Εργασία Solr  
Συστήματα Ανάκτησης Πληροφοριών

Οικονομικό Πανεπιστήμιο Αθηνών

Τμήμα Πληροφορικής

Κουτσοπούλου Αθανασία Μαρία

3140092

# Εισαγωγή

Από τις συλλογές που δινόντουσαν, επιλέχτηκε η τέταρτη, δηλαδή η Cranfield, με βάση το αποτέλεσμα του ΑΜ MOD 6, 3140092 MOD 6 = 4. Στην συλλογή αυτή υπάρχουν τα εξής αρχεία:

* cran.all.1400 – περιέχει όλα τα κείμενα της συλλογής που είναι συνολικά 1400. Το αρχείο έχει την παρακάτω μορφή:
  + .I – συμβολίζει ότι η γραμμή αυτή αναφέρει το id του κειμένου που ακολουθεί.
  + .Τ – συμβολίζει ότι οι γραμμές που ακολουθούν, έως το επόμενο .Symbol που θα συναντήσουμε, αφορούν τον τίτλο του κειμένου.
  + .Α – συμβολίζει ότι οι γραμμές που ακολουθούν, έως το επόμενο .Symbol που θα συναντήσουμε, αφορούν τον συγγραφέα του κειμένου.
  + .Β – συμβολίζει ότι οι γραμμές που ακολουθούν, έως το επόμενο .Symbol που θα συναντήσουμε, αφορούν πληροφορίες σχετικά με την βιβλιογραφία του κειμένου.
  + .W – συμβολίζει ότι οι γραμμές που ακολουθούν, έως το επόμενο .Symbol που θα συναντήσουμε, αφορούν περιεχόμενο του ίδιου του κειμένου.

Ωστόσο, να σημειωθεί ότι σε ορισμένα κείμενα υπήρχε λανθασμένη μορφοποίηση, όπως στο κείμενο με id 240, όπου μετά από όλα τα άνω .Symbol που αναφέρουμε, υπήρχε ξανά το .Α και το .Β, και η γραμμές που τα ακολουθούσαν δεν είναι δυνατόν να αφορούν πράγματι τον συγγραφέα και την βιβλιογραφία, ή όπως στο κείμενο με id 576 υπήρχε δύο φορές το .W. Σε αυτά αλλά και σε άλλες δύο περιπτώσεις που εμφανίζονταν τέτοιου είδους λάθη στην μορφοποίηση, τα επαναλαμβανόμενα .Symbols και οι γραμμές που τα ακολουθούσαν αγνοήθηκαν.

* cran.qry – περιέχει όλα τα queries που θέλουμε να εκτελέσουμε πάνω στην συλλογή. Το αρχείο έχει την παρακάτω μορφή:
  + .I – συμβολίζει ότι η γραμμή αυτή αναφέρει το id του κειμένου που ακολουθεί.
  + .W – συμβολίζει ότι οι γραμμές που ακολουθούν, έως το επόμενο .Symbol που θα συναντήσουμε, αποτελούν ίδιο το query.

Ωστόσο, να σημειωθεί ότι οι απαρίθμηση των queries ήταν λανθασμένη. Ενώ τα queries είναι συνολικά 225, τα ids έπαιρναν τυχαίες τιμές από 001 – 365. Έτσι λοιπόν κατά την μετατροπή τους στην κατάλληλη μορφή για να εκτελεστεί το query μέσω του solr, χρησιμοποιήθηκε counter και δόθηκαν τα ids από 1 – 225 στην σειρά.

* cranqrel – περιέχει όλα τα σχετικά κείμενα ανά query. Το αρχείο έχει την παρακάτω μορφή:
  + QueryId RelativeDocId RelevanceNumber

Να σημειωθεί ότι προκειμένου να γίνει χρήση του trec\_eval έγινε επεξεργασία του αρχείου ώστε να έρθει στην επιθυμητή μορφή, δηλαδή:

* + QueryId Iter RelativeDocId RelevanceNumber

Όπου το iter είναι ένα string που αγνοείται, στην περίπτωσή μας είναι το μηδέν.

* Readme – περιέχει πληροφορίες σχετικά με την μορφή του cranrel, κυρίως για τη σημασία των διαφορετικών τιμών του relevanceNumber.

Το java project που δημιουργήθηκε είναι υπεύθυνο για όλα τα παρακάτω.

