Ανάκτηση Πληροφορίας
Τεκμηρίωση Εργασίας

Ημερομηνία: 12/6/2012
Ονόματα/Επώνημα :

Πετρίδης Παναγιώτης (1831)
Τολιόπουλος Θεόδωρος (1841)
Χατζηχριστοδούλου Ζωη (1851)

Εισαγωγικά

Το πρόγραμμα αναπτύχθηκε σύμφωνα με τα ζητούμενα της εκφώνισης και ανταποκρίνεται σε όλα όσα έχουν ζητηθεί. Υλοποιήθηκαν δύο (2) επεκτάσεις (δείτε ανάλογο παράρτημα στο τέλος).

Γενικά το πρόγραμμα χρησημοποιεί συνολικά 11 κλάσεις. Κάθε μια απο αυτές υλοποιεί διαφορετικό κομμάτι του προβληματος.

Οι κλάσεις που χρησημοποιήθηκαν- Επεξήγηση

1) Aganaktish.java

Η κλαση αυτή αποτελεί την κύρια κλάση του προγραμματος και σε αυτην γίνεται η έναρξη όλων των απαραιτήτων διεργασιών για την λειτουργια του προγράμματος. Δηλαδή το φόρτωμα των απαραίτητων συλλογών και η ενεργοποίηση του User Interface. Περιέχει συναρτήσεις οι οποίες εχουν γενικό χαρακτήρα.

Συναρτήσεις της κλάσης αυτης:

TreeMap<Integer, ArrayList<PairDouble>> **runqueries**(String path, String method, int k),

Η συνάρτηση αυτή αναλαμβάνει την υποβολή τον queries στο συστημα με την χρηση του Vector μοντελου. Λαμβανει τα ερωτηματα απο το txt στο path και τρεχει τα ερωτήματε χρησημοποιόντας μια απο τις μεθόδους (method) που έχουν υλοποιηθεί και επιστρέφει για καθε ερώτημα τα k πρώτα αποτελέσματα. Τα αποτελέσματα επιστρέφονται σε TreeMap , με key το id του ερωτήματος και value τα λιστα με τα αποτελέσματα που επιστρεφτηκαν. (Περισσότερα για την PairDouble παρακάτω.)

TreeMap<Integer, ArrayList<String>> **getRelevant**(String path),

Όμοια με την runqueries. Λαμβάνει απο το txt στο path τα αποτελέσματα και τα επιστρέφει σε TreeMap οπου κλειδί είναι το id του ερωτήματος και value τα λίστα με τα σχετικά εγραφα.

TreeMap<Integer,double[]> **calculateStats**(TreeMap ourMap, TreeMap hisMap),

Λαμβάνει τα αποτελέσματα των δύο παραπάνω συναρτήσεων και συγρίνοντας τους υπολογίζει το recall και το precision για καθε ερώτημα. Τα αποτελέσματα επιστρέφονται σε TreeMap με κλειδή το id του ερωτήματος και value τα recall και precision.

String[] **getTxt**()

Επιστρέφει το σύνολο των ονομάτων των αρχείων που βρίσκονται σε μία συλλογή . Χρησημοποιείται για την εμφάνιση των αρχείων ωστε να τα επεξεργαστεί ο χρηστης κατάλληλα (διαγραφη αρχείου απο συλλογη, επιβεβαίωση προσθεσης).

2)CollectionClass.java

Η κλάση CollectionClass υλοποιεί τις συλλογές εγγράφων. Μία συλλογή εγγράφων είναι ένας φάκελος που περιέχει μέσα δύο υποφακέλους. Ο ένας, με όνομα txt, περιέχει όλα τα έγγραφα της συλλογής, στην συγκεκριμένη υλοποίηση μόνο text documents (.txt), και ο δεύτερος φάκελος, με όνομα catalog, περιέχει τον αντεστραμμένο κατάλογο που δημιουργήθηκε με βάση τα έγραφα της συλλογής και είναι σε μορφή .dat . Οι συλλογές MED και CRAN, περιέχουν και έναν ακόμα φάκελο, με όνομα query, που έχει δύο αρχεία κειμένου. Το ένα (query) περιέχει τα ερωτήματα για την αντίστοιχη συλλογή και το αλλο (relevant) περιέχει τα σχετικά έγγραφα για το κάθε ερώτημα. Οι φάκελοι των συλλογών βρίσκονται σε έναν φάκελο με όνομα Sylloges που πρέπει να ειναι μαζί με το .jar . Τέλος μαζί με τον φάκελο Sylloges πρέπει να υπάρχει και το αρχείο Sylloges.dat, το οποίο περιέχει ένα ArrayList από String με τα ονόματα των συλλογών που υπάρχουν.

