Συστήματα ανάκτησης πληροφοριών

Προγραμματιστική εργασία, φάση 4

Δημήτρης Μπάστας, 3130139
Φίλιππος Δουραχαλής, 3170045

Η εκπόνηση της εργασίας πραγματοποιήθηκε στο περιβάλλον Intellij IDEA.

Για τη συγκεκριμένη φάση αρχικά κατεβάζουμε το αρχείο pom.xml το οποίο περιέχει τα dependencies της Lucene και του DeepLearning4J που θα χρειαστούμε για κάνουμε build το project. Για τις ανάγκες της εργασίας χρησιμοποιήθηκαν οι κλάσεις που δόθηκαν στο μάθημα, στις οποίες έχουν προστεθεί οι κλάσεις από τις φάσεις 1 και 3, κατάλληλα τροποποιημένες ώστε να είναι δυνατή η προεπεξεργασία των κειμένων και των ερωτημάτων που θα δοθούν στον indexer και στη συνέχεια στο νευρωνικό δίκτυο.

Όπως και στις προηγούμενες φάσεις, στην main κλάση «**W2VRetrieval**» ορίζουμε τρεις σταθερές:

a) **TREC_EVAL_PATH**: Το directory όπου έχουμε το εργαλείο trec_eval μαζί με τα αρχεία qrels.text και results.text
b) **CACM_PATH**: Το directory όπου βρίσκεται η συλλογή μας
c) **INDEX_PATH**: Η διαδρομή προς το index

1ο σκέλος

Σε αυτό το κομμάτι δημιουργούμε ένα word2vec μοντέλο δίνοντάς του τα περιεχόμενα του custom πεδίου content που έχουμε δημιουργήσει από το ευρετήριο. Εναλλακτικά μπορούμε να το τροφοδοτήσουμε με το αρχείο της συλλογής απευθείας. Εκπαιδεύουμε το δίκτυο με τον αλγόριθμο Skip-Gram καθώς παρατηρήσαμε καλύτερα αποτελέσματα απ’ ότι με τη χρήση CBOW.

```java
public void testRankingWithTFIDFAveragedWordEmbeddings() throws Exception {
    int choice = 1; //determines whether the model will be created anew or if it will be loaded from txt file
    String collectionPath = CACM_PATH + "cacm.all";
    Path path = Paths.get(INDEX_PATH);
    Path path2 = Paths.get( first: ".\\index\\2");
    Directory directory = FSDirectory.open(path);
    Directory directory2 = FSDirectory.open(path2);

    List<MyDoc> parsedDocs = DocParser.parse(collectionPath);
    if (parsedDocs == null || parsedDocs.isEmpty()) {
        System.out.println("Error parsing documents");
        return;
    }
    try {
        IndexWriterConfig config = new IndexWriterConfig(new WhitespaceAnalyzer());

        IndexWriter indexWriter = new IndexWriter(directory, config);
        FieldType type = new FieldType(TextField.TYPE_STORED);
        type.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS_AND_OFFSETS);
        type.setTokenized(true);
        type.setStored(true);
        type.setStoreTermVectors(true);
        type.setStoreTermVectorOffsets(true);
        type.setStoreTermVectorPositions(true);

        for (MyDoc doc : parsedDocs) {
            Indexer.indexDoc(indexWriter, doc, type);
        }
        String fieldName = "content";
        IndexReader reader = DirectoryReader.open(indexWriter);

        FieldValuesSentenceIterator fieldValuesSentenceIterator = new FieldValuesSentenceIterator(reader, fieldName);
        SentenceIterator sentenceIterator = new BasicLineIterator(collectionPath);
        Word2Vec vec;
        if (choice == 1) {
            vec = new Word2Vec.Builder()
                    .layerSize(100)
                    .windowSize(7)
                    .tokenizerFactory(new DefaultTokenizerFactory())
                    .elementsLearningAlgorithm(new SkipGram<>())
                    .iterate(fieldValuesSentenceIterator)
                    .build();
            vec.fit();
        } else {
            File gModel = new File( pathname: "S:\\Downloads\\7\\model.txt");
            vec = WordVectorSerializer.readWord2VecModel(gModel);
        }
```

Μόλις το νευρωνικό δίκτυο εκπαιδευτεί με τα δεδομένα της συλλογής υπολογίζουμε τα διανύσματα με εξομάλυνση κατά TF-IDF των ερωτημάτων και των κειμένων. Με αυτόν τον τρόπο μπορούμε για κάθε κείμενο να υπολογίσουμε την ομοιότητα του με το ερώτημα βάσει των διανυσμάτων που υπολογίσαμε και στη συνέχεια να τα προσθέσουμε σε ένα TreeMap ώστε να τα ταξινομήσουμε και να εμφανίσουμε αυτά που είναι πιο σχετικά κατά φθίνουσα σειρά. Για κάθε κείμενο υπολογίζουμε τη συνημιτονοειδή ομοιότητά του με το ερώτημα μέσω των διανυσμάτων και στη συνέχεια τα ταξινομούμε βάσει των σκορ τους. Από αυτά κρατάμε τα πρώτα k για να εμφανίσουμε στον χρήστη. Για τον IndexSearcher χρησιμοποιούμε το

