Συστήματα Ανάκτησης Πληροφοριών
2η φάση προγραμματιστικής εργασίας

Ευαγγέλου Αθανάσιος
ΑΜ: 3130242

Προτού αναφερθώ στο τι υλοποιήθηκε για τη 2η φάση της εργασίας, κατά τη γνώμη μου, ήταν σωστό και πρέπον να διορθωθούν αρκετά από όσα υλοποιήθηκαν στη 1η φάση με σκοπό την ομαλότερη λειτουργία του όλου προγράμματος και καθώς οφείλω μια καλύτερη εξήγηση για το πρόγραμμα που υλοποίησα στη 1η φάση θα αναφερθώ πρώτα σε αυτά.

Αναφορικά με της διορθώσεις που έγιναν, πρωτίστως, η δημιουργία των indices γίνεται πλέον μέσα στο πρόγραμμα και όχι μέσω kibana. Αυτό επιτυγχάνεται με τη μέθοδο CreateIndex(indexName) :

```
private static void CreateIndex(String indexName) throws IOException, ParseException {
    GetIndexRequest request = new GetIndexRequest(indexName);
    boolean exists = restClient.indices().exists(request, RequestOptions.DEFAULT);
    if (exists) {
        System.out.println("Index already exists.");
    } else {
        System.out.println("Creating index " + indexName);
        CreateIndexRequest createIndexRequest = new CreateIndexRequest(indexName);
        JSONParser parser = new JSONParser();
        JSONObject json = (JSONObject) parser.parse(new FileReader( fileName: "Indexes/index.json"));
        createIndexRequest.source(json.toJSONString(), XContentType.JSON);
        CreateIndexResponse response = restClient.indices().create(createIndexRequest, RequestOptions.DEFAULT);
        System.out.println("Response:\n\tAcknowledged: " + response.isAcknowledged() + "\n\tShards Acknowledged: " + response.isShardsAcknowledged());
    }
}
```

Εδώ, πρώτα ελέγχεται αν υπάρχει ήδη index με το όνομα που δίνεται ως παράμετρος, αν υπάρχει τότε τυπώνεται σχετικό μήνυμα αν δεν υπάρχει τότε δημιουργείται το index. Τα indices δημιουργούνται βάση του template που βρίσκεται αποθηκευμένο στο index.json στο φάκελο Indices. Το αρχείο αυτό περιέχει τόσο τον analyzer όσο και τον αλγόριθμο TF-IDF:

```
{
  "settings": {
    "number_of_shards": 1,
    "similarity": {
      "scripted_tfidf": {
        "type": "scripted",
        "weight_script": {
          "source": "double idf = Math.log((field.docCount+1.0)/(term.docFreq+1.0)) + 1.0; return query.boost * idf;"
        },
        "script": {
          "source": "double tf = Math.sqrt(doc.freq); double norm = 1/Math.sqrt(doc.length); return weight * tf * norm;"
        }
      }
    },
    "analysis": {
      "analyzer": {
        "my_analyzer": {
          "type": "standard",
          "stopwords": "_english_"
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "text": {
        "type": "text",
        "similarity": "scripted_tfidf"
      }
    }
  }
}
```

Συστήματα Ανάκτησης Πληροφοριών
2η φάση προγραμματιστικής εργασίας

Ευαγγέλου Αθανάσιος
ΑΜ: 3130242

Όσο αφορά τον analyzer χρησιμοποιείται ο standard με φίλτρο που αγνοεί τα αγγλικά stop words, ενώ όσο αφορά τον αλγόριθμο TF-IDF χρησιμοποιείται scripted similarity στο πεδίο text όπου προστίθενται βάρη που προκύπτουν από τις συναρτήσεις :

double idf = Math.log((field.docCount+1.0)/(term.docFreq+1.0)) + 1.0; return query.boost * idf;

που αναλαμβάνει να υπολογίσει το Inverse Document Frequency (IDF) και,

double tf = Math.sqrt(doc.freq); double norm = 1/Math.sqrt(doc.length); return weight * tf * norm;

που υπολογίζει το Term Frequency (TF).