Η κλάση CollectionClass περιέχει τις εξής συναρτήσεις:

void **loadCollections**(),

Η συνάρτηση loadCollections() καλείται μόλις ξεκινήσει το πρόγραμμα και φορτώνει από το αρχείο Sylloges.dat όλες τις διαθέσιμες συλλογές και τις αποθηκεύει στην static μεταβλητή Sylloges.

void **createCollection**(String txtPath, String collectionName),

Η συνάρτηση createCollection χρησιμοποιείται για να δημιουργήσει μία καινούρια συλλογή. Η παράμετρος collectionName είναι για το όνομα της καινούριας συλλογης. Η παράμετρος txtPath είναι το path του φακέλου, ο οποίος περιέχει τα documents από τα οποία θα δημιουργηθεί η συλλογή. Η παράμετρος txtPath

μπορεί να πάρει επίσης και το path για ένα μονο document , για παράδειγμα ...\med.txt , ώστε να δημιουργήσει μία συλλογή απο τα documents med_docs και cran_docs ή οποιαδήποτε άλλα documents με παρόμοια μορφή. Η συνάρτηση createCollection ελέγχει αν το txtPath είναι φάκελος ή document και καλεί είτε την createCollectionSeveralTxt είτε την createCollectionOneTxtFile αντίστοιχα. Τέλος εισάγει το όνομα της συλλογης στην μεταβλητή Sylloges και ανανεώνει το αρχείο Sylloges.dat .

void **createCollectionSeveralTxt**(String txtPath, String pathNewCollection),

Η συνάρτηση createCollectionSeveralTxt χρησιμοποιείται για τη δημιουργεία νέας συλλογης από πολλά documents. Παίρνει ως πρώτο όρισμα το path του φακέλου που είναι τα documents που θέλουμε να περιέχονται στη συλλογή και ως δεύτερο όρισμα το path της συλλογής που δημιουργούμε. Στη συνέχεια δημιουργεί ένα TreeMap το οποίο θα είναι ο κατάλογος της καινούριας συλλογής. Αντιγράφει ένα ένα τα αρχεία από το txtPath στο pathNewCollection\txt και ταυτόχρονα ενημερώνει τον κατάλογο για το συγκεκριμένο document με τη συνάρτηση createCatalog. Τέλος αποθηκεύει τον κατάλογο στην static μεταβλητή collectionCatalog και καλεί την συνάρτηση saveCatalog για την αποθήκευση του στον φάκελο catalog της συλλογής με όνομα catalog.dat .

void **createCollectionOneTxtFile**(String txtPath, String pathNewCollection),

Η συνάρτηση createCollectionOneTxtFile χρησιμοποιείται για τη δημιουργεία νέας συλλογης από ένα document. Παίρνει ως πρώτο όρισμα το path του document από το οποίο θα δημιουργήσουμε τη συλλογή και ως δεύτερο όρισμα το path της συλλογής που δημιουργούμε. Δημιουργεί ένα TreeMap, το οποίο είναι ο κατάλογος της νέας συλλογής. Στη συνέχεια ξεκινάει το parse του document. Κάθε φορά που βρίσκει στο αρχικό document το .I δημιουργεί ένα καινούριο document στην νέα συλλογή και ανανεώνει τον κατάλογο με την createCatalog. Τέλος αφού έχει ολοκληρωθεί η διαδικασία τις διαχώρισης τον document, αποθηκεύει τον κατάλογο στην static μεταβλητή collectionCatalog και καλεί την συνάρτηση saveCatalog για την αποθήκευση του στον φάκελο catalog της συλλογής με όνομα catalog.dat .

