2019-2020

Συστήματα Ανάκτησης Πληροφοριών

Προγραμματιστική Εργασία

Επέκταση Ερωτημάτων με Συνώνυμους Όρους για τη Βελτίωση των Αποτελεσμάτων της Ανάκτησης

3η ΦΑΣΗ- Επέκταση ερωτήματος με συνώνυμα από το word2vec

Δημιουργία του νευρωνικού δικτύου

Αρχικά , ξεκίνησα δημιουργώντας μια μέθοδο `trainModel` στην κλάση `Indexer` η οποία εκτελείται μετά την κατασκευή του ευρετηρίου. Στη συγκεκριμένη μέθοδο το δίκτυο εκπαιδεύεται στη IR_2020_clean.txt και περιλαμβάνει έναν `BasicLineIterator.` Οι παράμετροι που δοκίμασα είναι για `layersize`= 100 , 150, μέγεθος παραθύρου 3 ,4 ,5 και 8 και τους δύο αλγόριθμους `skipGram` και `CBOW.` Έπειτα , αποθηκεύω το μοντέλο με όνομα myModel.txt και την εντολή:

```
WordVectorSerializer.writeWord2VecModel(vec, "docs/myModel.txt");
```

για να μπορώ να το χρησιμοποιήσω κατά το query parsing.

Χρήση του νευρωνικού δικτύου για δημιουργία συνωνύμων στα ερωτήματα

Δημιουργώντας την μέθοδο `customAnalyzerForQueryExpansion` στη κλάση `Searcher`, φτιάχνω έναν `Analyzer` ο οποίος χρησιμοποιεί filters για την δημιουργία του επαυξημένου ερωτήματος. Μέσα στην μέθοδο αυτή , χρησιμοποίησα τον `StandardTokenizer` και έπειτα με την σειρά `StandardFilter` , `LowercaserFilter` , `StopwordsFilter` και `W2VFilter`. Η κλάση `W2VFilter` λαμβάνει ως παράμετρο το νευρωνικό δίκτυο που έχει δημιουργηθεί διαβάζοντας το μέσω της εντολής:

```
Word2Vec vec = WordVectorSerializer.readWord2VecModel("docs/myModel.txt");
```

Και επίσης το accuracy(`minAcc=0.9`) που θέλουμε να ληφθεί υπόψιν (πόση ομοιότητα έχουν οι λέξεις).

Δοκιμάζοντας και σε αυτό το σημείο διαφορετικούς Tokenizers και Filters και διαφορετικό accuracy κατέληξα στον παραπάνω συνδυασμό που μου έδινε το μεγαλύτερο mean average precision . Ενδεικτικά παρουσιάζω μερικά αποτελέσματα για διαφορικούς Tokenizers και Filters και accuracy και διαφορετικές παραμέτρους νευρωνικού δικτύου

micAcc=**0.9** , αλγόριθμος = Skipgram , layersize= 150 , windowsize=3 , Filters & Tokinizers όπως παρουσιάστηκαν παραπάνω

```
num_rel_ret             all     78
map                     all     0.2557
```

Εικόνα 1: Για K= 50 και όλα τα ερωτήματα,Skipgram

micAcc=**0.85** , αλγόριθμος = Skipgram , layersize= 150 , windowsize=3 , Filters & Tokinizers όπως παρουσιάστηκαν παραπάνω

```
num_rel_ret             all     79
map                     all     0.2441
```

Εικόνα 2: Για K= 50 και όλα τα ερωτήματα,Skipgram

micAcc=**0.85** , αλγόριθμος = CBOW, layersize= 150 , windowsize=4 , Filters & Tokinizers όπως φαίνονται παρακάτω

```
num_rel_ret             all     68
map                     all     0.1958
```

