2019-2020

Συστήματα Ανάκτησης Πληροφοριών

Προγραμματιστική Εργασία

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

2η ΦΑΣΗ-Επέκταση ερωτήματος με συνώνυμα από το WordNet

Αρχικά, ξεκινώντας, μαζί με τ ον κώδικα όπως τον χρησιμοποίησα στην 1η φάση της εργασίας , χρησιμοποίησα τον original κώδικα ο οποίος έχει δοθεί από τη διδάσκουσα κ. Κυριακοπούλου δοκιμάζοντας τα αρχικά συνώνυμα όπως έχουν ανέβει στο αρχείο wn_s.pl.

```
CustomAnalyzer.Builder builder = CustomAnalyzer.builder()
        .withTokenizer(StandardTokenizerFactory.class)
        .addTokenFilter(StandardFilterFactory.class)
        .addTokenFilter(EnglishPossessiveFilterFactory.class)
        .addTokenFilter(LowerCaseFilterFactory.class)
        .addTokenFilter(StopFilterFactory.class)
        .addTokenFilter(PorterStemFilterFactory.class)
        .addTokenFilter(SynonymGraphFilterFactory.class, sffargs);
```

Εικόνα 1:original code

Τα αποτελέσματα που εξάχθηκαν από την παραπάνω μέθοδο *ενδεικτικά* για k =50 πρώτα ανακτηθέντα κείμενα σε κάθε ερώτημα παρουσιάζονται παρακάτω:

	map	num_rel
Q01	0.5136	12
Q02	0.0663	3
Q03	0.5687	14
Q04	0.0658	4
Q05	0.2401	12
Q06	0.0011	1

	map	num_rel
Q07	0.1747	10
Q08	0.4641	10
Q09	0.0272	2
Q10	0.0175	2
all	0.2139	70

Ο πίνακας που παρουσιάζεται παρακάτω δείχνει τα αποτελέσματα για mean average precision(map) και τον αριθμό σχετικών ανακτηθέντων κειμένων (num_rel_ret) για τα κ=20,30,50 πρώτα ανακτηθέντα κείμενα χωρίς καμία αλλαγή στον κώδικα (original code) και χωρίς διαγραφή των ρημάτων και των επιρρημάτων από τα συνώνυμα(Not removing verbs and adverbs) δείχνοντας τις αλλαγές μεταξύ των αποτελεσμάτων στην κλάση όπου χρησιμοποιείται για τα συνώνυμα.

	Not Removing verbs and adverbs			
	original code			
	SynonymGraphFilterFactory		SynonymFilterFactory	
k	num_rel_ret	map	num_rel_ret	map
20	42	0.1681	46	0.1939
30	53	0.1888	54	0.2078
50	70	0.2139	67	0.2305

Στη προσπάθεια να αυξηθεί ο αριθμός των σχετικών ανακτηθέντων κειμένων αλλά και το mean average precision δοκίμασα να αλλάξω την κλάση για την εισαγωγή συνωνύμων χρησιμοποιώντας την κλάση SynonymFilterFactory ,η οποία είναι deprecated έκδοση της κλάσης SynonymGraphFilterFactory από την lucene, όμως τα νούμερα δεν ανέβηκαν. Έπειτα ,μετά από αρκετές δοκιμές ,διαγράφοντας το TokenFilter με την κλάση StandardFilterFactory , αλλάζοντας την κλάση StandardTokenizer σε WhiteTokenizer και προσθέτοντας το Token Filter RemoveDuplicatesTokenFilterFactory βλέπουμε μια **βελτίωση** στα αποτελέσματα. Ο whiteTokenizer χωρίζει

το text stream με βάση τα κενά ανάμεσα στα tokens και επιστρέφει τα tokens χωρίς whitespaces.
Παράδειγμα:

In: "To be, or what?"

Out: "To", "be,", "or", "what?"