# Μετατροπή Cran.all.1400 σε newDocs.xml

Αρχικά, έπρεπε να γίνει μετατροπή της συλλογής κειμένων μας, δηλαδή του cran.all.1400 αρχείου, σε αρχείο xml αρχείο με την κατάλληλη μορφή ώστε να είναι δυνατόν να φορτωθούν τα κείμενα στο Solr. Η μορφή του xml είναι η εξής:

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<add>

<doc>

<field name="identifier"> </field>

<field name="title"> </field>

<field name="author"> </field>

<field name="bAttr"> </field>

<field name="text"> </field>

</doc>

</add>

H μετατροπή γίνεται μέσα από την main της κλάσης *ModifyCollection*. Αυτή καλεί την μέθοδο *parseFile*, όπου και τελικά υλοποιεί την μετατροπή. Λαμβάνει ως είσοδο το όνομα του αρχείου, δηλαδή το cran.all.1400. Για να χρησιμοποιηθεί το αρχείο στο Solr προκειμένου να φορτωθούν τα κείμενα, δημιουργούμε αρχικά ως root element το “**add**” και σε αυτό προσθέτουμε νέο παιδί element “**doc**” για κάθε νέο κείμενο που συναντάμε στο αρχείο εισόδου. Σε κάθε “doc”, έχουμε πέντε παιδιά elements “**field**” με το name attribute να παίρνει τις τιμές που φαίνονται στην παραπάνω εικόνα. Και τέλος σε κάθε “field” element, έχουμε ως παιδί ένα **text node** με το αντίστοιχο περιεχόμενο που βρέθηκε στο αρχείο εισόδου.

Ας προσπαθήσουμε να εξηγήσουμε λίγο παραπάνω την λογική του κώδικα.   
Δημιουργούμε το “add” root element και στην συνέχεια προσθέτουμε σε αυτό ως παιδί ένα νέο “doc” element όταν έχουμε ολοκληρώσει όλες τις προσθήκες στο “doc” αυτό. Συγκεκριμένα οι πληροφορίες για τα text nodes λαμβάνονται με την παρακάτω λογική.   
Καταρχάς έχουμε 4 StringBuilders που αντιστοιχούν στο περιεχόμενο του “title”, “author”, “bAttr”, και τέλος “text”, δηλαδή των .T, .A, .B, .W αντίστοιχα.   
Ακόμα έχουμε 4 Boolean μεταβλητές που χρησιμοποιούμε ως flags ώστε όταν διαβάσουμε γραμμή που δεν ξεκινά με κάποιο από τα .Symbol, να ξέρουμε σε πιο StringBuilder πρέπει να κάνουμε append το περιεχόμενο της γραμμής αυτής. Τα flags αυτά παίρνουν τις ανάλογες τιμές κάθε φορά που η γραμμή που διαβάστηκε ξεκινά με κάποιο .Symbol.  
Έχουμε λοιπόν ένα Switch Case Statement, που κάθε φορά που διαβάζουμε νέα γραμμή, ελέγχουμε αν ξεκινά με κάποιο .S, και αν ναι εκτελούνται οι αντίστοιχες εντολές του εκάστοτε case που κυρίως δίνουν τιμές σε διάφορα flags που μας χρειάζονται στην συνέχεια προκυμμένου να αποφανθούμε για το τι πρέπει να συμβεί, διαφορετικά εκτελούνται οι εντολές του default, όπου εκεί έχουμε if-else if έτσι ώστε να ανανεώσουμε το σωστό StringBuilders.   
Ιδιαίτερη περίπτωση είναι το case .I όπου, εκτός από όσα αναφέρθηκαν για όλα τα case, εδώ – εάν δεν είναι το πρώτο κείμενο – δημιουργούμε τα field elements και τα αντίστοιχα text nodes τους για το προηγούμενο κείμενο. Δηλαδή, αν διαβάσουμε νέο .Ι, πριν κάνουμε οτιδήποτε άλλο, θα πάμε να δημιουργήσουμε τα fields και όλες τις σχετικές πληροφορίες για το προηγούμενο κείμενο, συνεπώς για το τελευταίο μέχρι στιγμής doc element, χρησιμοποιώντας τα 4 StringBuilders που αναφέραμε νωρίτερα.   
Το ίδιο κάνουμε και όταν φτάσουμε στο τέλος του αρχείου εισόδου, καθώς το τελευταίο doc element, δεν έχει λάβει ακόμη τα fields του, αφού δεν ακολούθησε κάποιο .I στο αρχείο εισόδου.