Εικόνα 3:Για K= 50 και όλα τα ερωτήματα ,CBOW

```
Tokenizer tokenizer = new WhitespaceTokenizer();
double minAcc = 0.85;
TokenStream stream =new LowerCaseFilter(tokenizer);
stream = new StopFilter(stream, StopAnalyzer.ENGLISH_STOP_WORDS_SET);
stream = new W2VSynonymFilter(stream, vec, minAcc);
```

Εικόνα 4:Filters και Tokenizers

micAcc=**0.85** , αλγόριθμος = CBOW, layersize= 150 , windowsize=4 , Filters & Tokinizers όπως φαίνονται παρακάτω

```
num_rel_ret             all     79
map                     all     0.2441
```

Εικόνα 5::Για K= 50 και όλα τα ερωτήματα ,CBOW

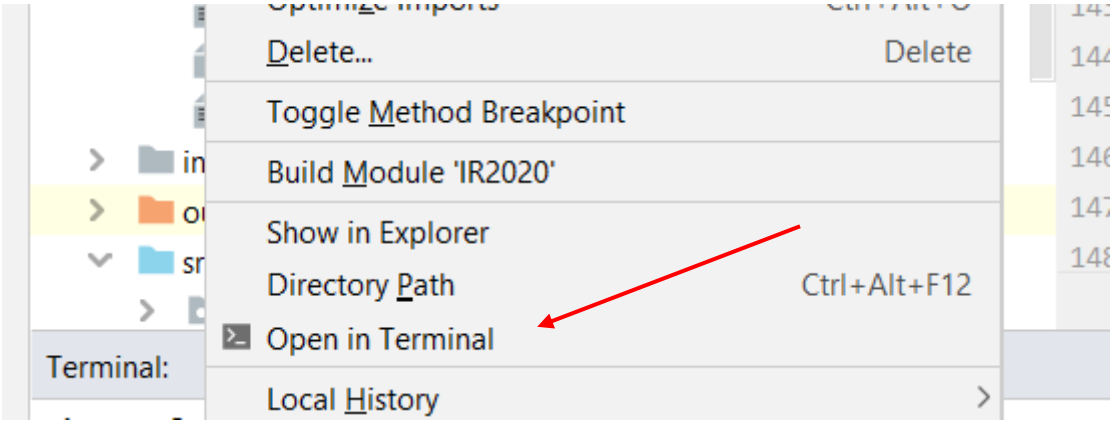
```
Tokenizer tokenizer = new ClassicTokenizer();
double minAcc = 0.85;

TokenStream stream = new ClassicFilter(tokenizer);
stream =new LowerCaseFilter(tokenizer);
stream = new StopFilter(stream, StopAnalyzer.ENGLISH_STOP_WORDS_SET);
stream = new W2VSynonymFilter(stream, vec, minAcc);
```