Εικόνα 2:Είσοδος και Έξοδος σε WhitespaceTokenizer

Ο StandardTokenizer χρησιμοποιεί ως delimeter και αυτός τα κενά(whitespaces) αλλά και τα σημεία στίξης τα οποία τα αφαιρεί ,με διάφορες εξαιρέσεις όπως παρουσιάζονται και στο documentation της lucene(https://lucene.apache.org/solr/guide/6_6/tokenizers.html).Επίσης ,χωρίζει τα tokens στις παύλες
Παράδειγμα:

In: "Please, email john.doe@foo.com by 03-09, re: m37-xq."

Out: "Please", "email", "john.doe", "foo.com", "by", "03", "09", "re", "m37", "xq"

Εικόνα 3:Είσοδος και Έξοδος του StandardTokenizer

Πιθανότατα λόγω της αφαίρεσης διάφορων χρήσιμων σημείων στίξεων ,τα οποία να είναι χρήσιμα μέσα στα κείμενα , να οφείλονται τα καλύτερα αποτελέσματα στον WhitespaceTokenizer. Επίσης ,το token filter RemoveDuplicatesTokenFilterFactory χρησιμοποιήθηκε διότι θεώρησα χρήσιμο να διαγράφονται οι διπλότυποι όροι όπου εμφανίζονται μετά την παραγωγή των συνώνυμων όρων καθώς όπως παρουσιάζεται στο παράδειγμα παρακάτω έχουμε την παραγωγή αρκετών διπλότυπων όρων.

Παράδειγμα:

```
<analyzer type="query">
  <tokenizer class="solr.StandardTokenizerFactory"/>
  <filter class="solr.SynonymGraphFilterFactory" synonyms="synonyms.txt"/>
  <filter class="solr.EnglishMinimalStemFilterFactory"/>
  <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
</analyzer>
```

In: "Watch TV"

Tokenizer to Synonym Filter: "Watch"(1) "TV"(2)

Synonym Filter to Stem Filter: "Watch"(1) "Television"(2) "Televisions"(2) "TV"(2) "TVs"(2)

Stem Filter to Remove Dups Filter: "Watch"(1) "Television"(2) "Television"(2) "TV"(2) "TV"(2)

Out: "Watch"(1) "Television"(2) "TV"(2)

Εικόνα 4:Είσοδος και Έξοδος του Token Filter RemoveDuplicates

(Στο παράδειγμα παραπάνω χρησιμοποιείται και ο EnglishMinimalStemFilterFactory τον οποίο χρησιμοποίησα αλλά τα αποτελέσματα παρέμειναν ίδια)

Ο πίνακας που παρουσιάζεται παρακάτω δείχνει τα αποτελέσματα για mean average precision(map) και τον αριθμό σχετικών ανακτηθέντων κειμένων (num_rel_ret) για τα κ=20,30,50 πρώτα ανακτηθέντα κείμενα μέσω της χρήσης του whitetokenizer(+RemovedublicateTokenFilter)

	Not Removing verbs and adverbs			
	WhiteTokenizer			
	SynonymGraphFilterFactory		SynonymFilterFactory	
k	num_rel_ret	map	num_rel_ret	map
20	49	0.1899	50	0.2170
30	66	0.2232	61	0.2391
50	85	0.2572	76	0.2655