# Solr – Core Creation and Load Files

Αρχικά μέσα από το bin directory ξεκινάμε το Solr, με την εντολή:

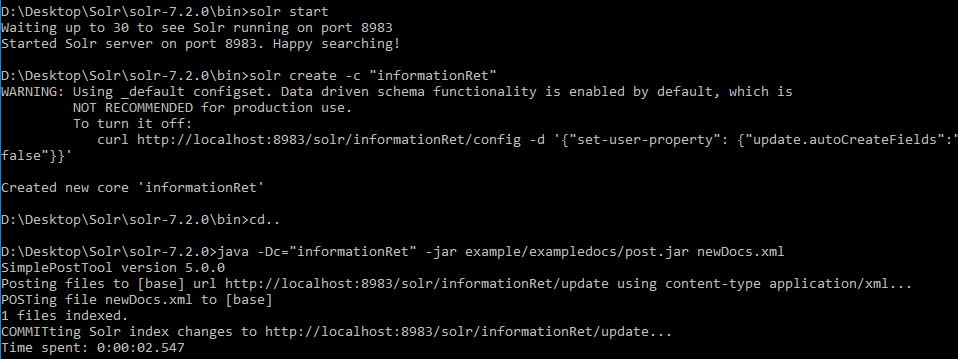
**solr start**

Στην συνέχεια δημιουργούμε νέο core με την εντολή:

**solr create -c “informationRet”**   
, μέσα στο οποίο θα φορτώσουμε τα documents της συλλογής και στην συνέχεια θα εκτελέσουμε τα queries.

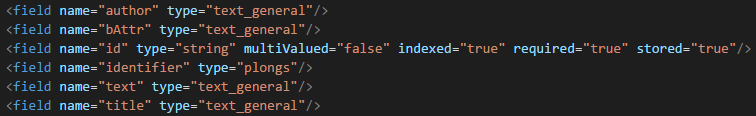
Τέλος, πλέον από το main solr directory, φορτώνουμε τα documents της συλλογής από το *newDocs.xml* αρχείο που δημιουργήσαμε στο προηγούμενο βήμα (Μετατροπή cran.all.1400 σε .xml) στο core που μόλις δημιουργήσαμε, μέσω του post.jar τους solr, τρέχοντας την εντολή:

**java -Dc=”informationRet” -jar example/exampledocs/post.jar newDocs.xml**



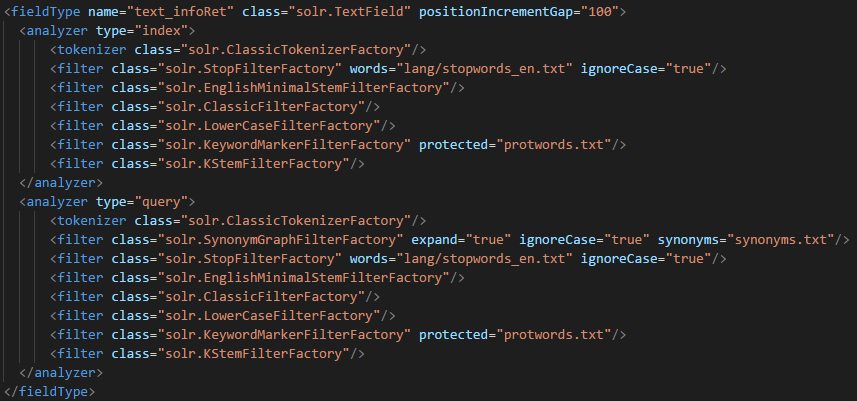
# Solr – Modify Managed Schema

Προκειμένου να έχουμε ένα αποδοτικότερο indexing, και να χρησιμοποιήσουμε διάφορες τεχνικές τύπου stemming, stopwords κτλ, τροποποιήσαμε το Managed Schema που δημιούργησε το Solr, το οποίο βρίσκεται στο directory: **solr-7.2.0\server\solr\ informationRet \conf**. Αυτό είχε τα εξής πεδία:



Όμως το text\_general, δεν υλοποιεί καμία από της ζητούμενες τεχνικές. Έτσι λοιπόν, πρώτη μας σκέψη ήταν να χρησιμοποιήσουμε κάποιο από τα υπάρχοντα fieldType πεδία ως type στα πεδία των κειμένων μας, εκτός του identifier, δηλαδή τα title, author, bAttr και text, έτσι ώστε να πετύχουμε το ζητούμενο. Συγκεκριμένα, σκεφτήκαμε να χρησιμοποιήσουμε το text\_en όμως προκειμένου να έχουμε μεγαλύτερο έλεγχο στο ποιος tokenizer αλλά και ποια filtra θα χρησιμοποιηθούν αποφασίσαμε τελικά να δημιουργήσουμε ένα εξ ολοκλήρου νέο fieldType.