ClassicSimilarity καθώς δεν μας ενδιαφέρει το αποτέλεσμα που επιστρέφει η Lucene και το αγνοούμε.
Χρησιμοποιούμε επίσης έναν spellchecker που μας παρέχει η Lucene, ώστε εάν το μοντέλο δεν έχει μάθει κάποιον όρο που συναντάμε σε ένα ερώτημα (άρα θεωρείται άγνωστος), να βρούμε παρόμοιους (γνωστούς) όρους με αυτόν και να υπολογίσουμε το word embedding του. Έτσι βοηθάμε περεταίρω το σύστημα να βρει συναφή κείμενα αφού δεν αγνοούμε απλά τους άγνωστους όρους.

```
SpellChecker spellChecker = new SpellChecker(directory2);
LuceneDictionary spelldict = new LuceneDictionary(reader, field: "content");
IndexWriterConfig config2 = new IndexWriterConfig(new WhitespaceAnalyzer());
spellChecker.indexDictionary(spelldict, config2, fullMerge: false);
```

```
IndexSearcher searcher = new IndexSearcher(reader);
searcher.setSimilarity(new WordEmbeddingsSimilarity(vec, fieldName, WordEmbeddingsSimilarity.Smoothing.TF_IDF));
```

```
QueryParser parser = new QueryParser(fieldName, new WhitespaceAnalyzer());
System.out.println(queryString);
Query query = parser.parse(queryString);

TopDocs hits = searcher.search(query, TOP_RESULTS);
Map<Double, Integer> scores = new HashMap<>();
for (int i = 0; i < hits.scoreDocs.length; i++) {
    ScoreDoc scoreDoc = hits.scoreDocs[i];

    writer.println(qid + " 0 " + scoreDoc.doc + " 0 " + scoreDoc.score + " 1");
    System.out.println(qid + " 0 " + scoreDoc.doc + " 0 " + scoreDoc.score + " 1");
    Explanation ex = searcher.explain(query, scoreDoc.doc);
    System.out.println(ex);
```

Τέλος αποθηκεύουμε το μοντέλο που παρήγαγε το Word2Vec για μελλοντική χρήση.

Τα αποτελέσματα που λαμβάνουμε τρέχοντας το trec_eval με το results.text που πήραμε από την εκτέλεση του κώδικα για τις τιμές k = 20, 30 και 50 είναι τα εξής:

k	20	30	50
MAP	0.0029	0.0030	0.0037
P_5	0.0154	0.0154	0.0192
P_10	0.0115	0.0173	0.0192
P_15	0.0141	0.0192	0.0192
P_20	0.0144	0.0163	0.0163

runid	all	1
num_q	all	52
num_ret	all	1560
num_rel	all	796
num_rel_ret	all	24
map	all	0.0030
gm_map	all	0.0001
Rprec	all	0.0151
bpref	all	0.0212
recip_rank	all	0.0394
iprec_at_recall_0.00	all	0.0409
iprec_at_recall_0.10	all	0.0095
iprec_at_recall_0.20	all	0.0000
iprec_at_recall_0.30	all	0.0000
iprec_at_recall_0.40	all	0.0000
iprec_at_recall_0.50	all	0.0000
iprec_at_recall_0.60	all	0.0000
iprec_at_recall_0.70	all	0.0000
iprec_at_recall_0.80	all	0.0000
iprec_at_recall_0.90	all	0.0000
iprec_at_recall_1.00	all	0.0000
P_5	all	0.0154
P_10	all	0.0173
P_15	all	0.0192
P_20	all	0.0163
P_30	all	0.0154
P_100	all	0.0046
P_200	all	0.0023
P_500	all	0.0009
P_1000	all	0.0005

runid	all	1
num_q	all	52
num_ret	all	1040
num_rel	all	796
num_rel_ret	all	15
map	all	0.0029
gm_map	all	0.0000
Rprec	all	0.0105
bpref	all	0.0142
recip_rank	all	0.0352
iprec_at_recall_0.00	all	0.0395
iprec_at_recall_0.10	all	0.0056
iprec_at_recall_0.20	all	0.0000
iprec_at_recall_0.30	all	0.0000
iprec_at_recall_0.40	all	0.0000
iprec_at_recall_0.50	all	0.0000
iprec_at_recall_0.60	all	0.0000
iprec_at_recall_0.70	all	0.0000
iprec_at_recall_0.80	all	0.0000
iprec_at_recall_0.90	all	0.0000
iprec_at_recall_1.00	all	0.0000
P_5	all	0.0154
P_10	all	0.0115
P_15	all	0.0141
P_20	all	0.0144
P_30	all	0.0096
P_100	all	0.0029
P_200	all	0.0014
P_500	all	0.0006
P_1000	all	0.0003