Ακόμη, διευρύνοντας την προσπάθεια μου για αύξηση των αριθμών των αποτελεσμάτων διέγραψα τα ρήματα από τα συνώνυμα του wordnet καθώς, όπως και στο παράδειγμα της κ. Κυριακοπούλου ,δεν πετυχαίνει πολύ καλά τα συνώνυμα των όρων(π.χ. dog->(v.)chase). Κατά τη διαδικασία της εφαρμογής των νέων συνωνύμων , παρατήρησα πως εάν διαγράψουμε με το ίδιο σκεπτικό και τα επιρρήματα ,καθώς ούτε αυτά αντιπροσωπεύουν σωστά μερικούς από τους όρους , έχουμε ακόμη καλύτερα αποτελέσματα. Για την διαγραφή των ρημάτων και των επιρρημάτων έτρεξα τον παρακάτω κώδικα ο οποίος βρίσκεται στην κλάση TXTparsing.

```
public static void main(String [] args) throws IOException {

    String txt_file = TXTparsing.readEntireFileIntoAString("docs/wn_s.pl");
    String [] line =txt_file.split( regex: "[\\r\\n]+");

    BufferedWriter out = new BufferedWriter(new FileWriter( fileName: "wn_s1.pl"));
        for(String l : line) {
            if(!l.contains(",v,") && !l.contains((",r,")))
                out.write( str: l+"\n");
        }
    out.close();
}
```

Εικόνα 5:Κώδικας για διαγραφή ρημάτων από το αρχείο wn_s.pl

Επομένως, με το ίδιο σκεπτικό συνέχισα και δοκίμασα αρκετές πρακτικές οι οποίες φαίνονται στον ακόλουθο συγκεντρωτικό πίνακα.

	Removing verbs and adverbs			
	original code			
	SynonymGraphFilterFactory		SynonymFilterFactory	
k	num_rel_ret	map	num_rel_ret	map
20	51	0.2297	53	0.2447
30	64	0.2536	67	0.2747
50	81	0.2798	82	0.2998

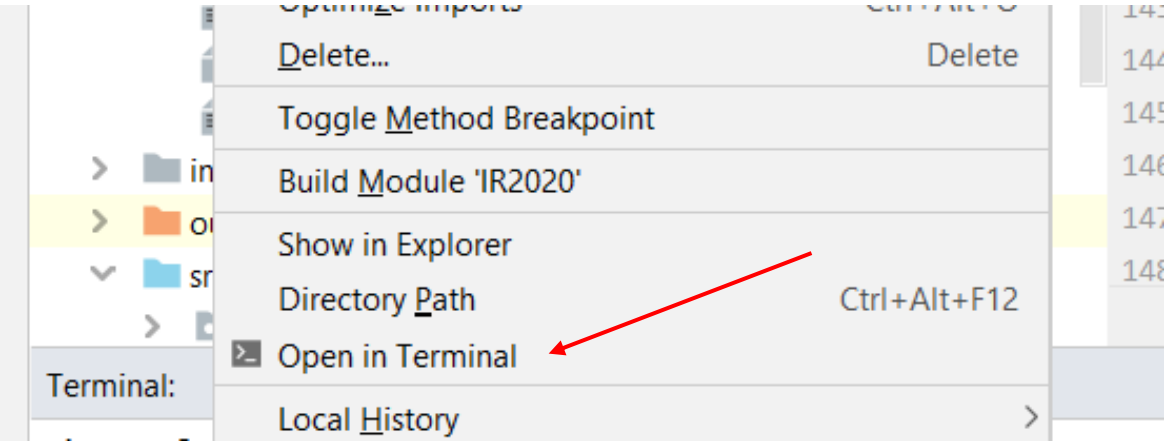
	Removing verbs and adverbs			
	WhiteTokenizer			
	SynonymGraphFilterFactory		SynonymFilterFactory	
k	num_rel_ret	map	**num_rel_ret**	**map**
20	54	0.2377	**55**	**0.2507**
30	70	0.2694	**72**	**0.2859**
50	86	0.2973	**87**	**0.3136**

Έτσι ,διαγράφοντας τα ρήματα και τα επιρρήματα , αντικαθιστώντας τον StandardTokenizer σε WhiteTokenize και το SynonymGraphFilterFactory σε SynonymFilterFactory και προσθέτοντας το RemoveDublicatesTokenFilterFactory έφτασα στα πιο υψηλά δυνατά αποτελέσματα συγκρίνοντας με τις υπόλοιπες μεθόδους που δοκίμασα.