void **openCollection**(String collection),

Η συνάρτηση openCollection παίρνει ως όρισμα το όνομα μιας αποθηκευμένης συλλογής και αποθηκεύει στην static μεταβλητή collectionPath το path της συλλογής, στην static μεταβλητή docNum κρατάει τον αριθμό των documents που έχει η συλλογή. Τέλος καλεί τη συνάρτηση loadCatalog για να ανοίξει τον κατάλογο της συλλογής.

void **closeCollection(),**

Η συνάρτηση closeCollection καλείται για να κλείσει την συλλογή που είναι ανοιχτή. Καλεί την closeCatalog για να κλείσει τον κατάλογο και τέλος θέτει την μεταβλητή collectionPath ως ένα κενό String.

void saveCollections(),

Η συνάρτηση saveCollections αποθηκεύει την μεταβλητή Sylloges, που περιέχει τα ονόματα των συλλογών, στο αρχείο Sylloges.dat .

void **addToCollection**(String path),

Η συνάρτηση addToCollection εισάγει στην συλλογή που είναι ανοιχτή ένα ή πολλά documents. Το όρισμα path είναι το path από το οποίο θα πάρει τα documents. Ελέγχει αν πρόκειται για ένα document ή φάκελο και καλεί την addToCollectionOneTxt ή την addToCollectionSeveralTxt αντίστοιχα. Τέλος αλλάζει την μεταβλητή newCatalog ώστε όταν κλείσει τη συλλογή να αποθηκευτεί ο νέος κατάλογος.

void **addToCollectionOneTxt**(String path),

Η συνάρτηση addToCollectionOneTxt αντιγράφει το document από το path που παίρνει ως όρισμα, στον φάκελο txt της συλλογής, ελέγχοντας αν το όνομα υπάρχει ήδη, και ενημερώνει τον κατάλογο καλώντας τη συνάρτηση InsertIntoCatalog.

void **addToCollectionSeveralTxt(**String path),

Η συνάρτηση addToCollectionSeveralTxt αντιγράφει ένα ένα τα document από το path που παίρνει ως όρισμα, στον φάκελο txt της συλλογής, ελέγχοντας αν τα ονόματα υπάρχουν ήδη, και ταυτόχρονα ενημερώνει τον κατάλογο καλώντας τη συνάρτηση InsertIntoCatalog.

void **deleteTxt**(String txtName),

Η συνάρτηση deleteTxt παίρνει ως όρισμα το όνομα του document προς διαγραφή. Καλεί την συνάρτηση deleteFromCatalog για την ενημέρωση του καταλόγου και τέλος διαγράφει το document από τη συλλογή και αλλάζει την μεταβλητή newCatalog ώστε όταν κλείσει τη συλλογή να αποθηκευτεί ο νέος κατάλογος.

String **removespaces**(String line),

Η συνάρτηση removespaces χρησιμοποιείται για το parse των document med_docs και cran_docs.

Φορμάρει την line ώστε αν μία λέξη έχε κοπεί και περιέχει – στο τέλος της γραμμής, να μην την πάρει ως δύο διαφορετικές λέξεις. Η ανάγκη χρήσης μιας τέτοιας συνάρτησης προέρχεται απο το γεγονός οτι τα txt που δίνονται διατρέχονται σειρα προς σειρα αρχικά και στο τελός συνολικά ως εγραφο. Οπότε λέξεις που βρίσκονται στο τέλος ή ειναι μισές (correla- (αλλαγή γραμμης) tion) σπανε είτε λόγω της παυλας είτε λόγω κενου που έτυχε να υπαρχει..

3)Catalog.java

Η κλάση Catalog υλοποιεί τον αντεστραμένο κατάλογο. Κάθε συλλογή έχει έναν κατάλογο που είναι αποθηκευμένος στον φάκελο catalog ως catalog.dat . Ο κατάλογος είναι ένα TreeMap με keys ένα String που είναι η λέξη και values ένα ArrayList από στιγμιότυπα της κλάσης Pair με id το όνομα του document και counter τον αριθμό εμφανίσεων της λέξης στο document.

