ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

ΗΥ 463 – Συστήματα Ανάκτησης Πληροφοριών

2008 - 2009

Εργασία: QueryEvaluator

Εισαγωγή

Σκοπός της εργασίας αυτής είναι η επέκταση του **υποσυστήματος αποτίμησης επερωτήσεων** (QueryEvaluator) της μηχανής αναζήτησης Μίτος. Μελέτες έχουν δείξει ότι το 58% των χρηστών παρατηρούν τα αποτελέσματα της πρώτης σελίδας αποτελεσμάτων ενώ μόνο το 19% συνεχίζει και στη δεύτερη. Εκ τούτου η διαβάθμιση των στοιχείων της απάντησης είναι εξαιρετικά σημαντική και για το λόγο αυτό οι μηχανές αναζήτησης διαβαθμίζουν τα στοιχεία της απάντησης λαμβάνοντας υπόψη και τη *συνάφεια* τους με την επερώτηση αλλά και την *εγκυρότητά* τους όπως αυτή μπορεί να εκτιμηθεί από τις τεχνικές ανάλυσης συνδέσμων (Link Analysis Techniques) που έχουν προκύψει τα τελευταία χρόνια. Αυτή τη στιγμή, η μηχανή αναζήτησης Mitos, υποστηρίζει τα εξής μοντέλα διαβάθμισης: VSM (Vector Space Model), Boolean, Extended Boolean και Fuzzy Model. Επίσης υποστηρίζει τη διαβάθμιση PageRank.

Σκοπός της εργασίας είναι η δημιουργία δύο νέων μεθόδων διαβάθμισης: α) **Okapi-BM25** και β) **Hybrid Ranking**. Τα μοντέλα αυτά πρέπει να ακολουθούν τη σχεδίαση των ήδη διαθέσιμων μεθόδων διαβάθμισης της μηχανής, και θα πρέπει να υλοποιηθούν με όσο γίνεται πιο αποδοτικό τρόπο. Τέλος και για να δούμε ποιο μοντέλο είναι πιο αποτελεσματικό, θα πρέπει να γίνει αξιολόγηση όλων των μεθόδων διαβάθμισης που υποστηρίζει η μηχανή αναζήτησης Mitos, χρησιμοποιώντας σαν σημείο αναφοράς μία από τις συλλογές αξιολόγησης του TREC.

Α) Υλοποίηση του Okapi BM25

Okapi BM25 ονομάζεται μία συνάρτηση βαθμολόγησης συνάφειας. Βασίζεται στο πιθανοκρατικό μοντέλο ανάκτησης και υπάρχουν διάφορες παραλλαγές της. Μία από τις επικρατέστερες είναι η παρακάτω:

Δοθείσας επερώτησης Q, που περιλαμβάνει τους όρους $q_1,...,q_n$, η βαθμολογία με βάση το BM25 για κάποιο κείμενο *D* είναι

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\text{avgdl}}\right)},$$

όπου $f(q_i,D)$ είναι το tf (term frequency) του όρου q_i στο κείμενο *D*, $|D|$ είναι ο αριθμός των εμφανίσεων των λέξεων στο κείμενο *D*, και *avgdl* είναι το μέσο μήκος ενός κειμένου στη συλλογή μας. Τα k_1 και *b* είναι ελεύθερες μεταβλητές (συνήθως k_1 = 2.0 και *b* = 0.75). IDF(q_i) είναι το Inverse Document Frequency του όρου q_i, το οποίο συνήθως υπολογίζεται ως:

$$\text{IDF}(q_i) = \log\frac{N-n(q_i)+0.5}{n(q_i)+0.5},$$

όπου *N* είναι ο συνολικός αριθμός κειμένων της συλλογής και $n(q_i)$ είναι ο αριθμός των κειμένων που περιλαμβάνουν τον όρο q_i.

Για την υλοποίηση του αλγορίθμου θα πρέπει να επεκταθεί (extend) η **`abstract class QueryEvaluator`**, που υπάρχει στον κατάλογο **`WEB-INF/project/src/mitos/queryEvaluator/evaluators`** του κώδικα της μηχανής, με μία κλάση με όνομα **`EvaluatorOkapiBM25`** που θα ανήκει στο package **`mitos.queryEvaluator.evaluators`**. Πιο συγκεκριμένα θα πρέπει να υλοποιήσετε τη μέθοδο:

```
/**
 * Computes the similarity degree of each document to the given query taking into account
 * both its pageRank
 * score and its the similarity degree based on a specific retrieval model.
 *
 * @return a sorted map, which maps each file to to its similarity to the query.
 * Map.Entry: (Float ranking, ArrayList (with Integers)).
*/
 abstract public SortedMap <Float, ArrayList<Integer>> evaluate(String query)
                                                                    throws Exception;
```

Για τον υπολογισμό της τελικής βαθμολόγησης, μπορείτε να χρησιμοποιήσετε την

```
protected void setPageRankerResults(TreeMap <Integer, Float > ranks) throws Exception
```

της **`QueryEvaluator`**, η οποία βαθμολογεί με 70% την τιμή του εκάστοτε αλγόριθμου αποτίμησης και 30% την τιμή του pageRank. Η μέθοδος αυτή παίρνει σαν όρισμα ένα TreeMap με όλα τα συναφή FileIds και τα εκάστοτε pageRanks, και μεταβάλλει το

```
protected SortedMap < Float, ArrayList <Integer> > results
```

της QueryEvaluator, το οποίο κρατά τα τελικά rankings και το αντίστοιχο ArrayList με τα FileIds. και επιστρέφεται από την abstract μέθοδο evaluate. Για την υλοποίηση του EvaluatorOkapiBM25, θα πρέπει να χρησιμοποιήσετε κατάλληλες μεθόδους του mitos.indexer.Index, ώστε να πάρετε τις απαιτούμενες πληροφορίες για τα κείμενα της συλλογής και τους όρους που περιέχουν. Μπορείτε να κοιτάξετε τους ήδη υπάρχοντες evaluators για να δείτε ενδεικτικές χρήσεις του Index. Θυμηθείτε ότι θα πρέπει να γίνει μία αποδοτική υλοποίηση της μεθόδου.