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

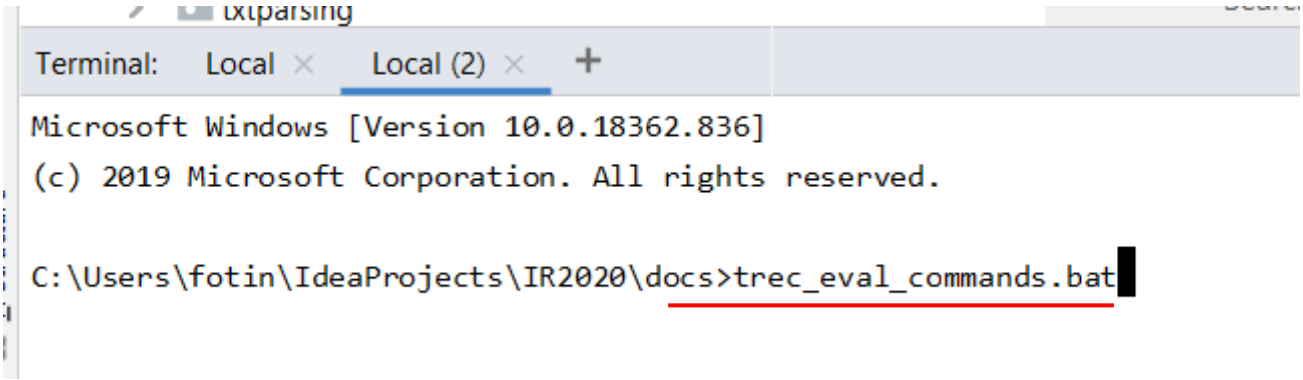
ώστε να τρέξω τις κατάλληλες εντολές για το trec_eval ,το οποίο βρίσκεται στο φάκελο docs. Για να τρέξει σωστά το αρχείο πρέπει να βρίσκονται στον φάκελο docs το exe του trec_eval και τα αρχεία που βγήκαν ως αποτέλεσμα του Searcher.java(Newresults20.txt etc). Οι απαντήσεις του trec_eval για κάθε query 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)	num_rel_ret(1η φάση)	num_rel_ret(2η φάση)	map@k(1η φάση)	map@k(2η φάση)
Q01	5	0.1698	0.2941	4	5		
	10	0.3857	0.5294	8	9		
	15	0.5265	0.6753	11	12		
	20			12	13	0.5706	0.7202
	30			14	15	0.6473	0.7978
	50			15	16	0.6741	0.8247
Q02	5	0.1389	0.1389	2	2		
	10	0.1746	0.1746	3	3		
	15	0.1746	0.1746	3	3		
	20			3	3	0.1746	0.1746
	30			3	3	0.1746	0.1746
	50			3	3	0.1746	0.1746
Q03	5	0.2536	0.2536	4	4		

	10	0.3743	0.3743	6	6		
	15	0.3743	0.3743	6	6		
	20			8	8	0.4322	0.4322
	30			10	10	0.4798	0.4789
	50			14	14	0.5689	0.5687
Q04	5	0.0464	0.0464	2	2		
	10	0.0464	0.0464	2	2		
	15	0.0607	0.0607	3	3		
	20			3	3	0.0607	0.0607
	30			3	3	0.0607	0.0607
	50			4	4	0.0694	0.0696
Q05	5	0.1	0.0896	3	3		
	10	0.1	0.0896	3	3		
	15	0.1	0.0896	3	3		
	20			5	5	0.1295	0.1207
	30			9	9	0.198	0.1892
	50			12	12	0.2489	0.2401
Q06	5	0.0263	0.0000	1	0		