Η κλάση Καταλογ περιέχει τις εξής συναρτήσεις:

TreeMap **createCatalog**(TreeMap<String, ArrayList<Pair>> catalog, String path),

Η συνάρτηση createCatalog παίρνει ένα TreeMap ως όρισμα το οποίο είναι ο προσωρινός κατάλογος και ένα path που είναι το document που πρέπει να μπει στον κατάλογο. Αποθηκεύει το document σε ένα String και κάνει τις κατάλληλες αλλαγές ώστε να μην περιέχει σημεία στίξης και σύμβολα που δεν χρειάζονται στον κατάλογο. Στη συνέχεια παίρνει μία μία τις λέξεις του String και είτε την εισάγει στον κατάλογο, αν δεν υπάρχει, είτε ενημερώνει την ArrayList της ώστε να περιέχει και το document που εξετάζεται. Τέλος επιστρέφει τον ανανεωμένο κατάλογο.

void **closeCatalog(),**

Η συνάρτηση closeCatalog ελέγχει την μεταβλητή newCatalog για να δει αν έχει αλλάξει ο κατάλογος. Αν όντως έχει γίνει αλλαγή τοτε καλεί την saveCatalog. Τέλος θέτει την μεταβλητή collectionCatalog ίση με null.

void **saveCatalog(),**

Η συνάρτηση saveCatalog αποθηκεύει τον κατάλογο της συλλογής που είναι ανοιχτή στο αρχείο catalog.dat του φακέλου catalog. Αν δεν υπάρχει ο φάκελος τον δημιουργεί.

Επίσης όταν η συνάρτηση καλείται με ένα String όρισμα αποθηκεύει τον κατάλογο στην συλλογή που δίνεται από το όρισμα.

void **loadCatalog(),**

Η συνάρτηση loadCatalog φορτώνει τον κατάλογο της συλλογής που είναι ανοιχτή στην μεταβλητή collectionCatalog.

void **InsertIntoCatalog**(String path),

Η συνάρτηση InsertIntoCatalog είναι παρόμοια με την createCatalog όμως αντί να αποθηκεύει τον κατάλογο προσωρινά, χρησιμοποιεί τον κατάλογο της συλλογής που είναι ανοιχτή, από την μεταβλητή collectionCatalog και τον ενημερώνει για κάθε λέξη.

void **deleteFromCatalog**(String path),

Η συνάρτηση deleteFromCatalog παίρνει το path του document που θα διαγραφεί, το κρατάει σε ένα String και μετά τις κατάλληλες αλλαγές, για κάθε λέξη του String αφαιρεί από τον κατάλογο το στιγμιότυπο της κλάσης pair που έχει id το όνομα του document.

void **removeCommon**(int percentage),

Η συνάρτηση removeCommon παίρνει ως όρισμα έναν αριθμό από το 0 εώς το 100, το οποίο είναι το ποσοστό βάση του οποίου θα αφαιρεθούν λέξεις από τον κατάλογο. Στη συνέχεια ελέγχει κάθε λέξη του καταλόγου και αν το ποσοστό εμφάνισής της είναι μεγαλύτερο από το ποσοστό.

4)Pair.java

Η κλάση αυτή υλοποιέι τους κομβους των λιστών του καταλόγου.Τα πεδία της είναι:

private String id (το ιd ενος εγραφου)

private int counter (ποσες φορες εμφανίζετε η λέξη στην οποία αντιστοιχει η λίστα στο εργαφο)

5)Boolean.java

Η κλαση Boolean υλοποιή την αναζητηση στον κατάλογο συμφωνα με το Boolean μοντέλο.

Η διαδικασία έχει ως εξής:

Αρχικά υποβάλεται το query. Στο query είναι απαραίτητη η χρηση παρενθέσεων στις λογικές εκφρασεις οταν αυτες ξεπερνούν τις 3 λεξεις.

Δηλαδή πρέπει να είναι πάντα της μορφής

word1 AND (word2 AND word3) ή

(word1 OR word2) AND (NOT word3) ή

word1 OR word2

παντα με χρηση κενών. Υποστιρίζεται οποιοδήποτε μήκος ερωτημάτων αρκεί πάντα να τηρούνται τα κενά και οι παρενθέσεις εμφώλευσης.

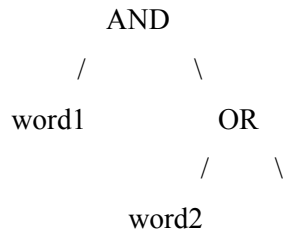
Αφού ληφθεί το ερώτημα γίνεται χρηση στοίβας ωστε να μετατραπεί σε μορφή κατάλληλη για την διευκόληνση δημιουργίας του λογικού δέντρου.