runid	all	1
num_q	all	52
num_ret	all	2600
num_rel	all	796
num_rel_ret	all	39
map	all	0.0037
gm_map	all	0.0001
Rprec	all	0.0155
bpref	all	0.0326
recip_rank	all	0.0424
iprec_at_recall_0.00	all	0.0438
iprec_at_recall_0.10	all	0.0130
iprec_at_recall_0.20	all	0.0030
iprec_at_recall_0.30	all	0.0000
iprec_at_recall_0.40	all	0.0000
iprec_at_recall_0.50	all	0.0000
iprec_at_recall_0.60	all	0.0000
iprec_at_recall_0.70	all	0.0000
iprec_at_recall_0.80	all	0.0000
iprec_at_recall_0.90	all	0.0000
iprec_at_recall_1.00	all	0.0000
P_5	all	0.0192
P_10	all	0.0192
P_15	all	0.0192
P_20	all	0.0163
P_30	all	0.0154
P_100	all	0.0075
P_200	all	0.0038
P_500	all	0.0015
P_1000	all	0.0008

2ο σκέλος

Σε αυτό το σκέλος φορτώνουμε το προεκπαιδευμένο μοντέλο που δίνεται και παρατηρούμε τα αποτελέσματα.

k	20	30	50
MAP	0.0025	0.0027	0.0036
P_5	0.0115	0.0115	0.0115
P_10	0.0115	0.0115	0.0115
P_15	0.0128	0.0128	0.0128
P_20	0.0135	0.0135	0.0135

```
runid                   all     1
num_q                   all     52
num_ret                 all     1092
num_rel                 all     796
num_rel_ret             all     14
map                     all     0.0025
gm_map                  all     0.0001
Rprec                   all     0.0079
bpref                   all     0.0205
recip_rank              all     0.0343
iprec_at_recall_0.00    all     0.0343
iprec_at_recall_0.10    all     0.0115
iprec_at_recall_0.20    all     0.0011
iprec_at_recall_0.30    all     0.0000
iprec_at_recall_0.40    all     0.0000
iprec_at_recall_0.50    all     0.0000
iprec_at_recall_0.60    all     0.0000
iprec_at_recall_0.70    all     0.0000
iprec_at_recall_0.80    all     0.0000
iprec_at_recall_0.90    all     0.0000
iprec_at_recall_1.00    all     0.0000
P_5                     all     0.0115
P_10                    all     0.0115
P_15                    all     0.0128
P_20                    all     0.0135
P_30                    all     0.0090
P_100                   all     0.0027
P_200                   all     0.0013
P_500                   all     0.0005
P_1000                  all     0.0003
```

```
runid                   all     1
num_q                   all     52
num_ret                 all     1612
num_rel                 all     796
num_rel_ret             all     19
map                     all     0.0027
gm_map                  all     0.0001
Rprec                   all     0.0089
bpref                   all     0.0262
recip_rank              all     0.0370
iprec_at_recall_0.00    all     0.0370
iprec_at_recall_0.10    all     0.0121
iprec_at_recall_0.20    all     0.0011
iprec_at_recall_0.30    all     0.0000
iprec_at_recall_0.40    all     0.0000
iprec_at_recall_0.50    all     0.0000
iprec_at_recall_0.60    all     0.0000
iprec_at_recall_0.70    all     0.0000
iprec_at_recall_0.80    all     0.0000
iprec_at_recall_0.90    all     0.0000
iprec_at_recall_1.00    all     0.0000
P_5                     all     0.0115
P_10                    all     0.0115
P_15                    all     0.0128
P_20                    all     0.0135
P_30                    all     0.0122
P_100                   all     0.0037
P_200                   all     0.0018
P_500                   all     0.0007
P_1000                  all     0.0004
```

```
runid                   all     1
num_q                   all     52
num_ret                 all     2652
num_rel                 all     796
num_rel_ret             all     40
map                     all     0.0036
gm_map                  all     0.0002
Rprec                   all     0.0107
bpref                   all     0.0449
recip_rank              all     0.0405
iprec_at_recall_0.00    all     0.0433
iprec_at_recall_0.10    all     0.0163
iprec_at_recall_0.20    all     0.0011
iprec_at_recall_0.30    all     0.0000
iprec_at_recall_0.40    all     0.0000
iprec_at_recall_0.50    all     0.0000
iprec_at_recall_0.60    all     0.0000
iprec_at_recall_0.70    all     0.0000
iprec_at_recall_0.80    all     0.0000
iprec_at_recall_0.90    all     0.0000
iprec_at_recall_1.00    all     0.0000
P_5                     all     0.0115
P_10                    all     0.0115
P_15                    all     0.0128
P_20                    all     0.0135
P_30                    all     0.0122
P_100                   all     0.0077
P_200                   all     0.0038
P_500                   all     0.0015
P_1000                  all     0.0008
```