	10	0.0263	0.0000	1	0		
	15	0.0263	0.0000	1	0		
	20			1	0	0.0263	0.0000
	30			1	0	0.0263	0.0000
	50			6	0	0.0504	0.0000
Q07	5	0.0625	0.0625	1	1		
	10	0.0764	0.0764	2	2		
	15	0.0934	0.0934	3	3		
	20			3	3	0.0934	0.0934
	30			9	9	0.1868	0.1868
	50			12	12	0.2379	0.2382
Q08	5	0.3571	0.3571	5	5		
	10	0.5714	0.5714	8	8		
	15	0.5714	0.5714	8	8		
	20			11	10	0.6929	0.6489
	30			11	11	0.6929	0.6846
	50			11	11	0.6929	0.6846
Q09	5	0.131	0.0952	3	2		

	10	0.15	0.1131	4	2		
	15	0.1698	0.1426	5	5		
	20			5	7	0.1698	0.1800
	30			7	9	0.1927	0.2106
	50			9	12	0.2107	0.2591
Q10	5	0.05	0.0250	1	1		
	10	0.0722	0.0583	2	2		
	15	0.0722	0.0583	2	2		
	20			2	3	0.0722	0.0760
	30			3	3	0.0826	0.0760
	50			3	3	0.0826	0.0760
ALL	5	0.1336	0.1362	26	25		
	10	0.1977	0.2034	39	38		
	15	0.2169	0.2240	45	45		
	20			53	55	0.2422	0.2507
	30			70	72	0.2742	0.2859
	50			89	87	0.301	0.3136

Τα πεδία όπου είναι κενά είναι ίδια με avgpre@k=map@k.

Όπως βλέπουμε υπάρχουν ελάχιστα ερωτήματα στα οποία έχει πετύχει η μέθοδος των συνωνύμων καλύτερο σκορ σε σχέση με τα αποτελέσματα της προηγούμενης φάσης της εργασίας όπως το ερώτημα Q01 και Q09 όπου οι συνώνυμοι όροι που έχουν αποδοθεί για το 1ο ερώτημα (multimodal travel services) είναι:

```
Searching for: multimod Synonym(change locomotion travel traveling travelling) Synonym(of servic) location
```

Και για το 9ο ερώτημα(cross-domain orchestration of services) είναι (χωρίς συνώνυμους όρους) :

```
Searching for: cross-domain orchestr servic
```

Το ερώτημα Q06 (Mobility-as-a-Service tools) έχει αποτύχει πλήρως και του έχουν αποδοθεί οι παρακάτω όροι :

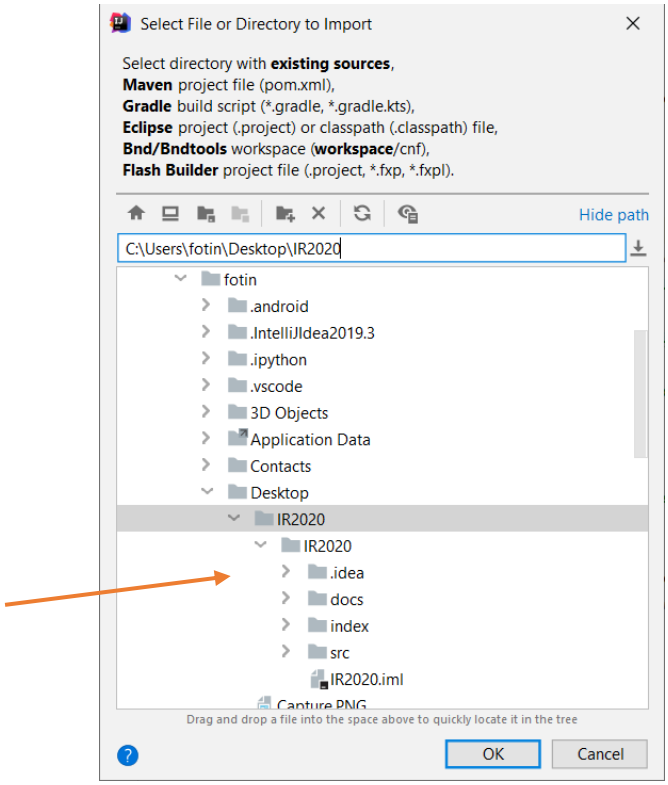
```
Searching for: mobility-as-a-servic Synonym(cock creature dick instrument pecker peter prick puppet putz shaft tool)
4818 total matching documents
```