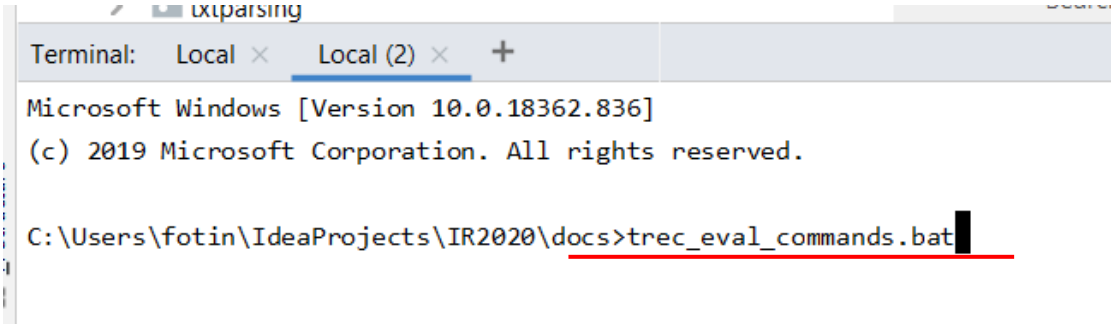
Εικόνα 6:Filters και Tokenizers

Τα ονόματα των νέων αρχείων που δημιουργήθηκαν είναι Newresultsk.txt όπου k=20,30,50 και βρίσκονται στο φάκελο docs. Επίσης δημιούργησα ένα batch file με όνομα trec_eval_commands.bat ώστε να τρέξω τις κατάλληλες εντολές για το trec_eval ,το οποίο βρίσκεται στο φάκελο docs. Για να τρέξει σωστά το αρχείο πρέπει να βρίσκονται στον φάκελο docs το exe του trec_eval και τα αρχεία που βγήκαν ως αποτέλεσμα του Searcher.java(Newresults20.txt etc). Οι απαντήσεις του trec_eval για κάθε query είναι στον φάκελο docs/answers/asnwerk.txt όπου k=5,10,15,20,30,50.

(Με IntelliJ)Για πιο γρήγορο run του batch file πατάμε δεξί κλικ στο φάκελο *docs* και έπειτα *Open in Terminal*.



Έπειτα, πληκτρολογούμε τη παρακάτω εντολή και έχουμε τη δημιουργία των αρχείων στο φάκελο docs/answers: `trec_eval_commands.bat`



Παρακάτω παραθέτω τον ολοκληρωμένο πίνακα για κάθε ερώτημα με τις απαντήσεις του trec_eval με την μέθοδο που μου επέστρεψε τα υψηλότερα αποτελέσματα:

Query	k	avgpre@k(1η φάση)	avgpre@k(2η φάση)	avgpre@k(3η φάση)	num_rel_ret(1η φάση)	num_rel_ret(2η φάση)	num_rel_ret(3η φάση)	map@k(1η φάση)	map@k(2η φάση)	map@k(3η φάση)
Q01	5	0.1698	0.2941	0.2941	4	5	5			
	10	0.3857	0.5294	0.5294	8	9	9			
	15	0.5265	0.6753	0.6178	11	12	11			
	20				12	13	12	0.5706	0.7202	0.6549
	30				14	15	12	0.6473	0.7978	0.6549
	50				15	16	14	0.6741	0.8247	0.6966
Q02	5	0.1389	0.1389	0.1167	2	2	2			
	10	0.1746	0.1746	0.1167	3	3	2			
	15	0.1746	0.1746	0.1359	3	3	3			
	20				3	3	3	0.1746	0.1746	0.1359
	30				3	3	3	0.1746	0.1746	0.1359
	50				3	3	3	0.1746	0.1746	0.1359
Q03	5	0.2536	0.2536	0.1143	4	4	3			
	10	0.3743	0.3743	0.2129	6	6	5			
	15	0.3743	0.3743	0.2871	6	6	7			
	20				8	8	8	0.4322	0.4322	0.3228
	30				10	10	11	0.4798	0.4789	0.4145
	50				14	14	12	0.5689	0.5687	0.4431

Q04	5	0.0464	0.0464	0.0238	2	2	1			
	10	0.0464	0.0464	0.0476	2	2	2			
	15	0.0607	0.0607	0.0655	3	3	3			
	20				3	3	3	0.0607	0.0607	0.0655
	30				3	3	3	0.0607	0.0607	0.0655
	50				4	4	4	0.0694	0.0696	0.0732
Q05	5	0.1	0.0896	0.0875	3	3	2			
	10	0.1	0.0896	0.0875	3	3	2			
	15	0.1	0.0896	0.1176	3	3	4			
	20				5	5	5	0.1295	0.1207	0.1349
	30				9	9	6	0.198	0.1892	0.1528
	50				12	12	9	0.2489	0.2401	0.1925
Q06	5	0.0263	0.0000	0.0737	1	0	2			
	10	0.0263	0.0000	0.1123	1	0	4			
	15	0.0263	0.0000	0.1311	1	0	5			
	20				1	0	6	0.0263	0.0000	0.1508
	30				1	0	8	0.0263	0.0000	0.1875
	50				6	0	10	0.0504	0.0000	0.2107
Q07	5	0.0625	0.0625	0.0875	1	1	2			
	10	0.0764	0.0764	0.0875	2	2	2			