Σημαντικό, κατά τη γνώμη μου, είναι να αναφερθεί ότι κομμάτια του κώδικα της 1ης φάσης ξαναγράφτηκαν με σκοπό είτε να μπορούν να επαναχρησιμοποιηθούν και για ερωτήματα της 2ης φάσης είτε καθαρά για λόγους βελτιστοποίησης. Κυριότερο παράδειγμα αυτού είναι η μέθοδος Search(String parameter, String indexName) η οποία πλέον μπορεί να χρησιμοποιηθεί για να εφαρμόσει το query που περιέχει στο index που δίνεται ως παράμετρος.

Επίσης, η υλοποίηση μιας νέας κλάσης, Listing, καθώς και μιας λίστας από αντικείμενα αυτού του τύπου στην οποία αποθηκεύονται τα απαιτούμενα δεδομένα των αποτελεσμάτων των queries ώστε να μπορούν να χρησιμοποιηθούν στη συνέχεια για την αξιολόγηση.

Μια μικρού διόρθωση είναι η προσθήκη ενός ελέγχου για την περίπτωση που τα αρχεία που χρησιμοποιήθηκαν στη 1η φάση έχουν ήδη υποστεί την επεξεργασία που πρέπει, δηλαδή αν υπάρχουν αποθηκευμένα ως JSON στον ομώνυμο φάκελο να χρησιμοποιηθούν αυτά αντί να γίνει όλη η διαδικασία επεξεργασίας των xml άδικα.

Μερικές παραδοχές που αφορούν αυτό το κομμάτι, δεν γίνεται έλεγχος ένα-προς-ένα για τα αρχεία, αντ' αυτού γίνεται έλεγχος αν υπάρχουν ακριβώς τόσα .json αρχεία στον φάκελο JSON όσα είναι και τα .xml αρχεία στον φάκελο phase1_collection, δηλαδή στην περίπτωση που τα αρχεία στον φάκελο JSON είναι έστω κατά 1 περισσότερα ή λιγότερα τότε θα γίνει η επεξεργασία των xml, που προσθέτει επιπλέον κόστος. Θεωρήθηκε μη αναγκαία η περισσότερη έμφαση σε αυτό το μέρος του κώδικα καθώς δεν αποτελεί κύριο σκοπό της εργασίας, όμως υλοποιήθηκε σε αυτή τη μορφή ως μικρή βελτιστοποίηση.

Τέλος, διορθώθηκαν τυχών απροσεξίες ή και λάθη όπως το γεγονός ότι δεν αποθηκεύονταν σωστά τα αποτελέσματα του query και ότι το πρόγραμμα δεν αγνοούσε το πρώτο από τα αποτελέσματα που επέστρεφε το query, τώρα αυτά δουλεύουν όπως πρέπει, τα ανακτηθέντα δεδομένα αποθηκεύονται με τον προβλεπόμενο τρόπο ώστε να μπορούν να χρησιμοποιηθούν για το evaluation και τα queries επιστρέφουν τα 21 καλύτερα αποτελέσματα εκ των οποίων δεν λαμβάνεται υπόψιν. Επίσης, όλα τα αποτελέσματα των αξιολογήσεων βρίσκονται πλέον αποθηκευμένα στο φάκελο evaluations, συμπεριλαμβανομένων και αυτών της 1ης φάσης.

Συστήματα Ανάκτησης Πληροφοριών
$2^{η}$ φάση προγραμματιστικής εργασίας

Ευαγγέλου Αθανάσιος
ΑΜ: 3130242

Αναφορικά με τις υλοποιήσεις για τη $2^{η}$ φάση, τα νέα ερωτήματα που χρησιμοποιήθηκαν ήταν αυτά που αναφέρονται στον πίνακα της Ομάδας 1, αυτά βρίσκονται στο φάκελο new_queries με ονομασία αντίστοιχη του ερωτήματος που αυτά αναφέρονται.

Για την υλοποίηση των ερωτημάτων a και b χρησιμοποιήθηκε η μέθοδος Search(String parameter, String indexName), στην οποία χρησιμοποιείται ένα more-like-this query που παίρνει ως παράμετρο το κάθε ερώτημα με τη σειρά, ώστε να εκτελέσει τα queries. Τα ανακτηθέντα δεδομένα αποθηκεύονται, όπως προανέφερα, σε λίστα, και αμέσως μετά γράφονται σε αρχείο κειμένου.