Παρότι η μέθοδος όπου επέλεξα αποτυγχάνει στο συγκεκριμένο ερώτημα η μέθοδος με τον original code της κ. Κυριακοπούλου και διαγραφή των ρημάτων και επιρρημάτων μας δίνει για k=5,10,15,20 num_rel_ret=1 και map=0.0175, για k=30 num_rel_ret=2 και map=0.0212 ,για k=50 num_rel_ret=4 και map=0.0294 ,αλλά δεν προβλέπει καλα τα άλλα ερωτήματα όπου τα πήγε καλυτερα η μέθοδος που επέλεξα.

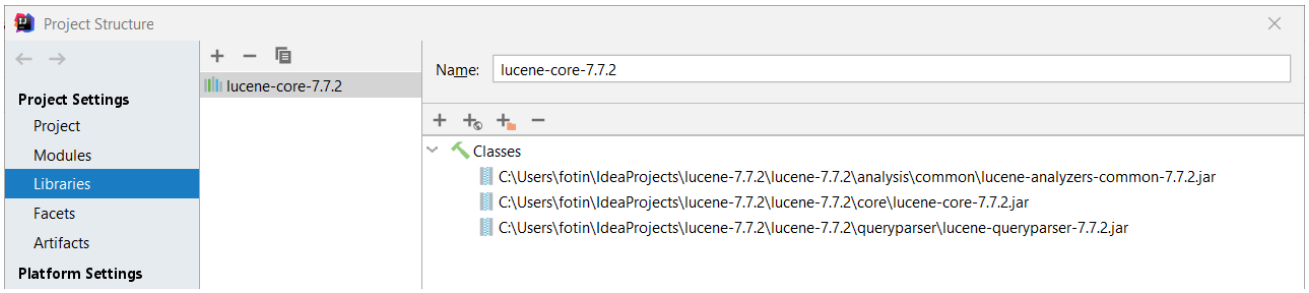
Οδηγίες Φόρτωσης

Προκειμένου να φορτώσετε την εργασία :