	15	0.0934	0.0934	0.1186	3	3	4			
	20				3	3	6	0.0934	0.0934	0.1569
	30				9	9	8	0.1868	0.1868	0.1962
	50				12	12	10	0.2379	0.2382	0.2292
Q08	5	0.3571	0.3571	0.1964	5	5	3			
	10	0.5714	0.5714	0.1964	8	8	3			
	15	0.5714	0.5714	0.1964	8	8	3			
	20				11	10	3	0.6929	0.6489	0.1964
	30				11	11	3	0.6929	0.6846	0.1964
	50				11	11	4	0.6929	0.6846	0.2024
Q09	5	0.131	0.0952	0.0952	3	2	2			
	10	0.15	0.1131	0.1131	4	2	3			
	15	0.1698	0.1426	0.1131	5	5	3			
	20				5	7	4	0.1698	0.1800	0.1243
	30				7	9	4	0.1927	0.2106	0.1243
	50				9	12	10	0.2107	0.2591	0.1786
Q10	5	0.05	0.0250	0.0250	1	1	1			
	10	0.0722	0.0583	0.0536	2	2	2			
	15	0.0722	0.0583	0.0786	2	2	3			
	20				2	3	3	0.0722	0.0760	0.0786

	30				3	3	3	0.0826	0.0760	0.0786
	50				3	3	3	0.0826	0.0760	0.0786
ALL	5	0.1336	0.1362	0.1114	26	25	23			
	10	0.1977	0.2034	0.1557	39	38	34			
	15	0.2169	0.2240	0.1862	45	45	46			
	20				53	55	53	0.2422	0.2507	0.2021
	30				70	72	61	0.2742	0.2859	0.2207
	50				89	87	79	0.301	0.3136	0.2441

Μία εμφανή διαφορά στην συγκεκριμένη φάση , παρότι σε μερικά ερωτήματα δεν έχουμε πετύχει καλά αποτελέσματα, στο **ερώτημα Q06**(«Mobility-as-a-Service tools») έχουμε *καλύτερα* αποτελέσματα σε σχέση με τις άλλες προηγούμενες φάσεις . Αυτό που βλέπουμε για το ερώτημα Q06 είναι πως το τελικό ερώτημα μετά από την εφαρμογή των φίλτρων και των συνωνύμων τα οποία έχει υπολογίσει είναι τα εξής :

```
mobility diborylate stably-inherited service service expertesfri leicester tools tools descriptor profilometers
```

Εικόνα 7: Τελικό ερώτημα για το Q06 μετά από εφαρμογή συνωνύμων

Ενώ κατά τη δεύτερη φάση τα συνώνυμα που είχαν προκύψει ήταν :

```
Searching for: mobility-as-a-servic Synonym(cock creature dick instrument pecker peter prick puppet putz shaft tool)
4818 total matching documents
```

Βλέπουμε μια βελτίωση διότι τα συνώνυμα που προέκυψαν από το νευρωνικό δίκτυο ταιριάζουν περισσότερο στη εύρεση κειμένων που να είναι συναφή με τα ερωτήματα.

Ακόμη στο ερώτημα Q07(“fragmentation of IoT through federation”) παρατηρούμε μία μικρή διαφορά ως προς το mean average precision σε κ<30 που παρατηρείται να είναι μεγαλύτερο και να έχει βρει περίπου 1 κείμενο επιπλέον ως συναφές.

```
fragmentation tetragrammaton tnyear iot iot reoperation seperation through through nanodesorption unioneuropean federation federation adrien alderney
```

Εικόνα 8: Τελικό ερώτημα μετά από συνώνυμα (3η φάση)

Στα υπόλοιπα ερωτήματα δεν βλέπουμε κάποια εμφανή βελτίωση ,σε κάποιες περιπτώσεις είναι χαμηλότερα τα αποτελέσματα και από τις δύο φάσεις , το οποίο ίσως εξηγείται στο γεγονός ότι το σώμα κειμένων που έχει δοθεί είναι μικρό και έτσι το δίκτυο δεν καταφέρνει να εκπαιδευτεί σωστά .

