάτων (**topics**) της συλλογής αξιολόγησης, να στέλνει επερωτήσεις στο σύστημα που φτιάξατε στη Φάση Α, να αποθηκεύει τα κορυφαία αποτελέσματα (όπως αυτά επιστρέφονται από το σύστημά σας) σε ένα αρχείο με όνομα «**results.txt**», εν συνεχεία να διαβάζει το αρχείο με τα μερικά αποτελέσματα συνάφειας από τη συλλογή αξιολόγησης (**qrels.txt**) και συγκρίνοντας το με τα αποτελέσματα του συστήματος σας (στο **results.txt**), να υπολογίζει τις τιμές των μέτρων αξιολόγησης **για κάθε θέμα**, και θα τις αποθηκεύει σε ένα TSV (tab-separated values) αρχείο με όνομα «**eval_results.txt**». Χρησιμοποιώντας τις μετρικές, θα αξιολογήσετε συνολικά 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**
Η συλλογή αξιολόγησης και βοηθητικές πληροφορίες περιγράφονται στο αρχείο **Δεύτερο Σκέλος της Εκφώνησης**.

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://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 συστήματος 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 ή όποιου άλλου συστήματος θέλετε ή είστε εξοικειωμένοι.

Το καλύτερο σύστημα θα είναι αυτό με τις περισσότερες «νίκες» σε ένα **νέο** σύνολο 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 ΦΑΣΗΣ-Β: Ένα αρχείο <<AM1-AM2>>.zip το οποίο να περιέχει

- /doc/report.{doc|pdf}: Η γραπτή αναφορά στην οποία πρέπει να περιγράψατε τι ακριβώς κάνατε και τα αποτελέσματα της πειραματικής αξιολόγησης (συστήνετε να χρησιμοποιήσετε το πρότυπο που σας έχει δοθεί - HY463_Report_Template_2017).
- /doc/**eval_results.txt**: Το αρχείο **eval_results.txt** όπως παράγεται από το πρόγραμμά σας.
- /src: Με τον κώδικά σας **(ΠΡΟΣΟΧΗ: Να περιέχει τα .java αρχεία, όχι τα .class, ή ιδανικά ολόκληρο το Netbeans ή Eclipse project). Ιδανικά να είναι όλο το project, άρα και η νέα έκδοση του συστήματος της φάσης Α.**
- /dist: Με τα αρχεία .jar τα οποίο 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-1: ΔΙΑΧΩΡΙΣΜΟΣ ΑΛΦΑΡΙΘΜΗΤΙΚΟΥ ΣΕ ΛΕΞΕΙΣ

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 Α-2: 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 Α-3: 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 Α-4: 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(Exception e) {}
        }
}
```