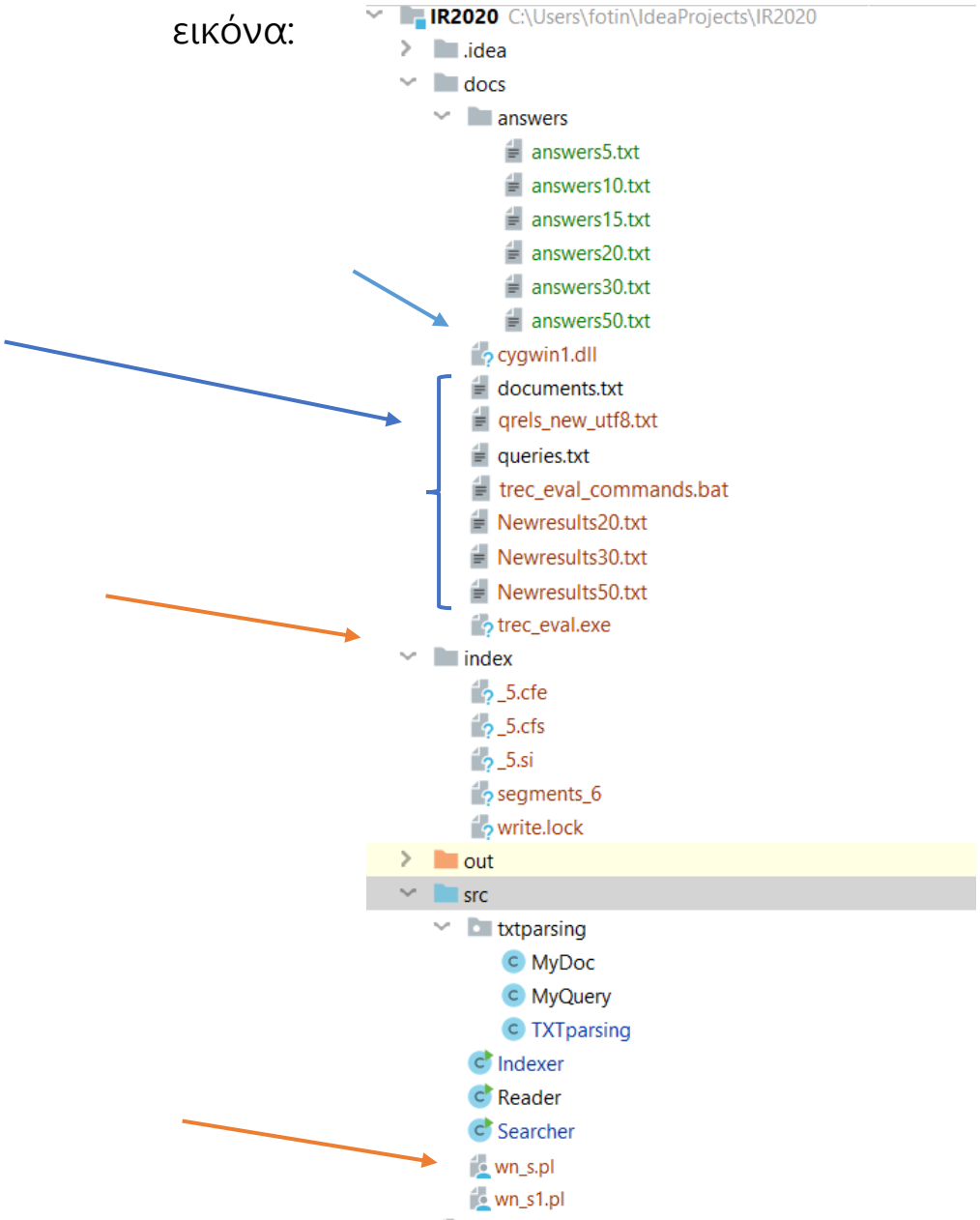
1. κάνετε unzip το αρχείο και μέσω του Intellij επιλέγετε να φορτωθεί File>New>Project from Existing Sources όπου από εκεί επιλέγετε το φάκελο που έγινε unzip.



Εάν δεν έχουν φορτωθεί οι βιβλιοθήκες αυτόματα , επιλέγετε CTLR+ALT+SHIFT+S και μέσω του project structure επιλέγετε libraries και εισάγετε την lucene όπως στο 2ο εργαστήριο.



2. Το ευρετήριο δεν έχει παραμείνει επομένως πρέπει να τρέξετε τον **Indexer,** αφού πρώτα στο φάκελο docs εισαχθούν τα αρχεία qrels.txt ,queries.txt ,documents.txt καθώς και αν θέλετε να τρέξετε και το εργαλείο trec_eval πρέπει να μπει στο φάκελο docs.
3. Επίσης για τη σωστή εκτέλεση με τα συνώνυμα χωρίς ρήματα και επιρρήματα πρέπει να εκτελεστεί το TXTparsing java αρχείο όπου το αρχείο wn_s.pl πρέπει να βρίσκεται μέσα στο φάκελο src και θα δημιουργηθεί το αρχείο wn_s1.pl στον ίδιο φάκελο χωρίς τα ρήματα και τα επιρρήματα.
4. Η τελική διάταξη του project πρέπει να είναι όπως στην παρακάτω εικόνα:



Η κλάση Reader υπάρχει ώστε να διαβάζω το ευρετήριο μου , δεν χρειάζεται σε κάποιο μέρος της εκτέλεσης.

Πηγές:

- https://lucene.apache.org/solr/guide/6_6/understanding-analyzers-tokenizers-and-filters.html