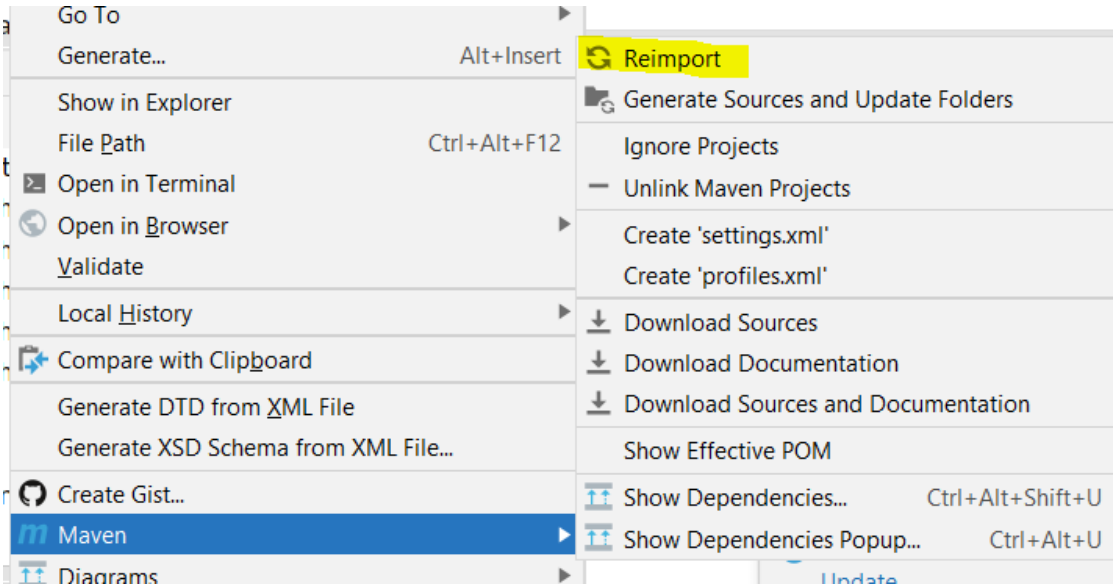
Οδηγίες φόρτωσης:

Αρχικά , κάνουμε unzip τον φάκελο 3_IR.zip που περιέχει το project.

Με IntelliJ πατάμε Open και έπειτα επιλέγουμε τον φάκελο 3_IR .

Μέσα στον φάκελο docs τοποθετούμε τα αρχεία `qrels_new_utf8.txt , queries.txt, documents.txt , IR_2020_clean.txt` και τα αρχεία για την εκτέλεση του trec_eval.

Αφήνουμε το IntelliJ να κάνει resolve τα Maven dependencies. Αν δεν γίνει αυτόματα πατάμε μέσα στο pom.xml δεξί κλικ > Maven > Reimport