Δηλαδη ένα ερώτημα οπως το: word1 AND (word2 OR word3) θα παρει την μορφή:

AND (word1 OR(word2 word3)) .

Τέλος δημιουργείτε το λογικό δέντο σύμφωνα με το επεξεργασμενο ερώτημα. Το δέντρο αυτό αποτελείται απο κομβους OR, AND, NOT , VARIABLE. Καθε ενας απο αυτούς εχει 2 παιδια (το NOT εχει ενα παιδι, το 2ο κλαδι είναι NULL) εκτός απο τον VARIABLE ο οποίος είναι παντα στα φυλλα του δεντρου και περιεχει μία απο τις λέξης του ερωτήματος. Το δεντο δημιουργείται αναδρομικά πάντα συμφωνα με το ερώτημα.

Δηλαδή το δεντρο για ενα ερώτημα οπως αυτό του παραδείγματος θα ειναι το



Κάθε κομβος VARIABLE, δηλαδή κάθε φύλλο ελέγχει την υπαρξη της λέξης που περιεχει στον κατάλογο και επιστρέφει τα αποτελέσματα στον πατερα του(λιστα με τα εγραφα που περιεχουν την λεξη). Ο πατερας ανάλογα με το αν ειναι NOT, OR ή AND εκτελει πράξεις συμπλήρωσης, τομής ή ενωσης στις λιστες των παιδίων του. Το τελικό αποτέλεσμα το δίνει η ρίζα του δέντρου.

Οι συναρτήσεις τις κλάσης αυτής είναι:

ArrayList<String> **searchBooleanQuery**(String myQuery)

Η συνάρτηση αυτή ξεκινάει την διαδικασία που περιγράφτηκε παραπάνω και περιλαμβάνει τα τμήματα επεξεργασίας του query με στοιβα και την εντολή δημιουργίας του δέντρου. Επιστρέφει την λίστα με τα αποτελέσματα.

Node **createTree**(String query)

Η συνάρτηση αυτη δέχεται το επεξεργασμένο ερώτημα και δημιουργεί αναδρομικά το δέντρο. Μετα την δημιουργία επιστρέφει την ρίζα του δεντρου.

6) Node.java και VariableNode.java

πεδία του Node:

private Node lchild (το αριστερο παιδι)

private Node rchild (το δεξι παιδί)

private String type; (ο τύπος του κομβου, AND, NOT, OR)

private static ArrayList<String> totaldocs (το συνολο τον αρχειων, απαραίτητο για ερωτήματα NOT αφου οι κομβοι variableNode επιστρέφουν τα αρχεία που περιεχουν την λέξη ενώ θέλουμε το αντίθετο. Αρχικοποιείται κατα την δημιουργία του δεντρου απο τον καταλογο.)

πεδία του VariableNode:

private String var (η λέξη που αντιστοιχεί στον κομβο)

Οι κλασεις αυτες υλοποιούν τους κομβους του δέντρου που περιγραψάμε παραπάνω. Η VariableNode κληρονομεί απο την Node. Και οι δύο περιέχουν συναρτήσεις get και set για τα πεδία τους των οποίων ο σκοπος είναι προφανης. Περα απο αυτα διαθέτουν μια συνάρτηση ArrayList **evaluate**(TreeMap catalog) η οποία:

a) στα VariableNode επιστρέφει τα εγραφα που περιέχουν την λεξη var

b) στα Nodes καλει την evaluate στα παιδια, και εκτελει πραξεις στις λιστες που επιστρεφουν.