Β) Υλοποίηση ενός Hybrid Retrieval Model

Το Hybrid ranking model είναι ένα υβριδικό μοντέλο διαβάθμισης (δικής μας έμπνευσης). Ονομάζεται υβριδικό διότι συνδυάζει ιδέες από το Boolean Model, ενός Best-match μοντέλου όπως είναι το VSM ή το Okapi και ενός μοντέλου διαβάθμισης βασισμένου σε ανάλυση συνδέσμων. Πιο συγκεκριμένα, ας θεωρήσουμε ότι η επερώτηση του χρήστη q, αποτελείται από |q| όρους. Τότε η απάντηση του Hybrid ranking model (answer(q)), θα είναι μια γραμμική διάταξη από blocks, δηλαδή answer(q) = $<B_{|q|}, B_{|q|-1}, \ldots, B_1>$, όπου το B_i αποτελείται από όλες εκείνες τις σελίδες που περιέχουν i όρους της επερώτησης ($1 \leq i \leq |q|$). Τα στοιχεία κάθε block B_i, διατάσσονται με βάση το ranking του χρησιμοποιούμενου μοντέλου διαβάθμισης.

Καλείστε να δημιουργήσετε μία νέα κλάση EvaluatorHybrid, αντίστοιχη με την EvaluatorOkapiBM25 που περιγράφηκε παραπάνω. Η EvaluatorHybrid θα πρέπει να μπορεί να δουλεύει με το ranking που παρέχεται είτε από τον EvaluatorVector, είτε από τον EvaluatorOkapiBM25 που θα έχετε υλοποιήσει (μπορείτε να θεωρήσετε πως από default θα χρησιμοποιεί το VSM μοντέλο). Για αυτό το λόγο θα παρέχεται και μία μέθοδος για να μπορεί να γίνει η επιλογή του επιθυμητού μοντέλου διαβάθμισης (**public void setBlockRankingModel(QueryEvaluator)**). Η επιθυμητή διάταξη των σελίδων της απάντησης μπορεί να προκύψει από την εξής έκφραση βαθμολόγησης:

$$\mathbf{hybridRank(d_i) = (simRank(d_i) + | qB |) / (| q | + 1)}$$

όπου $simRank(d_i)$ ο βαθμός συνάφειας των d_i με βάση το επιλεγμένο μοντέλο ανάκτησης, ο οποίος παίρνει τιμές από [0…1] και |qB| ο αριθμός των όρων της επερώτησης που εμφανίζεται στο συγκεκριμένο block. Θυμηθείτε ότι και πάλι πρέπει να βρείτε ένα αποδοτικό τρόπο για την υλοποίηση

του παραπάνω μοντέλου. Στη τελική γραπτή αναφορά δώστε ενδεικτικούς χρόνους αξιολόγησης.

Γ) Αξιολόγηση αποτελεσματικότητας με συλλογή TREC

Καλείστε να κάνετε μετρήσεις αποτελεσματικότητας των αλγορίθμων διαβάθμισης (VSM, Okapi-BM25. Hybrid-VSM, Hybrid-Okapi-BM25) χρησιμοποιώντας μία από τις συλλογές του TREC (Text Retrieval Conference). Θα σας δοθεί πρόσβαση σε μία βάση δεδομένων με όνομα **trec**, στην οποία θα έχει ευρετηριαστεί η συλλογή αυτή, έτσι ώστε να μπορεί να χρησιμοποιηθεί από τη μηχανή Mitos. Στη συνέχεια, θα πρέπει να εμπλουτίσετε τη κλάση **`TRECEvaluate`**, η οποία υπάρχει στο directory WEB-INF/project/src/mitos/testSuite/trec. Η κλάση αυτή φορτώνει τα queries του TREC, και αξιολογεί ένα μοντέλο ανάκτησης χρησιμοποιώντας τα μέτρα Precision/Recall. Εσείς θα πρέπει να υλοποιήσετε τις υπόλοιπες μετρικές (R-Precision, F-Measure, E-Measure, Fallout) που ήδη γνωρίζετε.

Δ) JUnit tests για τη σωστή λειτουργία των αλγορίθμων

Θα δοθούν αναλυτικές οδηγίες στο wiki.

Χρονοδιάγραμμα

Προτείνεται το εξής χρονοδιάγραμμα:

- 1η εβδομάδα (3-10 Απριλίου)
  - Εξοικείωση με τον υπάρχοντα κώδικα του Μίτου
- 2η εβδομάδα (27 Απριλίου-3 Μαΐου)
  - Υλοποίηση Hybrid Ranking
- 3η εβδομάδα (4 Μαΐου -10 Μαΐου)
  - Υλοποίηση Okapi BM25
- 4η εβδομάδα (11 Μαΐου -17 Μαΐου)
  - TREC evaluation, JUnits, πειράματα, δοκιμές και σύνταξη αναφοράς εργασίας (με μετρήσεις)

Παρατήρηση

Η καλύτερη υλοποίηση θα ενσωματωθεί στη μηχανή Μίτος και η ομάδα θα πάρει 5% bonus (στο βαθμό εργασίας).