Εικόνα 9: Import dependencies

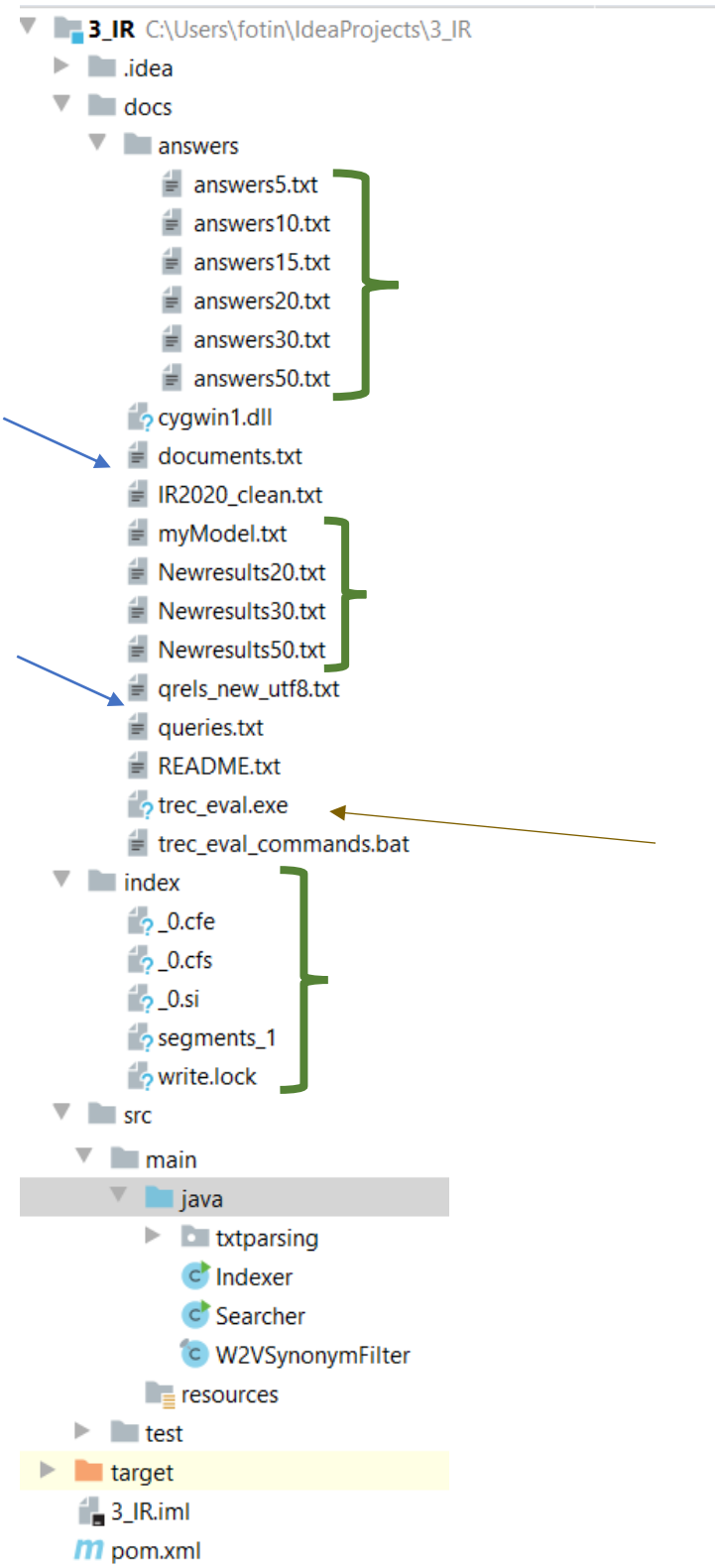
Ξεκινάμε με το τρέξιμο του Indexer(src>main>java) για να δημιουργηθεί το ευρετήριο και για να εκπαιδευτεί το μοντέλο.

Μετά την εκτέλεση του Indexer θα έχει δημιουργηθεί μέσα στο φάκελο index το ευρετήριο και μέσα στο φάκελο docs το αρχείο myModel.txt

Έπειτα εκτελούμε τον Searcher(src>main>java) που μετά την εκτέλεση θα έχουμε τα αποτελέσματα στο φάκελο docs με όνομα Newresultsk.txt όπου k = 20,30,50.

Τέλος , εκτελούμε όπως παρουσιάστηκε και στη σελίδα 3 το αρχείο trec_eval_commands.bat όπου δημιουργεί στον φάκελο docs>answers τα αποτελέσματα που προκύπτουν.

Η τελική διάταξη που πρέπει να προκύψει είναι η παρακάτω:



Εικόνα 10:Τελική διάταξη φακέλων και αρχείων για τη σωστή εκτέλεση