Το νέο fieldType ονομάστηκε **text\_infoRet** και το θέσαμε ως type στο title, author, bAttr και text. Το οποίο φαίνεται εδώ:



Σε αυτό χρησιμοποιήσαμε τα εξής:

* **ClassicTokenizerFactory –** ως tokenizer επιλέχτηκε ο συγκεκριμένος όπου κάνει tokenize όπως ακριβώς ο StandardTokenizer με delimeters το space και τα σημεία στίξης, όμως με εξαιρέσεις τα παρακάτω:
  + Τελείες όπου δεν ακολουθούνται από κενό κρατούνται ως μέρος του token, πχ example.keep
  + Παύλα που συνδέει αριθμούς ή λέξεις που περιέχουν αριθμούς κρατείται χωρίς να σπάει, πχ goto-page7
  + Αναγνωρίζει internet domains και emails
* **StopFilterFactory** με words="lang/stopwords\_en.txt" ignoreCase="true" – όπου εφαρμόζει την τεχτνική των stopwords, δηλαδή αγνοεί αυτές τις λέξεις που βρίσκονται στο stopwords.txt. Να σημειωθεί εδώ, πως έχουμε ανανεώσει το αρχείο αυτό χειροκίνητα.
* **EnglishMinimalStemFilterFactory –** filter όπου μετατρέπει όποια λέξη βρίσκεται στον πληθυντικό στην αντίστοιχη λέξη στον ενικό.
* **ClassicFilterFactory –** όπου αφαιρεί τις τελείες από τα ακρόνυμα και το κτητικό s πχ Ο.Π.Α. -> ΟΠΑ και cat’s -> cat
* **LowerCaseFilterFactory –** όπου μετατρέπει όλα τα κεφαλαία γράμματα σε μικρα
* **KeywordMarkerFilterFactory** protected="protwords.txt" **–** όπου θέτει ως «προστατευμένες» τις λέξεις που βρίσκονται στο protwords.txt και έτσι δεν υπόκεινται σε stemming
* **KStemFilterFactory –** όπου εφαρμόζει stemming στα tokens, με λιγότερο «επιθετικό» τρόπο από αυτόν του PorterStemFilter
* **SynonymGraphFilterFactory –** το οποίο χρησιμοποιείται μόνο στα queries, και χρησιμοποιεί τις λέξεις που υπάρχουν στο synonyms.txt ώστε να αντικαταστεί όσες υπάρχουν στο query με κάθε συνώνυμο της.

Έτσι λοιπόν έχουμε πλέον το modified managed schema.

# Solr Reindex Docs

Προκειμένου να γίνει reindex στα documents, πρέπει αρχικά να κάνουμε restart τον server. Επομένως μέσα στο bin directory τρέχουμε τις παρακάτω εντολές:

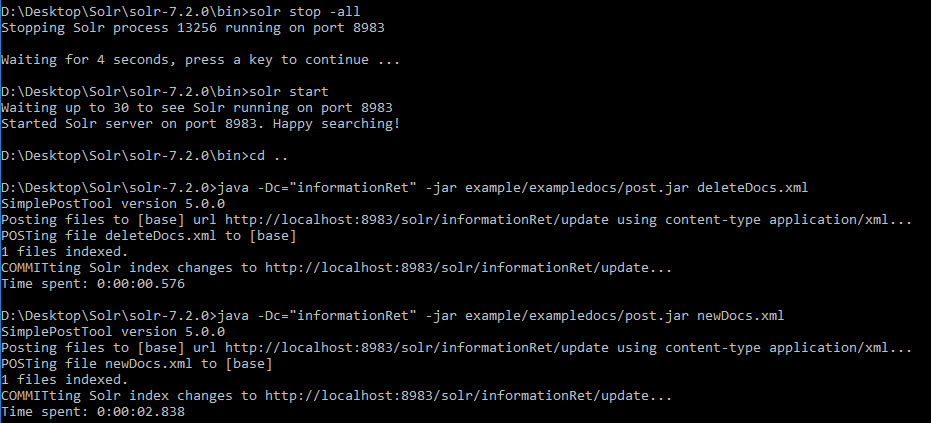
**solr stop -all**

**solr start**

Τέλος, πρέπει να σβήσουμε τα documents από το core και να τα ξαναφορτώσουμε, ώστε να γίνουν reindex με βάση το modified managed schema. Προκειμένου να διαγράψουμε τα documents χρειαζόμασταν ένα νέο αρχείο όπου και δημιουργήσαμε, το *deleteDocs.xml* και μέσα περιέχει τo εξής: *<delete><query>\*:\*</query></delete>* . Έτσι μέσα στον main solr directory και χρησιμοποιώντας ξανά το post.jar του solr, τρέχουμε τις παρακάτω εντολές:

**java -Dc=”informationRet” -jar example/exampledocs/post.jar deleteDocs.xml**

**java -Dc=”informationRet” -jar example/exampledocs/post.jar newDocs.xml**



# Μετατροπή cranqrel

Προκειμένου να τρέξουμε το trec\_eval και να πάρουμε τα στατιστικά αποτελέσματα για τα documents που γίνονται retrieve μέσω του Solr, πρέπει να μετατρέψουμε το cranqrel όπως αναφέραμε στην εισαγωγή. Επομένως, έχουμε δημιουργήσει την μέθοδο *modifyQrelFIle* στην κλάση *ModifyCollection*, όπου στην ουσία διαβάζουμε όλο το αρχείο, και στην συνέχεια ξαναγράφουμε από την αρχή στο αρχείο από την αρχή σβήνοντας όλα τα προηγούμενα. Όταν το ξαναγράφουμε, παίρνουμε γραμμή γραμμή ότι διαβάσαμε, κάνουμε trim, και μετά αντικαθιστούμε το πρώτο “space” που συναντάμε με “space 0 space”.

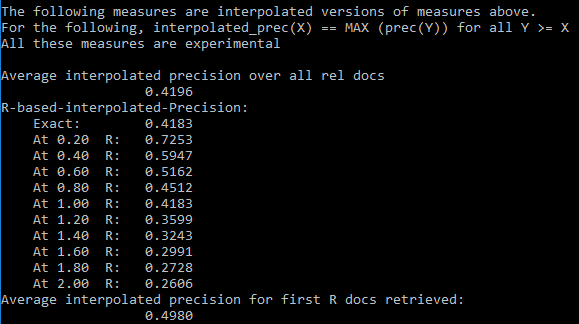
# Queries Execution

Να σημειωθεί πως έχουμε δημιουργήσει μια βοηθητική κλάση Query, με πεδία το id και το query string. Έτσι αρχικά μέσω της μεθόδου *readAndTransformQueries*, στην κλάση *ExecuteQueries*, διαβάζουμε όλα τα queries από το cran.qry και δημιουργούμε νέο Query object και το προσθέτουμε στο *queries Arraylist* της κλάσης. To αρχείο όπως αναφέραμε στην εισαγωγή έχει συγκεκριμένη μορφή, επομένως, κάθε γραμμή που διαβάζουμε ελέγχουμε αν ξεκινά με .I, τότε αν δεν είναι το πρώτο query, τότε προσθέτουμε το query string που διαβάσαμε μέχρι τώρα στο προηγούμενο Query object, και δημιουργούμε και το νέο Query object μόνο με id μέχρι στιγμής. Όποτε δεν ξεκινά με .Ι, κάνουμε append στο StringBuilder που χρησιμοποιούμε για να κατασκευάσουμε το query string. (Όμοια λογική με το διάβασμα του cran.all.1400.)  
Αφού λοιπόν έχουμε πλέον το *queries Arraylist* με όλα τα queries, τότε μέσω της *runQueries* της κλάσης *ExecuteQueries,* τρέχουμε ένα ένα τα queries. Συγκεκριμένα, αρχικά δημιουργούμε ένα νέο solr client και για κάθε ένα query μέσα στο Arraylist, πρώτα κάνουμε tokenize το query string με το κενό, και δημιουργούμε ένα stringbuilder ως εξής: “text:token OR text:token2 …”. Έπειτα δημιουργούμε ένα HashMap με τα query parameters, όπως το q, rows και fl, το οποίο το περνάμε στο MapSolrParams. Στη συνέχεια, παίρνουμε το response από τον client ως ένα SolrDocumentList, και για κάθε ένα document από αυτά γράφουμε στο νέο μας txt αρχείο με τα αποτελέσματα, μια εγγραφή της μορφής: qid 0 docId 1 score STANDARD. To όνομα του αρχείου μας είναι το queryResults.

# Trec Eval

Για την αξιολόγηση των αποτελεσμάτων μας, τρέχουμε την εντολή:

**trec\_eval -q -a cranqrel queryResults**



Στο rar της εργασία έχουμε προσθέσει και το informationRetTrecEval.txt όπου περιέχει τα αποτελέσματα τις εντολής συνολικά.