7) Vector.java

Η κλαση αυτη υλοποιέι το vector μοντέλο αναζήτησης. Υποστηρίζονται οι μέθοδοι Cosine, Jaccard και Dice. Περιέχει τρεις απλές συναρτήσεις

ArrayList<PairDouble> **initializeVector**(String method, String query)

Η συνάρτηση αυτή ξεκινάει την διαδικασία υπολογισμου των βαρων. Αρχικα σπάει το query σε λέξεις και ενεργοποιει αντιστοιχα τις συναρτησεις που υπολογιζουν τα βαρη για το ερώτημα και για τα εγραφα (δειτε

παρακάτω). Τέλος επιστρέφει μια ταξινομημένη λιστα με ζευγη < id, ομοιότητα>. Τα ζευγη υλοποιούνται απο την PairDouble

TreeMap **calculateQueryWeights**(String[] words)

Η συνάρτηση αυτη δέχεται τις λέξεις του ερωτήματος και υπολογίζει τα βάρη αυτών συμφωνα με την θεωρία διατρέχοντας τον κατάλογο.Επιστρέφει ενα TreeMap οπου κλειδί είναι η λέξη και value ειναι το βάρος αυτης.

double **calculateDocVectors**(String id, TreeMap queryMap, String method)

Καλείται για κάθε αρχείο υπολογίζει τα tf, idf και επιστρέφει το weight του το οποίο υπολογιστικε βαση του method.

8)PairDouble και CustomComparator

Οι κλάσεις αυτές υλοποιουν την ταξινομημένη λίστα που επιστρέφει η initialize vector.

Η PairDouble ειναι ενα ζευγός απο String , και double οπου το πρώτο είναι το id και το δευτερο το βάρος του.

Η CustomComparator υλοποιεί την Compare ωστε να γινετε δυνατή η ταξινόμηση της λιστας απο PairDouble.

Επεκτάσεις

Οι επεκτάσεις που υλοποιήθηκαν είναι το User InterFace, και η απαλοιφή συχνών λέξεων.

User InterFace

Αναλαμβάνει κάθε επικοινωνία με τον χρήστη. Αποτελείται απο ενα κυριο παράθυρο το οποίο επικοινωνει με τον χρηστη και του δινει την δυνατοτητα να διατρέχει όλες τις δυνατότητες του προγράμματος μέσω DialogWindows. Οι επιλογες του χρηστη ανοιγουν τα αντιστοιχα παράθυρα και απενεργοποιουν το κυριο παράθυρο. Παρεχονται δυνατότητες όπως μαζικη επιλογη διαγαφης εγράφων με κλικ σε αυτα, επισκοπιση συλλογης , browse αναμεσα σε συλλογες και δυνατότητα εκτέλεσης ερωτημάτων οταν αυτες επιλεχθούν καθώς και οποιαδήποτε άλλη λειτουργία ζητήθηκε και η εκτέλεση της οποίας μπορει να διευκοληνθει με InterFace.

Υλοποιείται απο την MainWindow.java

Απαλοιφή συχνών λέξεων

Δινεται η δυνατότητα στον χρήστη να επενέβει στον κατάλογο και να απαλείψει τις λέξεις που θεωρεί αυτος συχνες. Μέσω του InterFace μπορεί θέτοντας ενα ποσοστο να σβήσει τις λέξεις που το ξεπερναν σε εμφανισεις ανα εγραφο. Αν το ποσοστο ειναι π.χ. 60% οι λέξεις που εμφανιζονται σε εγραφα που το ξεπερνουν θεωρουνται συχνες και αφαιρουνται.

Τελος όπως είχε ζητηθεί είναι δυνατή η εκτελεση μαζικων ερωτηματων για τα αρχεια με μορφη οπως αυτη των med, cran και η επιστροφη αποτελεσματων για τα recall και precision. Τα ερωτηματα μπορουν να εκτελεστουν μονο για συλλογες που εχουν τον φακελο query (ο οποιος περιεχει τα αρχεια με τα queries και τα relevant) στα αρχεια συλλογων.

Λογω του σεβαστου χρονικού διαστήματος που απαιτειται για να τρέξει το προγραμμα 30 ερωτηματα (τάξης των 2-4 λεπτων) παραθέτουμε σε txt τα αποτελέσματα για την συλλογη med, με την μέθοδο συνημιτονου, για ολά και για τα top 30 αρχεια. Η συλλογη cran εχει ασυνέπεις στα αρχεια queries και relevant_docs μιας και κάποια id που έχει το δευτερο δεν υπάρχουν στο πρώτο με αποτέλεσμα να μην επιστρέφονται αποτελεσματα για αυτα. Οι μεσοι όροι για τα recall και precision βρίσκονται στο τέλος των αρχείων