Όσο αφορά το ερώτημα c, για τη περίπτωση των default values στο more-like-this query χρησιμοποιήθηκε και πάλι η ίδια συνάρτηση με τα προηγούμενα ερωτήματα, ενώ για την περίπτωση που προστέθηκαν συγκεκριμένες τιμές στις παραμέτρους του query υλοποιήθηκε μια νέα συνάρτηση, MLTSearch(String parameter, String indexName, String min_term_freq, String max_query_terms, String min_doc_freq, String max_doc_freq, String minimum_should_match), πανομοιότυπη της πρώτης όπου οι διάφοροι mlt παράμετροι δίνονται ως παράμετροι στη μέθοδο, για τα ερωτήματα αυτά χρησιμοποιήθηκαν οι παρακάτω τιμές:

- min_term_freq: αναφέρεται στον ελάχιστο αριθμό εμφανίσεων του όρου κάτω από την οποία παύει να λαμβάνεται υπόψιν από το σύστημα, άρα αν επιλέξουμε μικρότερο αριθμό λαμβάνουμε υπόψιν και σπανιότερους όρους, που έχει ως αποτέλεσμα την πιθανή βελτίωση των αποτελεσμάτων. Στα πειράματα επιλέχθηκαν τιμές 1, 5 και 10

- max_query_terms: είναι ο ελάχιστος αριθμός όρων του ερωτήματος που επιλέγονται, μεγαλύτερες τιμές της προκαθορισμένης βελτιώνουν την απόδοση, όμως το πληρώνουμε σε χρόνο εκτέλεσης. Στα πειράματα επιλέχθηκαν οι τιμές 27, 31 και 100. Όπως και στη προηγούμενη περίπτωση, έτσι και εδώ, ενώ η απόδοση βελτιώνεται αν μεγαλώσουμε τη τιμή δεν παύει να υπάρχει ένα όριο πέραν του οποίου η ακρίβεια παραμένει ίδια.

- min_doc_freq: αφορά τον ελάχιστο αριθμό αρχείων στα οποία εμφανίζεται κάποιος όρος, όπως και στη περίπτωση min_term_freq, για μικρότερες τιμές επιτρέπουμε και σπανιότερους όρους. Επιλέχθηκαν οι τιμές 6, 40, 3.

- max_doc_freq: Ο μέγιστος αριθμός αρχείων όπου εμφανίζεται κάποιος όρος, αντίθετα από τη προηγούμενη περίπτωση, εδώ, αυξάνοντας την τιμή της παραμέτρου όροι που εμφανίζονται σε πολλά διαφορετικά αρχεία δεν θα ληφθούν υπόψη. Οι τιμές που επιλέχθηκαν για τα πειράματα ήταν 100 , 50 και 1,000 αντίστοιχα.

- minimum_should_match: βάση αυτής της παραμέτρου καθορίζεται το ελάχιστο ποσοστό των όρων που πρέπει να ταυτίζονται, επιλέχθηκαν οι τιμές 5%, 10%, 31% και παρατηρήθηκε σημαντική μείωση στο σύνολο των ανακτηθέντων αρχείων όσο αυξάνεται το ποσοστό ταύτισης.

Συστήματα Ανάκτησης Πληροφοριών
2η φάση προγραμματιστικής εργασίας

Ευαγγέλου Αθανάσιος
ΑΜ: 3130242

Τα αποτελέσματα των ερωτημάτων αποθηκεύονται ως εξής, με ονομασία phase_1_new_queries_results.txt αποθηκεύονται αυτά του a ερωτήματος, αυτά του ερωτήματος b ως phase_2_new_queries_results.txt, και του ερωτήματος c ως phase_2_default_mlt_results.txt, για το ερώτημα που αφορά more-like-this query με default values, και phase_2_mlt_results_test_1.txt, phase_2_mlt_results_test_2.txt και phase_2_mlt_results_test_3.txt για τα πειράματα με ειδικές παραμέτρους.